Computação Paraconsistente: Uma Abordagem Lógica à Computação Quântica

Juan Carlos Agudelo Agudelo

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

Novembro de 2009

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IFCH – UNICAMP
Bibliotecária: Maria Silvia Holloway – CRB2289

Ag91c

Agudelo, Juan Carlos Agudelo
Computação Paraconsistente: Uma Abordagem Lógica à Computação Quântica / Juan Carlos Agudelo Agudelo.
- - Campinas, SP : [s. n.], 2009.

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

1. Funções computáveis. 2. Turing - Máquinas. 3. Circuitos Lógicos. 4. Computação quântica. I. Carnielli, Walter A. (Walter Alexandre), 1952- II. Universidade Estadual de Campinas. Instituto de Filosofia e Ciências Humanas. III. Título.

Título em inglês: Paraconsistent computation: a logic approach to quantum computation

Palavras chaves em inglês (keywords): Computability theory
Turing machines
Logic circuits
Quantum computation

Área de Concentração: Lógica

Titulação: Doutor em Filosofia

Banca examinadora: Walter A. Carnielli, Marcelo E. Coniglio, Marcos C. de Oliveira, Marcelo Finger, Osvaldo Pessoa Jr.

Data da defesa: 19-11-2009

Programa de Pós-Graduação: Filosofia

Juan Carlos Agudelo Agudelo

Computação Paraconsistente: Uma Abordagem Lógica à Computação Quântica

Tese de Doutorado apresentada ao Departamento de Filosofia do Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas sob a orientação do Prof. Dr. Walter Alexandre Carnielli.

Este exemplar corresponde à redação final da tese defendida e aprovada pela Comissão Julgadora em 19/11/2009.

BANCA

Prof. Dr. Walter Alexandre Carnielli (Orientador)

Prof. Dr. Marcelo Esteban Coniglio

Prof. Dr. Marcos Cesar de Oliveira

Prof. Dr. Marcelo Finger

Prof. Dr. Osvaldo Frota Pessoa Junior

Prof. Dr. Silvio Seno Chibeni (Suplente)

Prof. Dr. Carlile Campos Lavor (Suplente)

Prof. Dr. Francisco Antonio de Moraes Accioli Doria (Suplente)

Novembro de 2009

A minha mãe e à memória de meu pai

Agradecimentos

Quero aqui expressar meus agradecimentos às pessoas que colaboraram, de diversas maneiras e de forma influente, na realização desta tese: A meu orientador, o Professor Walter Carnielli, por seu constante apoio e confiança, por sua amizade, por me compartilhar seus conhecimentos e idéias inovadoras, pelas horas de trabalho em conjunto e por me encorajar a seguir em frente na procura do conhecimento. Ao Professor Marcelo Coniglio, pelo apoio e pelo incentivo, por sua amizade e por também compartilhar comigo seu conhecimento. À Professora Itala D'Ottaviano, por seus ensinamentos e seu apoio constante. Aos meus colegas, em particular a Anderson de Araújo, Juliana Bueno-Soler, Evandro Gomes, Rodrigo de Alvarenga Freire, Leandro Suguitani, Rafael Testa, Samir Gorsky e Ricardo Grande, pela amizade e pelas conversas acadêmicas. A José Guimarães e Anderson de Araújo, em particular, por nosso estudo mais profundo sobre computabilidade. Aos membros da banca por terem levado a sério sua função com a maior competência e seriedade, e por suas sugestões e comentários muito pertinentes. A minha mãe, minhas irmãs, meus familiares e meus amigos por seu apoio incondicional e por tantos momentos belos que têm dado sentido à vida. A Luanda por sua companhia e carinho infinito, e também pelas correções ao meu português!

Resumo

Neste trabalho levantamos, e investigamos do ponto de vista conceitual, evidências de que a complexidade algorítmica pode ser vista como relativa à lógica. Propomos, para tanto, novos modelos de computação fundados sobre lógicas não-clássicas, estudando suas características quanto à expressabilidade computacional e eficiência. A partir desta visão, sugerimos um novo caminho para estudar a eficiência dos modelos de computação quântica, enfatizando a análise de uma lógica subjacente a tais modelos. O conteúdo da tese está estruturado da seguinte maneira: no primeiro capítulo apresentamos uma análise conceitual da noção de 'computação', indicando como este conceito tem mudado desde os trabalhos fundacionais da década de 1930, e discutindo se o conceito deve ser considerado como puramente físico, puramente lógico-matemático ou uma combinação de ambos. O Capítulo 2 introduz duas versões de 'máquinas de Turing paraconsistentes', usando sistemas lógicos diferentes e obtendo modelos com diferentes poderes computacionais (quanto à eficiência); tal resultado constitui uma primeira evidência a favor da relatividade lógica da computação que queremos defender. Outra evidência na mesma direção é apresentada no Capitulo 3, através da generalização dos circuitos booleanos para lógicas não-clássicas, em particular para a lógica paraconsistente mbC e para a lógica modal $S5$, e da análise do poder computacional de tais generalizações. O Capítulo 4 consiste numa introdução à computação quântica, para logo (no Capítulo 5) estabelecer algumas relações entre modelos de computação quântica e modelos de computação paraconsistente, de maneira a propor uma interpretação lógica dos modelos quânticos. No capítulo final (Capítulo 6) descrevemos várias relações entre mecânica quântica e lógica para-

consistente, relações estas que sugerem potencialidades com alto grau de relevância a respeito da abordagem paraconsistente dos fenômenos computacionais quânticos e que incitam a continuar explorando esta alternativa.

Abstract

This work provides evidences to view computational complexity as logic-relative, by introducing new models of computation through non-classical logics and by studying their features with respect to computational expressivity and efficiency. From this point of view, we suggest a new way to study the efficiency of quantum computational models consisting in the analysis of an underlying logic. The contents of the thesis is structured in the following way: the first chapter presents a conceptual analysis of the notion of 'computation', showing how this concept evolved since the decade of 1930 and discussing whether it can be considered a pure physical or a pure logic-mathematical concept, or a combination of both paradigms. Chapter 2 introduces two versions of 'paraconsistent Turing machines', by considering different logic systems and obtaining models with different computational capabilities (with respect to efficiency); such a result constitute a first evidence in favor of the logical relativity of computation that we are defending here. Another evidence in the same direction is presented in Chapter 3 through a generalization of boolean circuits to non-classical logics, particularly for the paraconsistent logic mbC and for the modal logic $S5$, and by analyzing the computational power of such generalizations. Chapter 4 consists in an introduction to quantum computation. This is used in Chapter 5 to establish some relationships between quantum and paraconsistent models of computation, in order to propose a logic interpretation of quantum models. The final chapter (Chapter 6) describes several connections between quantum mechanics and paraconsistent logic; such relationship suggests highly relevant potentialities in favor of the paraconsistent approach to quantum computation phenomena encouraging to

continue exploring this alternative.

Sumário

1 Um marco conceitual **1**

1.1 A computação como um processo realizado por humanos 5

1.1.1 Idéias de Gödel e Church 5

1.1.2 As máquinas de Turing e a Formulação de Post 8

1.2 A computação como um processo realizado por máquinas 11

1.2.1 A generalização de Gandy 11

1.2.2 Deutsch e sua proposta reducionista 13

1.3 Computabilidade: a vertente física versus a concepção lógico-matemática 17

1.3.1 Sobre a classe de funções computáveis 18

1.3.2 Sobre a eficiência dos processos de computação 20

1.4 Relativização da computação através de lógicas não-clássicas 23

2 Máquinas de Turing paraconsistentes **29**

2.1 Definições preliminares 32

2.2 Axiomatizando computações de máquinas de Turing 37

2.3 Máquinas de Turing paraconsistentes 44

2.3.1 Definição do modelo de MTPs 45

2.3.2 Poder computacional das MTPs 48

2.4 Máquinas de Turing paraconsistentes não-separáveis 52

2.4.1 Definição do modelo de MTPNSs 53

2.4.2 Poder computacional das MTPNSs 58

3 Circuitos paraconsistentes **61**

3.1 Definições preliminares 62

3.2 O cálculo de anéis de polinômios 66

3.3 Circuitos baseados em lógicas não-clássicas 75

3.3.1 Circuitos paraconsistentes mbC 76

3.3.2 Circuitos modais $S5$ 80

4 Computação quântica **85**

4.1 Uma breve introdução à mecânica quântica 87

4.2 Modelos de computação quântica 96

4.2.1 Máquinas de Turing quânticas 96

4.2.2 Circuitos quânticos 101

4.3 Algoritmos quânticos 105

4.3.1 Algoritmo para o problema de Deutsch 105

4.3.2 Algoritmo para o problema de Deutsch-Jozsa 107

4.3.3 Algoritmo de Simon 108

4.3.4 Algoritmo de Shor 109

4.3.5 Algoritmo de Grover 110

4.4 Sobre o poder computacional dos modelos quânticos 112

5 Computação paraconsistente versus computação quântica **113**

5.1 Simulação de MTQs através de MTPs 114

5.2 Relações entre CQs e L-circuitos 118

5.3 Expressando computações quânticas através de noções modais paraconsistentes 120

6 Paraconsistência, fenômenos quânticos e computação: algumas reflexões **127**

A Elementos de lógica paraconsistente e modal **133**

B Elementos de álgebra linear **141**

Referências Bibliográficas **149**

Capítulo 1

Um marco conceitual

É um fato conhecido, como nota por exemplo Richard Soare em [Soa96, págs. 287–288], que os matemáticos vêm realizando cálculos e algoritmos pelo menos desde a época dos Babilônios (por volta de 2000 a.C.). Talvez o mais conhecido algoritmo antigo (datado de 300 a.C.) seja o algoritmo de Euclides que encontra o máximo divisor comum entre dois números. Além de especificar algoritmos, no final do segundo século antes de Cristo os matemáticos e astrônomos gregos já tinham construído um sofisticado mecanismo, chamado *Antikythera*, que permitia computar informações referentes aos movimentos dos astros. Com esse mecanismo era possível calcular as fases da lua e determinar calendários solares, possibilitando predizer eclipses lunares e solares com base em progressões aritméticas definidas pelos Babilônios (cf. [FBM$^+$06]).

Por outro lado, o teólogo e filósofo catalão Ramon Llull (1232-1316) propôs, por volta de 1275, sua *Ars generalis ultima* ou *Ars Magna* (publicada em 1305) que consistia num método de combinar atributos (selecionados de várias listas) de tal modo que as combinações geravam idéias, e projetou diversos mecanismos baseados em rotações de discos para obter tais combinações. O método de Llull pode ser considerado uma das primeiras tentativas de produzir conhecimento por meios mecânicos, uma explicação geral desse método e sua gênese pode ser encontrado em [Hil71]. Llull tem sido, inclusive, considerado um precursor da moderna lógica simbólica e mesmo da programação de computadores, conforme Ernest Moody sugere em [Moo67].

Mais recentemente, em 1642, Blaise Pascal construiu uma máquina com capacidade de realizar somas e subtrações, a conhecida *Pascalina*, considerada por muitos como a primeira calculadora mecânica do mundo. Em 1671, Gottfried W. Leibniz foi um pouco mais longe e idealizou uma máquina com capacidade de realizar adições, subtrações, multiplicações, divisões e raízes quadradas. Por volta de 1834, Charles Babbage desenhou (mas não conseguiu construir totalmente) uma máquina mais complexa do que as anteriores, a qual é conhecida como *Analytical Engine* (cf. [Cop08]).

Este esboço histórico é suficiente para mostrar quão antiga é a noção de algoritmo, e quão velha é a tentativa humana de construir máquinas para realizar, de maneira automática, computações cada vez mais complexas. Só na década de 1930, porém, é que surgiram as definições rigorosas (matemáticas) da noção de procedimento algorítmico (também chamado *procedimento efetivo* ou *procedimento computável*). Tais definições emergiram principalmente na procura de uma resposta a um problema colocado pelo matemático David Hilbert, no final da década de 1920, conhecido como o *Entscheidungsproblem* ou *Problema da Decisão*, que pode ser enunciado da seguinte maneira:

> **Problema da Decisão**: existe um método (ou procedimento) 'efetivo' para se determinar a validade (ou, equivalentemente, a teoremicidade) de qualquer formula da lógica de predicados de primeira ordem?

O Problema da Decisão foi resolvido (com resposta negativa) inicialmente por Alonzo Church, em [Chu36a], e depois por Alan Turing, em [Tur36], com uma demonstração alternativa.

Na especificação do problema, Hilbert não indicava formalmente em que consistiria a noção de 'procedimento efetivo', mas deixava claro que as operações envolvidas nesses procedimentos deveriam ser de caráter finitário. A busca de uma resposta ao Problema da Decisão fez então necessário inicialmente propor uma definição matemática que de alguma maneira capturasse a idéia intuitiva de procedimento efetivo. Entre os principais formalismos que surgiram na década de 1930 nesta direção, encontram-se as *funções recursivas* (propostas por Jacques Herbrand e Kurt Gödel,

com importantes conseqüências obtidas por Stephen Kleene), as *funções λ-definíveis* (propostas por Alonso Church e Stephen Kleene), o modelo abstrato de máquinas proposto por Alan Turing em [Tur36] e a formalização dos processos de manipulação finita de símbolos exposta por Emil Post em [Pos36] (que é bastante similar à proposta de Turing, embora tenham sido formulações independentes). Estes (e outros) formalismos foram demonstrados serem extensionalmente equivalentes, i.e., a classe de funções definidas pelas diferentes teorias é exatamente a mesma (embora as definições sejam intensionalmente diferentes, como argumentado por Soare em [Soa96]).

A tese de que o modelo abstrato de máquina proposto por Turing (agora chamado de *máquina de Turing*), ou qualquer outro formalismo extensionalmente equivalente, captura de maneira precisa o conceito intuitivo de procedimento efetivo é usualmente conhecida como *Tese de Church-Turing*, e pode ser enunciada do seguinte modo:

> **Tese de Church-Turing**: um procedimento é efetivamente computável se, e somente se, pode ser computado por uma máquina de Turing.

Esta é uma tese que tem a peculiar propriedade de ser impossível de ser demonstrada matematicamente, pois compara um conceito intuitivo com uma definição matemática. A Tese de Church-Turing é usualmente justificada pelas evidências de que diferentes definições de computabilidade têm resultado ser extensionalmente equivalentes; contudo, existe uma razão que talvez tenha maior peso: o cuidado e o rigor (embora não constitua um argumento matemático) da análise de Turing. De fato, foi só a partir da definição de Turing que Gödel convenceu-se de que se tinha chegado a uma definição satisfatória do conceito de procedimento efetivo (cf. [Sie06]). Por outro lado, Robin Gandy, em [Gan88, pág. 76], chega a afirmar que a análise de Turing não constitui apenas um argumento a favor da Tese de Church-Turing, mas que chega a ser ela mesma, de fato, um teorema, que enuncia como:

> "**Teorema de Turing**: toda função efetivamente calculável por um ser humano abstrato [i.e. sem limitações de tempo e espaço] seguindo uma rotina fixa é efetivamente calculável por uma máquina de Turing - ou equivalentemente, efetivamente calculável no sentido definido por Church

- e reciprocamente." (cf. [Gan88, pág. 77], tradução nossa).

Wilfried Sieg, numa série de artigos entre 1994 e 2008 ([Sie94, Sie97, Sie02, Sie06, Sie08]), apresenta de maneira bastante detalhada as principais idéias de Gödel, Church e Turing em relação à noção de computabilidade. Nesses artigos, e também em [MS95] e [Soa96], fica claro que os formalismos propostos por Gödel, Church, Turing e Post são baseados na análise das operações realizadas por humanos quando envolvidos em processos de computação (no caso de Gödel e Church, dentro de um sistema formal, como será explicado adiante). Em [Sie02] Sieg também mostra que, a partir das idéias de Robin Gandy em [Gan80], a noção de procedimento efetivo começou a ser estudada desde um marco conceitual mais amplo, não apenas considerando as operações realizadas por humanos que computam, mas também qualquer operação possível de ser levada a cabo por uma máquina (sob certas restrições de finitude e localidade). Desta maneira a noção de procedimento efetivo passa do estágio de estar delimitada meramente pelas restrições de nossos sentidos ao estágio de estar delimitada pelas leis da natureza.

Em [Deu85] David Deutsch vai ainda mais longe, reinterpretando a Tese de Church-Turing como sendo um 'principio físico'. Além disso, Deutsch argumenta que a noção de computabilidade está delimitada por nossas teorias físicas e que, portanto, a ciência da computação deve ser considerada um ramo da física.

O presente capítulo apresenta os principais pontos conceituais das idéias de Gödel, Church, Turing e Post (Seção 1.1) e de Gandy e Deutsch (Seção 1.2) a respeito da noção de procedimento computável, visando mostrar como o estudo deste conceito tem mudado a partir da década de 1930. Logo depois, levando em conta as idéias expostas (e argumentos de outros cientistas), apresenta-se uma discussão a respeito de qual é (ou deve ser) a natureza da noção de computação (Seção 1.3). A partir desta discussão na Seção 1.4 descrevemos os objetivos que pretendemos atingir com esta tese, os quais consistem basicamente em retomar as idéias de Gödel e Church, de conceber processos de computação como sendo realizados dentro de um sistema lógico dedutivo (em conjunto com algumas idéias de Turing), mas com uma mudança conceitual radical: considerando lógicas não-clássicas (em particular, lógicas paraconsistentes) como lógicas subjacentes. Tal mudança conceitual nos permite definir

modelos de computação paraconsistentes, os quais serão descritos nos próximos capítulos, que servem para mostrar a noção de complexidade algorítmica como sendo relativa à lógica e para explicar algumas das características da computação quântica a partir de uma nova perspectiva.

1.1 A computação como um processo realizado por humanos

Como afirmado anteriormente, os formalismos propostos por Gödel, Church, Turing e Post para capturar a noção de procedimento efetivo são baseados na análise das operações realizadas por agentes humanos quando envolvidos em processos de computação. Porém, Gödel e Church concebiam tais processos como realizados dentro de um sistema formal dedutivo, enquanto Turing e Post os imaginavam como manipulações simbólicas não necessariamente encaixadas dentro de um sistema dedutivo. Essas idéias são apresentadas a seguir.

1.1.1 Idéias de Gödel e Church a respeito da noção de procedimento efetivo

Como já foi mencionado, tanto Gödel quanto Church (em conjunto com outros matemáticos) propuseram definições formais alternativas para o conceito intuitivo de procedimento efetivo: Gödel (e Herbrand) através das funções recursivas[1] e Chuch (junto com Kleene) através das funções λ-definíveis[2]. As funções recursivas são definidas, de maneira construtiva, partindo de um conjunto pequeno de funções base e permitindo algumas operações elementares para gerar novas funções.[3] Por outro lado, a teoria de funções λ-definíveis (ou *cálculo lambda*) provê uma linguagem para

[1] Embora o próprio Gödel, na época em que a teoria foi proposta, não estivesse convencido de que sua definição fosse o suficientemente abrangente para incluir todas as possíveis recursões, cf. [Dav82].

[2] Contrariamente a Gödel, Church em [Chu36b, pág. 346] afirma explicitamente que o propósito desse artigo é propor uma definição de calculabilidade efetiva que corresponda satisfatoriamente ao conceito intuitivo.

[3] Para uma definição formal veja, por exemplo, [CE05].

denotar funções e estabelece um conjunto de possíveis conversões para as fórmulas da linguagem. Dentro da linguagem são escolhidas algumas fórmulas para denotar números naturais, as quais são chamadas de 'numerais'. A idéia de computação no cálculo lambda consiste então em transformar fórmulas, representando funções e numerais como parâmetros, em 'formas normais' (através das conversões permitidas). Desta maneira, as conversões definem os passos do algoritmo e a forma normal representa o resultado da computação (cf. [Chu36b]).

A equivalência entre funções recursivas e funções λ-definíveis foi inicialmente demonstrada por Kleene. Segundo Church, "o fato de que essas duas definições amplamente diferentes e igualmente naturais de calculabilidade efetiva resultassem ser equivalentes, acrescenta força às razões para acreditar que elas constituem uma caracterização de essa noção que é consistente com seu entendimento intuitivo usual" (cf. [Chu36b, pág. 346], tradução nossa).

Além dos formalismos de funções recursivas, funções λ-definíveis e sua equivalência, o que mais nos interessa salientar aqui é que tanto Gödel quanto Church pensavam na idéia de computação como sendo realizada dentro de um sistema formal (dedutivo). Para Gödel a essência de um sistema formal consiste em substituir o raciocínio por operações mecânicas de fórmulas (cf. [Göd64]). Por isso, quando define a noção de 'sistema formal matemático', em [Göd34], exige que as fórmulas, os axiomas e as regras de inferência sejam definidas de maneira 'construtiva'. Indicando com isso que:

- para cada regra de inferência deve existir um 'procedimento finito' que permita determinar quando uma fórmula é uma *conseqüência imediata* (devido a essa regra) de um conjunto de fórmulas; e

- devem também existir procedimentos finitos para determinar quais são as fórmulas com sentido (fórmulas bem formadas) e quais são os axiomas do sistema.

Em [Göd64], Gödel indica que 'procedimento finito' pode ser entendido como 'procedimento mecânico' no sentido de Turing (i.e., computável por uma máquina de Turing).

Considerando S como sendo um sistema formal que valida os requisitos anteriormente mencionados, Gödel define uma função $\phi(x)$ (de naturais em naturais) como sendo "*computável em* S se para todo numeral m existe um numeral n tal que $\phi(m) = n$ é demonstrável em S" (cf. [Göd36, pág. 397], tradução nossa). Então, ele define uma seqüência de sistemas lógicos S_i, onde $i = 1, 2, \ldots$ indica a ordem do sistema (sendo S_1 a aritmética clássica). O sistema S_i contém, além dos axiomas lógicos, variáveis e quantificadores para números naturais, para classes de números naturais, para classes de classes de números naturais, e assim sucessivamente até classes de tipo i. Seguindo esta idéia, Gödel indica que toda função recursiva é computável em S_1; além disso, num comentário adicionado a [Göd36] (cf. [Göd86, pág. 399]), ele afirma que se uma função é computável em qualquer sistema S_i então é computável também em S_1 e que, portanto, a noção de computável é, num certo sentido, 'absoluta'.

Uma idéia similar à de Gödel é proposta por Church indicando que uma função F é "*computável dentro* da lógica [i.e. dentro do sistema formal dedutivo em consideração] se existir uma expressão f na lógica tal que $\{f\}(\mu) = \nu$ é um teorema se, e somente se, $F(m) = n$ é verdadeiro, sendo μ e ν expressões que representam os inteiros positivos m e n" (cf. [Chu36b, pág. 357], tradução nossa). Na definição do que é um sistema lógico, Church indica que as regras de inferência devem ser recursivas (no sentido das funções recursivas de Gödel) e que os conjuntos de axiomas e regras de inferência devem ser recursivamente enumeráveis. Esta idéia é equivalente à de Gödel, mas é exposta de maneira um pouco mais cuidadosa e sem fazer referência a uma seqüência de sistemas de diferentes ordens.

Nas propostas de Gödel e Church parece haver um argumento circular, pois ambos justificam a noção de computabilidade dentro de um sistema formal exigindo condições de recursividade (que corresponde a outra definição do conceito de computabilidade) sobre os axiomas e as regras de inferência do sistema. Contudo, como apontado em [Sie97, pág. 165] (tradução nossa), "nosso entendimento do conceito geral de calculabilidade é explicado [pelas idéias de Gödel e Church] em termos de derivabilidade numa lógica, e as condições [impostas sobre os axiomas e as regras de inferência] esclarecem a idéia de que dentro de tais formalismos lógicos se opera com

uma noção efetiva de conseqüência imediata". Dessa forma, embora as propostas de Gödel e Church possam perder em comparação com uma proposta com maior apelo à intuição como a das máquinas de Turing, descritas a seguir, sua vantagem consiste em mostrar como a computabilidade pode ser compreendida no âmbito de sistemas lógicos dedutivos.

1.1.2 As máquinas de Turing e a Formulação de Post

Em [Tur36], Turing propõe um modelo abstrato de máquinas (hoje chamadas *máquinas de Turing*). A definição do modelo é baseada na análise das operações realizadas por agentes humanos quando envolvidos em processos de computação. Turing inicia sua análise lembrando que computações são normalmente realizadas através da escrita de símbolos num papel. Imagina o papel dividido em células e observa que o caráter bidimensional do papel não é essencial no processo de computação, e portanto limita a análise das operações àquelas que podem ser realizadas num papel unidimensional (i.e., numa fita dividida em células).

Considerando que as operações no processo de computação são realizadas por agentes humanos, Turing indica que estas são determinadas pelos símbolos observados e pelo 'estado da mente' do agente que está realizando a computação. Assim, levando em conta as limitações sensoriais dos agentes humanos, coloca condições de *delimitação* e *localidade*:

- Delimitação: os símbolos de leitura/escrita e os estados da mente devem ser 'imediatamente reconhecíveis'; portanto, os conjuntos de símbolos e de estados da mente devem ser finitos.[4] Além disso, somente um número fixo de células da fita pode ser observado em cada instante de tempo.

- Localidade: somente as células da fita que estão sendo observadas podem ser modificadas.

Por último, Turing divide as operações anteriormente descritas determinando quais seriam as operações mais simples possíveis, i.e., as operações 'atômicas' através

[4] Se fossem infinitos haveriam símbolos ou estados da mente arbitrariamente similares (cf. [Tur36, pág. 249]).

das quais as operações mais complexas podem ser realizadas. Essas operações são:

- Mudança de um único símbolo dentro das células observadas, junto com uma possível mudança do estado da mente.

- Mudança das células observadas em uma única posição, junto com uma possível mudança do estado da mente.

É através destas operações atômicas que Turing define seu modelo abstrato de máquina.

Os números computáveis são então definidos por Turing como aqueles cuja expressão decimal pode ser calculada (escrita) através de suas máquinas, observando que estas máquinas podem, em princípio, ser usadas para realizar qualquer outro tipo de cálculo simbólico. Além disso, propõe seu modelo de máquina como abrangendo todas as possíveis computações, justificando esta afirmação em três argumentos: um apelo direto à intuição (o argumento apresentado acima), a prova de equivalência com outras definições (especificamente com o conceito de definibilidade no cálculo funcional de Hilbert e com as funções λ-definíveis) e ainda exibindo uma grande classe de números computáveis por suas máquinas.

Os argumentos de Turing são bastante convincentes; em particular o primeiro, baseado na intuição do que é um procedimento efetivo (analisando o que uma pessoa faz quando está envolvida num processo de computação). Esse argumento elimina a necessidade de lançar mão de outra noção de computabilidade, diferentemente dos argumentos de Gödel e Church sobre computações em sistemas dedutivos.

Vale a pena notar que no argumento de Turing está implícita a pressuposição de que todo procedimento efetivo pode ser realizado por um agente humano 'idealizado' (sem limites de papel nem de tempo) que computa. Soare chama a esta pressuposição de 'Tese de Turing', a qual enuncia assim:

> **Tese de Turing**: "se uma função é informalmente computável (i.e. definível por um procedimento mecânico finito ou algoritmo) então é *computorável* (i.e. computada por um computador humano idealizado no

sentido de Turing)." (cf. [Soa96, pág. 293], tradução nossa).[5]

Dessa maneira, levando em conta o 'Teorema de Turing' enunciado anteriormente e a validade dos recíprocos (i.e. das implicações nos sentidos contrários) desse teorema e da 'Tese de Turing', obtém-se o que chamamos anteriormente de Tese de Church-Turing.

Além de propor seu modelo de máquinas, Turing definiu uma máquina específica que chamou de *máquina universal*. Essa máquina recebe como entrada a definição codificada de qualquer uma outra máquina, mais um dado de entrada para ela, simulando o comportamento da máquina codificada de tal maneira a produzir o mesmo resultado que ela produziria. Usando esta definição, e outros elementos, Turing mostra que existem problemas que não podem ser computados por suas máquinas. Em particular, não existe uma máquina tal que, recebendo como entrada a codificação de uma máquina de Turing qualquer e um dado de entrada para ela, determine se essa máquina com esse dado de entrada pára ou não (este problema é conhecido como *Problema da Parada*).

Através da axiomatização de suas máquinas em lógica de predicados de primeira ordem Turing mostra uma importante equivalência: esta lógica é decidível (i.e., existe um algoritmo para determinar se uma fórmula é teorema ou não) se e somente se o Problema da Parada for computável. Portanto, pela não-computabilidade do Problema da Parada, Turing conclui o resultado notável de que não pode existir procedimento de decisão para a lógica de predicados de primeira ordem, i.e., que o *Entscheidungsproblem* tem uma resposta negativa.

Em 1936, o mesmo ano em que Turing introduziu seu modelo de máquinas abstratas, Post publicou o artigo [Pos36]. Nesse artigo, o qual foi escrito aparentemente sem conhecer as idéias de Turing, Post propõe um modelo de computação surpreendentemente similar às máquinas de Turing. No modelo de Post o 'solucionador de problemas' ou 'trabalhador' realiza manipulações simbólicas sobre uma fita unidimensional, potencialmente infinita nas duas direções e dividida em células, de maneira análoga às máquina de Turing. As manipulações primitivas contempladas

[5] A palavra utilizada por Soare para se referir a processos computados por humanos é *computorable*, a qual estamos traduzindo como *computorável*.

por Post para a manipulação dos símbolos são praticamente as mesmas estipuladas por Turing. Contudo, diferentemente de Turing, Post somente sugere (mas não demonstra) que sua formulação é equivalente às formulações de Gödel e Church, não propõe nenhuma definição equivalente ao modelo de máquina universal, somente faz uma breve menção da aplicabilidade de sua formulação a sistemas de lógica simbólica e propõe a redução de outras formulações à sua como uma 'hipótese de trabalho' (e não como uma tese ou uma definição).

1.2 A computação como um processo realizado por máquinas

Esta seção pretende mostrar como a análise do conceito de computabilidade mudou seu foco, passando de imitar agentes humanos a imaginar o que máquinas enquanto sistemas físicos podem executar: em outras palavras, a partir da análise de Gandy ocorre uma mudança de paradigma de maneira a levar em conta operações que podem ser realizadas por máquinas (enquanto objetos físicos sob certas restrições), sem que se tenha necessariamente que imitar o que os humanos fazem quando estão computando. A seguir, apresentaremos os argumentos de Gandy e Deutsch nesta direção.

1.2.1 A generalização de Gandy a respeito das máquinas de Turing

Em [Gan80] Gandy observa (já no resumo) que a "análise de Turing sobre o conceito de computação não se aplica diretamente a dispositivos mecânicos em geral" (tradução nossa) e depois afirma que o que Turing fez foi provar o que ele chama de 'Teorema T' (ou 'Teorema de Turing' em [Gan88], o qual já foi enunciado na página 3). Nesse 'teorema' é estabelecida uma relação entre o que pode ser computado por um ser humano abstrato (sem limitações de tempo e espaço) e o que pode ser computado por uma máquina de Turing. Para Gandy, "embora alguns argumentos de Turing possam ser aplicados indiferentemente a pessoas ou máquinas, existem passos cruciais na analise de Turing onde ele apela ao fato de que o cálculo é realizado por um ser humano. Um de tais apelos é usado para justificar o pressuposto de que a

computação procede como uma seqüência de passos elementares. Um humano pode somente escrever um símbolo em cada instante de tempo. Mas, se nos abstraímos de limitações práticas, é possível conceber uma máquina que imprime um número arbitrário de símbolos simultaneamente" (cf. [Gan80, págs. 124-5], tradução nossa).

O propósito central de Gandy em [Gan80] é analisar o que pode ser computado por máquinas em geral e prover argumentos para o que ele chama de 'Tese M', a qual enuncia assim:

> "**Tese M**: o que pode ser computado por uma máquina é computável [por uma máquina de Turing]." (cf. [Gan80, pág. 124], tradução nossa).

Para isso, propõe uma estrutura matemática como um formalismo abstrato para representar dispositivos mecânicos discretos que evoluem deterministicamente, excluindo explicitamente qualquer tipo de dispositivo analógico (pela possibilidade de que estes permitiriam realizar operações não-intuitivamente computáveis, de alguma maneira infinitas), mas permitindo a execução de operações em paralelo. Nesse formalismo os estados dos dispositivos mecânicos são representados através de uma classe particular de conjuntos chamados *conjuntos estruturais*[6], que permitem representar os componentes da máquina em hierarquias. As transições de estados são então representadas através de funções sobre esses conjuntos, sob certas restrições (análogas às restrições de delimitação e localidade impostas por Turing).

Mesmo abrindo a análise da noção de computabilidade a um escopo mais amplo, Gandy finalmente mostra que tudo o que pode ser computado por um dispositivo mecânico, descrito pelas estruturas matemáticas por ele consideradas, pode ser computado por uma máquina de Turing. Mostra, desta maneira, que a Tese de Church-Turing pode ser formulada de maneira equivalente neste escopo mais geral, completamente desvinculada de qualquer referência a agentes humanos. Este fato pode ser visto como mais uma evidência favorável à Tese de Church-Turing.

É importante salientar que, embora os princípios propostos por Gandy estejam parcialmente justificados em considerações físicas, sua formulação é totalmente abs-

[6]Não é nosso objetivo apresentar aqui os detalhes técnicos, mas proceder a uma análise conceitual da proposta de Gandy. Detalhes técnicos podem ser consultados diretamente em [Gan80], ou em [SB99], onde são apresentados de maneira simplificada.

trata e independente de qualquer teoria física. De fato, Gandy faz uma diferenciação (embora um pouco obscura) entre 'dispositivos mecânicos' e 'dispositivos físicos'. Aparentemente, os dispositivos físicos são aqueles dispositivos que podem ser definidos através de uma teoria física qualquer e os 'dispositivos mecânicos' são aqueles cuja evolução pode ser descrita de maneira discreta, determinística e com as condições de delimitação e localidade por ele impostas.

Como apontado por Gandy, "as únicas pressuposições físicas a respeito dos dispositivos mecânicos (...) são a existência de um limite inferior nas dimensões lineares de cada parte atômica [em congruência com os postulados da mecânica quântica] e a existência de um limite superior (a velocidade da luz) na velocidade de propagação das mudanças [em congruência com a teoria da relatividade]." (cf. [Gan80, pág. 126], tradução nossa).

1.2.2 Deutsch e sua proposta reducionista da computação à física

Como mencionado anteriormente, Deutsch em [Deu85] vai mais longe do que Gandy em sua interpretação de processos de computação como sendo realizados por máquinas. Para ele, máquinas são objetos físicos reais, portanto o que pode ser realizado através de máquinas está determinado pelas leis da física, e não por teorias matemáticas. Algumas colocações de Deutsch são bastante discutíveis. Nesta seção, vamos apresentar as idéias por ele propostas, deixando a discussão de algumas delas para a próxima seção.

Deutsch argumenta que a Tese de Church-Turing tem implícita uma afirmação física, e propõe explicitá-la através de um princípio físico (com igual estatuto epistemológico de outros princípios físicos), o qual enuncia da seguinte maneira:[7]

> **Versão física da Tese de Church-Turing**: "todo sistema físico finitamente realizável pode ser simulado perfeitamente por uma máquina de

[7] É interessante notar aqui que Post, em [Pos36], propõe que a continua verificação da Tese de Church-Turing (a qual ele aceitava só como uma hipótese de trabalho) pode levar a que se converta numa *lei natural*, ao invés de uma definição ou 'axioma'. Como apontado em [Soa96, pág. 301], isso sugere que Post talvez pensasse na Tese de Church-Turing como uma lei natural como o são as leis da física newtoniana. Contudo, Post não explorou a fundo esta alternativa.

> computação universal operando por meios finitos." (cf. [Deu85, pág. 99], tradução nossa).

Na sua versão da Tese de Church-Turing, Deutsch reinterpreta a noção intuitiva de procedimento efetivo como procedimentos que podem, em princípio, ser computados por um sistema físico 'finitamente realizável'; indicando por sistema físico finitamente realizável todo sistema composto por objetos físicos que podem ser preparados e medidos através de experimentos. Por 'simulação perfeita' ele entende que o comportamento da máquina deve ser 'funcionalmente indistinguível' do comportamento do sistema (i.e., o comportamento de entrada/saída da máquina deve se corresponder à preparação e às medições do sistema, sob uma certa interpretação de seus elementos). A noção de 'máquina de computação universal operando por meios finitos' substitui o uso de um modelo particular (como, por exemplo, o modelo de máquina de Turing), pretendendo atingir com isso uma maior generalidade.

As condições de Deutsch para considerar uma máquina como 'operando por meio finitos' são similares às já presentes na análise de Turing, mas com referência a elementos do sistema físico: "(i) somente um subsistema finito (embora não sempre o mesmo) pode mudar em cada passo de computação, (ii) as mudanças dependem unicamente do estado de um subsistema finito e (iii) as regras que especificam as mudanças do sistema podem ser dadas finitamente no sentido matemático (por exemplo como um inteiro)." (cf. [Deu85, pág. 100], tradução nossa).

Deutsch afirma que sua versão da Tese de Church-Turing é melhor definida, e menos ambígua, do que a versão original. Ele justifica esta afirmação dizendo que sua versão só faz referência a conceitos objetivos, tais como 'medição', 'preparação' e 'sistema físico' (presentes em 'teorias da medição'), diferentemente de conceitos ambíguos como 'procedimento efetivo', o qual não se encaixa dentro da estrutura das teorias físicas existentes.

Para Deutsch, as máquinas de Turing não satisfazem a versão física da Tese de Church-Turing na física clássica. Seu argumento para justificar essa afirmação é o seguinte:

1. A dinâmica clássica é contínua, portanto, o número de possíveis estados de um

sistema clássico formam necessariamente um contínuo.

2. As máquinas de Turing permitem unicamente uma quantidade enumerável de configurações.

3. Portanto, as máquinas de Turing não podem realizar uma simulação perfeita de qualquer sistema físico clássico (simulações discretas de sistemas contínuos podem ser consideradas só como aproximações, não como simulações perfeitas).

Por outro lado, Deutsch define um modelo de 'computador quântico universal', e afirma que esse modelo satisfaz sua versão da Tese de Church-Turing na mecânica quântica.

Embora o modelo de computador quântico proposto por Deutsch permita um número não enumerável de configurações (devido ao uso de números complexos em sua descrição), esse modelo só consegue computar a mesma classe de funções computadas por máquinas de Turing. Sua vantagem parece estar na eficiência dos algoritmos.[8] Mesmo assim, Deutsch afirma que "não existem razões *a priori* de porque as leis da física devem respeitar as limitações dos processos matemáticos que chamamos de 'algoritmos' (i.e. as funções computáveis por máquinas de Turing)" (cf. [Deu85, pág. 101], tradução nossa). Dessa forma, Deutsch reconhece que a Tese de Church-Turing poderia em princípio ser derrogada através de leis da física.

Outras afirmações discutíveis de Deutsch a respeito da natureza da noção de computabilidade, que serão avaliadas na próxima seção, são as seguintes:

- "Toda máquina de computação $\mathcal{M}$ computa uma única função. Contudo, parece não haver uma diferença fundamental entre mudar o estado inicial no qual $\mathcal{M}$ é preparada e mudar sistematicamente a constituição de $\mathcal{M}$, para que ela se converta em uma máquina diferente $\mathcal{M}'$, computando uma função diferente.

[8] Algumas características da mecânica quântica permitem certo tipo de processamento em paralelo, o que pode ser usado na definição de algoritmos eficientes. Contudo, não há ainda um resultado que mostre, de maneira definitiva, a superioridade desse modelo em relação ao modelo clássico de máquina de Turing em termos de eficiência (uma introdução à computação quântica é apresentada no Capítulo 4.)

Para isso é útil considerar máquinas com duas entradas, a primeira entrada correspondendo a um 'programa' para determinar qual função computar. Desta maneira, a cada máquina $\mathcal{M}$ corresponde um conjunto $C(\mathcal{M})$ de 'funções $\mathcal{M}$-computáveis'. Uma função f é $\mathcal{M}$-computável se $\mathcal{M}$ pode computar f quando é preparada com algum programa.

O conjunto $C(\mathcal{M})$ pode ser acrescentado através de aumentar o conjunto de mudanças na constituição de $\mathcal{M}$ que são identificadas como possíveis $\mathcal{M}$-programas. Dadas duas máquinas $\mathcal{M}$ e $\mathcal{M}'$ é possível construir uma máquina composta cujo conjunto de funções computáveis contenha a união de $C(\mathcal{M})$ e $C(\mathcal{M}')$.

Não existe uma razão puramente lógica que determine porque não podemos continuar *ad infinitum* construindo máquinas cada vez mais poderosas, nem para determinar que existe uma função que está por fora do conjunto de funções computáveis de todas as máquinas fisicamente possíveis. Embora a lógica não proíba a computação física de funções arbitrárias, aparentemente a física o faz." (cf. [Deu85, pág. 98], tradução nossa).

- "Não é obvio *a priori* que todas as funções recursivas sejam computáveis na realidade física. A razão pela qual é possível construir, por exemplo, calculadoras eletrônicas, e pela qual podemos realizar aritmética mental, não pode ser encontrada na matemática ou na lógica. *A razão é que as leis da física permitem a existência de modelos físicos para as operações aritméticas* tais como adição, subtração e multiplicação." (cf. [Deu85, pág. 101], tradução nossa).

- "Ver a hipótese de Church-Turing como um princípio físico não somente faz a ciência da computação ser um ramo da física, também faz que parte da física experimental seja um ramo das ciências da computação." (cf. [Deu85, pág. 115], tradução nossa).

1.3 Computabilidade: a vertente física versus a concepção lógico-matemática

A noção de computabilidade deve ser considerada como sendo puramente física? Ou, pelo contrário, deve ser considerada como sendo puramente lógico-matemática? Ou, de uma maneira mais conciliadora, deve considerar tanto elementos físicos quanto lógico-matemáticos?

Na década de 1930, quando a noção de procedimento efetivo começou a ser definida de maneira rigorosa, as teorias propostas eram abstratas, consistindo em definições puramente matemáticas ou lógicas, sem referência direta a teorias físicas (as descrições dos trabalhos de Gödel, Church, Turing e Post acima mostram isso claramente).[9] Por outro lado a proposta de Deutsch proclama, explicitamente, a noção de computabilidade como sendo puramente física. A oposição radical destas duas concepções é devida, em boa parte, à mudança de foco do que se entende quando se fala em 'computação': no âmbito dos trabalhos fundacionais da teoria da computabilidade a palavra computação se refere à noção intuitiva de procedimento efetivo, e a suas definições formais. Diferentemente, quando Deutsch fala em computação refere-se a um processo que pode ser realizado por uma máquina enquanto sistema físico real, portanto restrito às leis da física e não ao que possa ser o conceito intuitivo de procedimento efetivo.

Uma primeira crítica à proposta de Deutsch repousa sobre a alegação de que existam afirmações físicas implícitas na Tese de Church-Turing: é claro que essa tese é puramente abstrata, pois os modelos de computação definidos por Church e Turing são construtos matemáticos independentes de qualquer consideração física. Porém, a proposta de Deutsch é interessante no sentido de mudar o escopo da noção de computabilidade, abrindo uma discussão a respeito do que pode ser computado através de 'sistemas físicos finitamente realizáveis'. Entendendo a noção de computação no

[9] No trabalho de Turing as restrições sobre as operações que suas máquinas podem realizar são justificadas em restrições sensoriais dos seres humanos, o que pode ser considerado como restrições físicas. Porém estas justificações agem somente como princípios que guiam a intuição na direção da definição matemática do que seja um procedimento efetivo.

sentido amplo de processos que podem ser computados por máquinas fisicamente realizáveis, vale a pena nos perguntar sobre a classe de funções que podem ser computadas através de máquinas, e pelos limites sobre a eficiência dos processos de computação; além disso pode-se discutir se essas características são determinadas por considerações físicas ou lógico-matemáticas. A seguir apresentamos algumas reflexões nestas direções.

1.3.1 Sobre a classe de funções computáveis

Como descrito acima, Gandy já havia analisado a noção de computação desde um escopo mais amplo do que as análises de Gödel, Church, Turing e Post. A demonstração de Gandy de que seu modelo abstrato de máquina consegue computar a mesma classe de funções que as máquinas de Turing mostra a abrangência da Tese de Church-Turing. Por outro lado, a análise de Gandy mostra claramente como a classe de funções computáveis é determinada por questões matemáticas (embora se referindo abstratamente a objetos físicos sob certas restrições), pois são precisamente as restrições colocadas sobre as estruturas matemáticas, consideradas como descrevendo máquinas, que delimitam o que pode ser computado.

Devemos aceitar aqui um argumento de Deutsch, apresentado acima, que levaria a dizer que o modelo de Gandy pode não ser *a priori* construtível na realidade física. Contudo, é indiscutível que a tese de Gandy estabelece, matematicamente, os limites do que pode ser computado por qualquer sistema físico que possa ser descrito através das estruturas por ele contempladas. Isso mostra quão incorreto (e mesmo disparatado) é o argumento de Deutsch no sentido de que não existem razões lógicas (ou matemáticas) que determinem o que pode ser computado por máquinas fisicamente realizáveis.

A noção de sistema físico finitamente realizável proposta por Deutsch é ainda mais geral do que o modelo de máquinas considerado por Gandy, incluindo, em particular, sistemas cujos possíveis estados são representados por uma quantidade não enumerável de configurações. Porém, o modelo de computador quântico proposto por Deutsch computa a mesma classe de funções Turing-computáveis. Esse fato mostra, em lugar de apontar fraquezas das máquinas de Turing, a abrangência da

Tese de Church-Turing: ela pode ser estendida a certos modelos descritos através de números complexos (operando em instantes de tempo discretos, e sob restrições de delimitação e localidade). De fato, Ethan Bernstein e Umesh Vazirani, em [BV97], mostram que o computador quântico proposto por Deutsch pode ser considerado um modelo de computação discreto, em lugar de analógico.

Por outro lado, Itamar Pitowsky, em [Pit90], argumenta a favor da possibilidade das teorias físicas permitirem construir máquinas que computem mais do que uma máquina de Turing, e de maneira mais eficiente, contra-argumentando as idéias de Stephen Wolfram expostas em [Wol85]. Mas não é só Pitowsky que defende esta posição, vários modelos de *hiper-computação* (i.e., modelos teóricos de computação que podem, em princípio, calcular funções não computáveis por máquinas de Turing) têm sido propostos, vários deles usando teorias físicas na sua constituição, ou nas justificativas de suas possibilidades de construção.[10]

As teorias de hiper-computação estão longe de serem aceitas de maneira geral; são várias as críticas feitas a esses modelos. Em particular, Paolo Cotogno, em [Cot03], apresenta argumentos bastante convincentes contra a hiper-computação. Ele mostra como uma boa quantidade de modelos de hiper-computação estão sustentados, direta ou indiretamente, sobre a realização de uma tarefa infinita em tempo finito (o que é conhecido como uma *supertask*) ou na consideração de números reais com precisão infinita na descrição do modelo. Isso leva, de maneira imediata, à conclusão de que os processos contemplados nessas teorias não podem ser considerados como procedimentos efetivos (no sentido de Hilbert), e portanto não invalidam a Tese de Church-Turing, como sugerido por vários dos proponentes dessas teorias.

Cotogno também observa que devemos separar cuidadosamente o nível epistemológico do nível ontológico: a não-computabilidade é uma propriedade da maneira matemática como representamos nossas teorias físicas, mas isso não implica, necessariamente, que essa seja uma característica essencial do sistema físico (possivelmente exista uma maneira mais adequada de descrever esse sistema físico, sem introduzir elementos não computáveis). Portanto a definição de um modelo de hiper-computação dentro de uma teoria física não implica, necessariamente, sua construti-

[10] Para uma introdução a modelos de hiper-computação veja [Sta04].

bilidade física. Outro argumento contra a possibilidade de se construir máquinas de hiper-computação é baseado na impossibilidade de realizar medições com precisão infinita.

Consideramos então razoável concluir que a classe de funções computáveis por uma classe de máquinas, definidas através de certas estruturas matemáticas ou lógicas, é completamente determinada por características formais das estruturas em consideração. Porém, a análise matemática ou lógica não implica *a priori* a construtibilidade física das máquinas. É necessária uma teoria física (ou química, ou biológica) que permita levar o modelo abstrato de máquina a sua construção real. No caso do computador quântico de Deutsch (e de outros modelos de computação quântica) o modelo abstrato de máquina e a teoria física que permite a elaboração desse modelo se juntam, sendo esse modelo de máquina quântica definido precisamente através das estruturas matemáticas e princípios físicos usados para descrever os sistemas microscópicos. Talvez seja essa junção que faz Deutsch desconsiderar questões lógicas e matemáticas na delimitação do que pode ser computável.

1.3.2 Sobre a eficiência dos processos de computação

Considerando definida uma classe de funções como sendo computáveis, e pressupondo também que o modelo de computação que define essa classe pode ser fisicamente realizável, é possível ainda ocorrer que o algoritmo para computar uma função exija tantos recursos que ele não pode ser realizado na prática. A partir da década de 1960 o uso de recursos (principalmente espaço e tempo) dos processos de computação começou a ser estudado de maneira sistemática, surgindo uma nova área de pesquisa que hoje é conhecida como *complexidade algorítmica*.

O estudo da complexidade de um algoritmo avalia de que forma o uso de um determinado recurso é escalonado em relação ao tamanho do dado de entrada. A complexidade algorítmica divide as funções computáveis em *tratáveis* e *intratáveis*. Uma função (ou problema) é considerada tratável se existir um algoritmo que compute essa função com demanda de recursos crescendo de maneira no máximo polinomial em relação ao tamanho do dado de entrada; de outro modo, a função é considerada intratável.

As classes de complexidade algorítmicas são definidas para modelos de computação particulares. Por exemplo, a classe **P** é a classe de funções que podem ser computadas por máquinas de Turing determinísticas[11] em tempo polinomial, i.e. com a execução de uma quantidade de instruções delimitada por um polinômio em relação ao tamanho do dado de entrada. Porém, vários resultados mostraram que uma boa quantidade de modelos de computação são equivalentes quanto a sua eficiência às máquinas de Turing determinísticas, i.e. que conseguem computar qualquer função em um número de passos equivalentes (a menos de um número polinomial de passos) ao que levaria computar a mesma função numa máquina de Turing determinística. Isso levou a muitos pesquisadores da área a acreditar numa versão mais forte da Tese de Church-Turing (cf. [Pap94, pág. 36]), que pode ser enunciada assim:

> **Versão forte da Tese de Church-Turing**: todo modelo 'razoável' de computação calcula, com demanda de tempo equivalente (a menos de um número polinomial de passos), a mesma classe de funções que podem ser calculadas por máquinas de Turing determinísticas.

Esta versão é um tanto ambígua, já que não se explica claramente o que se quer dizer por modelo 'razoável' de computação. Em [Pap94, pág. 45] Christos Papadimitriou considera o modelo de máquina de Turing não-determinístico como irrazoável, por ser 'não realista', sem expor nenhuma razão para esta afirmação. Por outro lado, Peter van Emde Boas em [van90] divide os modelos de computação em duas classes, a primeira classe correspondendo àqueles modelos que podem ser simulados eficientemente (em tempo polinomial) por máquinas de Turing determinísticas, e a segunda classe correspondendo a modelos de computação que validam o que ele chama de 'Tese da computação em paralelo', que caracteriza as máquinas em paralelo como aquelas que permitem computar em tempo polinomial o que pode ser computado por máquinas seqüências (da primeira classe) em espaço polinomial. van Emde Boas

[11] Uma máquinas de Turing é chamada 'determinística' se para cada configuração da máquina existe no máximo uma instrução que pode ser executada; caso contrário, a máquina é chamada 'não-determinística', pelo fato de que em alguns instantes de tempo a máquina precisa realizar uma escolha, de maneira não-determinística, entre as múltiplas instruções que podem ser executadas. Estas noções são formalmente definidas no próximo capítulo.

afirma então que "a primeira classe de máquinas representa a classe de modelos seqüências razoáveis. Contudo, para os modelos de máquinas na segunda classe não é totalmente claro se podem ser considerados como razoáveis. Parece que as maravilhosas acelerações obtidas através de modelos de computação em paralelo da segunda classe de máquinas requer severas violações de leis básicas da natureza" (cf. [van90, pág. 14], tradução nossa).

O principal problema atual em complexidade algorítmica consiste em determinar se a classe de funções computáveis em tempo polinomial numa máquina de Turing não-determinística, denotada por **NP**, é igual à classe **P**. A maioria de pesquisadores da área acreditam que estas duas classes são diferentes, mas ninguém conseguiu demonstrá-lo até agora. Em [Sma98] Steve Smale inclui o problema de determinar se $\mathbf{P} = \mathbf{NP}$ entre uma lista dos principais problemas matemáticos a ser considerados no século XXI. Além disso, o Instituto Clay de matemáticas oferece um premio de US$1.000.000 para quem resolver esse problema (veja `http://www.claymath.org/millennium/index.php`). Por outro lado, na página web `http://www.win.tue.nl/~gwoegi/P-versus-NP.htm` aparece uma lista de enlaces a trabalhos que tentam contribuir à solução do problema $\mathbf{P} =? \mathbf{NP}$, muitos deles com erros já apontados.

Entendendo por modelos de computação razoáveis aqueles que podem ser construtíveis na realidade, os modelos de computação quântica representam o maior desafio para a versão forte da Tese de Church-Turing já que em princípio eles seriam fisicamente construtíveis, e há evidências de que eles poderiam ser mais eficientes do que as máquinas de Turing. Os modelos quânticos, embora definidos através de princípios físicos, podem ser estudados sem fazer referência alguma a propriedades físicas como é mostrado em [For03].

De maneira análoga ao que acontece com a determinação da classe de funções computáveis, parece claro que a eficiência dos algoritmos, dentro de um determinado modelo de computação, está determinada pelas estruturas matemáticas ou lógicas usadas na definição do modelo (embora seja difícil determinar os limites). Porém, a existência de um modelo matemático não implica *a priori* a construtibilidade física desse tipo de máquinas. É necessária uma teoria física (ou química, ou biológica) que permita representar esse modelo abstrato para levá-lo a sua construção real.

As idéias de Deutsch, expostas na Seção 1.2.2, buscam resumir a computação a um ramo da física, desconsiderando o papel da lógica e da matemática no estabelecimento dos limites do que pode ser computado e da eficiência dos processos de computação. Em trabalhos posteriores de outros pesquisadores, ao parecer bastante influenciados pelas idéias de Deutsch, o conceito de computação é tratado como sendo essencialmente físico. Em [EJ96], por exemplo, os autores começam o artigo afirmando que "Os conceitos de informação e computação podem ser adequadamente formulados somente no contexto de teorias físicas [...] Quando os efeitos quânticos começam a ser importantes, por exemplo ao nível de átomos e fótons, a teoria clássica da computação, puramente abstrata, torna-se fundamentalmente inadequada." Contrariamente, as conclusões a que chegamos nesta seção indicam que, mesmo que a noção de computação seja analisada num escopo mais amplo, os limites da computação e da eficiência das máquinas é determinada por propriedades lógicas ou matemáticas dos modelos considerados,[12] embora os modelos sejam definidos seguindo princípios físicos para assegurar sua construibilidade. Nesta visão os modelos abstratos de computação recobram sua relevância, sendo úteis para estudar a classe de funções computáveis e as características quanto à eficiência de máquinas definidas por certo tipo de estruturas lógico-matemáticas, sejam estas 'razoáveis' ou não, de um ponto de vista físico.

1.4 Relativização da computação através de lógicas não-clássicas

Pode-se pensar que a maneira clássica, abstrata, de analisar os processos de computação já chegou no seu limite: a Tese de Church-Turing estabelece o limite do computável e a versão forte da Tese de Church-Turing indica os limites quanto à eficiência dos processos de computação (modelos que quebram estes limites estão fa-

[12] É importante mencionar que em [CD08] Walter Carnielli e Francisco A. Dória discutem a dependência dos fundamentos da computabilidade a pressupostos lógicos, concluindo que de fato a ciência da computação é dependente da lógica; alguns dos argumentos são baseados em nossa proposta de modelos de 'máquinas de Turing paraconsistentes' (Capítulo 2).

dados a serem considerados 'irrazoáveis'). O objetivo desta tese é propor um caminho abstrato, consistindo numa relativização através da lógica, para estudar a eficiências dos modelos de computação, justificando como os modelos obtidos podem ser considerados razoáveis e indicando como essa análise abstrata pode ser usada para um melhor entendimento de alguns modelos de computação 'concretos', como é o caso dos modelos de computação quânticos. Para isso vamos retomar algumas idéias da teoria 'clássica' da computação (i.e. algumas idéias de Gödel, Church, Turing, Post e outros).

Como descrito na Seção 1.1.1, Gödel e Church consideravam os sistemas lógicos (sob certas restrições de recursividade) como teorias que permitem substituir o raciocínio por operações mecânicas de fórmulas, e definiam a noção de computação como realizada dentro de um sistema formal, exigindo condições de recursividade sobre os axiomas e as regras de inferência para que os sistemas possam ser considerados como operando sob uma noção de conseqüência imediata. Dessa maneira, a computabilidade é explicada em termos de derivabilidade numa lógica. É importante mencionar que os sistemas formais considerados por Gödel e Church pressupõem a lógica clássica como lógica subjacente.

A noção de *circuito lógico* nos permite pensar de uma outra maneira a noção de computação, como sendo realizada através de uma lógica. Um circuito lógico consiste numa conexão acíclica de um número finito de portas, onde cada porta realiza a operação correspondente a um operador lógico. Cada circuito tem um número fixo n de entradas e um número fixo m de saídas. Normalmente os valores de entrada/saída são os valores 0 e 1 (representando, respectivamente, os valores de verdade 'falso' e 'verdadeiro'). Portanto, um circuito lógico computa uma função $f\colon \{0,1\}^n \to \{0,1\}^m$, e é possível mostrar que para toda função deste tipo existe um circuito lógico que a computa. Estamos pressupondo, por enquanto, que os operadores lógicos são aqueles da lógica clássica, e que as operações das portas estão determinadas pelas tabelas de verdade desses conectivos.

O parágrafo anterior parece indicar que através de circuitos lógicos qualquer função pode ser calculada e que podemos fundamentar a noção de computação em operações puramente lógicas (sem precisar de um outro formalismo para expressar o

que é computável). Porém, um circuito lógico pode ser usado somente para computar funções onde o comprimento das entradas (e as saídas) é fixo. Se quisermos usar circuitos lógicos para computar funções do tipo $f\colon \{0,1\}^+ \to \{0,1\}$, onde $\{0,1\}^+$ representa o conjunto de seqüências finitas de símbolos 0 e 1 (sem incluir a seqüência vazia),[13] precisamos definir uma *família de circuitos lógicos*, i.e., um conjunto infinito de circuitos lógicos contendo um circuito para cada comprimento de entrada.

Para uma família de circuitos lógicos ser considerada um algoritmo é necessário que se possa obter a descrição de cada circuito da família de maneira algorítmica, e é aqui que precisamos recorrer a uma outra definição formal de procedimento algorítmico. Usualmente se coloca a restrição de que a família de circuitos seja *uniforme*, indicando com isso que a família deve poder ser gerada por uma máquina de Turing. Famílias uniformes de circuitos booleanos conseguem computar as mesmas funções que as computadas por máquinas de Turing (cf. [Sav98]).

Parece impossível definir a noção de computação usando elementos puramente lógicos; neste sentido, podemos dizer que a noção de computação é 'pré-lógica'.[14] Contudo, as idéias acima abrem a possibilidade de definir modelos de computação sobre lógicas não-clássicas, e de estudar suas conseqüências. É isso que pretendemos fazer neste trabalho, sem procurar definir modelos de hiper-computação (pois os sistemas formais que vamos considerar satisfazem as restrições impostas por Gödel e Church), e centrando-nos em mostrar como a noção de complexidade algorítmica pode ser vista como relativa à lógica.

Nessa direção, no Capítulo 2, através da definição de teorias de primeira ordem que representam computações em máquinas de Turing e da substituição da lógica subjacente de tais teorias por lógicas paraconsistentes, definimos dois modelos diferentes de máquinas de Turing paraconsistentes; com diferentes características quanto à eficiência. A definição desses modelos segue, de certa maneira, a idéia de Gödel e Church de que computações podem ser definidas através de sistemas dedutivos, mas amplia radicalmente a perspectiva deles lançado mão de outras lógicas, o que permite

[13] Funções $f\colon \{0,1\}^+ \to \{0,1\}$ servem para representar 'problemas de decisão', i.e. problemas cujas únicas respostas possíveis são 'sim' ou 'não'. Todo problema (representável por uma função) pode ser traduzido num problema de decisão.

[14] Agradeço a Anderson de Araújo por me levar a refletir sobre este assunto.

explicar a eficiência dos modelos de computação em termos de suas lógicas subjacentes. Os sistemas formais considerados, em lugar de axiomatizar alguma teoria aritmética, axiomatizam máquinas de Turing, aproveitando dessa maneira a análise cuidadosa de Turing sobre o conceito de procedimento efetivo. Além disso, tornam a análise dos processos computáveis independente de qualquer teoria aritmética.[15]

A noção de circuito lógico é generalizada, no Capítulo 3, para permitir definir circuitos através de lógicas não-clássicas; em particular, lógicas paraconsistentes e lógicas modais. A generalização é realizada usando um método semântico denominado *cálculo de anéis de polinômios*, introduzido por Walter Carnielli em [Car05], o qual é apto para diversas lógicas, incluindo lógicas não caraterizáveis por matrizes finitas. O uso de 'variáveis ocultas' na constituição dos circuítos, para algumas lógicas não-clássicas, permitem um certo paralelismo no processo de computação, o que pode ser usado para definir algoritmos eficientes.[16]

Uma introdução à computação quântica é apresentada no Capítulo 4, e no Capítulo 5 estabelecemos algumas relações entre modelos de computação paraconsistente (introduzidos nos capítulos anteriores) e modelos de computação quântica. Desta maneira, mostramos como os modelos abstratos que propomos permitem explicar algumas das características da computação quântica a partir de uma nova perspectiva. Na outra direção, mostramos como os princípios da mecânica quântica permitem representar algumas características dos modelos de computação paraconsistentes, o que abre um caminho para sua possível construção.

Levando em consideração a relevância atual da pesquisa em computação quântica, e o interesse em determinar a verdadeira capacidade que os modelos de computação quântica possam oferecer quanto à eficiência, o valor de nosso trabalho radica em mostrar um caminho alternativo para estudar estas questões. Além disso, nossa proposta tem a vantagem metodológica de permitir estudar questões de complexidade

[15] A partir das idéias apresentadas no Capítulo 2, e algumas idéias apresentadas no Capítulo 5, foi escrito o artigo [AC09], o qual já foi publicado no *Journal of Logic and Computation*.

[16] A generalização do modelo de circuitos lógicos para lógicas não-clássicas foi publicada em [AC07]. Além disso, na definição do modelo de circuitos lógicos para a lógica modal $S5$, foi necessário definir um cálculo de anéis de polinômios para essa lógica; com esse resultado se escreveu o artigo [AC09], o qual está sendo submetido a publicação numa revista internacional arbitrada.

algorítmica de um ponto de vista bastante abrangente, não se limitando apenas a tentar explicar o que a estrutura da mecânica quântica nos capacita, mas mostrando como propriedades lógicas subjacentes a alguns modelos de computação determinam sua eficiência.

Embora os capítulos 2 a 5 sejam bastante técnicos, as definições e demonstrações ali apresentadas constituem as evidências que sustentam a posição filosófica que estamos querendo defender: a eficiência dos processos de computação pode ser vista como relativa a uma lógica subjacente ao modelo de computação. Sob essa mesma perspectiva, faz sentido estudar o poder computacional dos modelos de computação quântica através do estudo de uma lógica subjacente a tais modelos. Por outro lado, como será descrito no Capítulo 6, a 'interpretação paraconsistente' de algumas características da mecânica quântica parece ser bastante frutífera. Devido à relevância filosófica que tem oferecer uma boa interpretação do formalismo quântico, de tal maneira que possa ser geralmente aceita, nosso trabalho também pode ser valioso por sugerir novas alternativas nessa direção.

Para facilitar o entendimento deste trabalho a pessoas de diversas áreas do conhecimento, incluímos dois apêndices: o apêndice A é uma introdução à lógica paraconsistente e à lógica modal, que são usadas na definição dos modelos de computação aqui introduzidos (capítulos 2 e 3); e o apêndice B contem alguns elementos de álgebra linear, necessários para a definição dos postulados da mecânica quântica e dos modelos de computação quântica (Capitulo 4).

Capítulo 2

Máquinas de Turing paraconsistentes

Como mencionado no Capítulo 1, Turing em [Tur36] define um modelo abstrato de *máquina automática* (conhecido como *máquina de Turing (MT)* na literatura contemporânea) e apresenta argumentos, bastante convincentes, sobre a adequação desse modelo para capturar a noção intuitiva de procedimento efetivo. Além disso, mostra que existem problemas que não podem ser computados no seu modelo, em particular, o problema da parada. A partir daí, indicando como expressar computações de suas máquinas em teorias de primeira ordem, Turing demonstra que a lógica de predicados de primeira ordem é decidível se e somente se o problema da parada é decidível (i.e., computável). Portanto, por redução ao absurdo, fica demonstrado que a lógica de predicados de primeira ordem é indecidível. Esta demonstração é apresentada de maneira simplificada por Richard Büchi em [Büc62], e uma versão mais recente e mais simples é apresentada por George Boolos e Richard Jeffrey em [BJ89, Cáp. 10].

A seguir (Seção 2.2), tomando como referência [BJ89] e acrescentando novos axiomas, definimos um método para construir teorias que representam computações em *máquinas de Turing determinísticas (MTDs)*. Para mostrar a adequação das teorias obtidas definimos formalmente o que significa 'representar uma computação numa teoria' e provamos que nossas teorias satisfazem tal definição. Em seguida mostramos que, se aplicado para axiomatizar computações em *máquinas de Turing*

não-determinísticas (MTNDs), esse método pode levar a teorias contraditórias, e portanto triviais (dado que estamos pressupondo a lógica clássica como lógica subjacente). É claro que a trivialização significa aqui um colapso indesejável da teoria de computação subjacente.

Para evitar a trivialização das teorias que representam computações em MTNDs existem, pelo menos, duas opções:

1. o caminho 'clássico' que consistiria em modificar as teorias de maneira a evitar contradições; e

2. o caminho 'paraconsistente', que partiria da possibilidade de mudar a lógica subjacente das teorias por uma lógica paraconsistente, não banindo as contradições mas tolerando-as (sem levar à trivialização).

O caminho clássico permite adequar as teorias para que possam representar computações em MTNDs (tal como é definido para esse tipo específico de máquinas), mas é estéril no sentido em que não permite a definição de um novo modelo de computação. O caminho paraconsistente, pelo contrário, abre a possibilidade de definir um novo modelo de computação através da interpretação das teorias contraditórias mas não triviais. É precisamente este segundo caminho que nos interessa seguir aqui.

Um primeiro movimento na direção do caminho paraconsistente consiste em definir um modelo de *máquinas de Turing paraconsistentes (MTPs)* usando a lógica $LFI1^*$ (Seção 2.3). Mostramos que as MTPs permitem um certo tipo de paralelismo, mas podem, porém, ser simuladas em tempo polinomial por MTDs e, portanto, não representam nenhuma vantagem quanto à eficiência respeito ao modelo clássico de MT.[1]

Um segundo modelo de computação paraconsistente é introduzido na Seção 2.3. O modelo é definido seguindo a mesma metodologia da definição do modelo anterior, mas usando uma lógica paraconsistente definida dentro da lógica modal $S5$ (em lugar de $LFI1^*$). Nessa lógica, além de definir um operador de negação paraconsistente, é definido um operador de conjunção 'não separável', o qual permite estabelecer

[1]Este primeiro modelo de MTPs já tinha sido definido em [Agu06], mas a demonstração de que pode ser simulado em tempo polinomial por MTDs é apresentada pela primeira vez aqui.

relações entre diferentes valores de diversos elementos da máquina (chamaremos estas máquinas de *máquinas de Turing paraconsistentes não-separáveis (MTPNSs)*). A característica de não-separabilidade permite aproveitar de uma melhor maneira o processamento em paralelo, tornando possível computar em tempo polinomial (nas MTPNSs) qualquer problema em **NP**. Desta maneira mostramos que, se $\mathbf{P} \neq \mathbf{NP}$ (como muitos pesquisadores em complexidade algorítmica acreditam), a noção de complexidade algorítmica resultaria relativa às lógicas subjacentes aos modelos de computação.

A 'teoria da computação paraconsistente' já é mencionada em [SC00] como uma área de pesquisa emergente. Nesse artigo as 'máquinas dialéticas' são sumariamente descritas como máquinas de Turing que agem em concordância com a lógica dialética (um tipo de lógica paraconsistente) na presença de certas contradições, mas não é apresentada nenhuma definição concreta de um modelo de computação paraconsistente. As máquinas dialéticas de 'tipo 1' são descritas como MTs tais que, quando 'encontram' uma contradição, podem 'proceder' com a computação de maneira satisfatória. Os autores indicam que "não é difícil descrever como uma máquina pode encontrar uma contradição: para algum enunciado A, ambos A e $\sim A$ aparecem nas saídas ou nas entradas." (cf. [SC00, pág. 196, tradução nossa]). No entanto, não é claramente especificado como A e $\sim A$ podem aparecer nas entradas ou saídas da máquina.[2] Além disso, também não é especificado como as máquinas dialéticas procedem quando encontram uma contradição. Logo depois, os autores esboçam o que eles chamam de 'idéia central da teoria da computabilidade paraconsistente', que consiste na afirmação de que "tais máquinas de Turing [as máquinas de Turing dialéticas de tipo 1] podem ser usadas para computar funções diagonais que são classicamente vistas como não-computáveis." (cf. [SC00, pág. 197, tradução nossa]).

[2] Varias questões podem ser colocadas sobre a vaga descrição dada em [SC00]: o 'enunciado A' corresponde a um símbolo de entrada/saída da máquina ou corresponde a uma seqüência desses símbolos? Como é representado $\sim A$ numa máquina dialética? $\sim A$ corresponde simplesmente a um símbolo (ou seqüência de símbolos), que se pressupõe como sendo contraditório a A, ou são as entradas e saídas desse tipo de máquinas marcadas com sinais positivos e negativos? A e $\sim A$ precisam aparecer na mesma célula para serem considerados contraditórios? Nesse caso, como podem aparecer juntos A e $\sim A$?

Seus argumentos nesta direção são obscuros devido à falta de uma definição clara do modelo de computação adotado.

Um segundo tipo de máquinas paraconsistentes, as máquinas dialéticas de 'tipo 2', são brevemente descritas em [SC00] como "máquinas cuja *metalógica* é dialética: para uma tal máquina, M, um de cujos estados é x, 'M está em x' e '$\sim$ (M está em x)' podem ser ambos verdadeiros." (cf. [SC00, pág. 197, tradução nossa]). Os autores então propõem duas questões à comunidade paraconsistente: (i) podem as máquinas dialéticas de tipo 2 computar funções classicamente não computáveis? (ii) continua o problema da parada sendo válido no caso das máquinas paraconsistentes?

Nos termos usados em [SC00], as MTPs e as MTPNSs propostas aqui corresponderiam a máquinas paraconsistente de tipo 2. Como é mostrado nas seções 2.3.2 e 2.4.2, nem as MTPs nem as MTPNSs conseguem computar problemas não computáveis em MTs clássicas. Desta maneira, damos uma resposta negativa (restrita aos modelos por nós propostos) à primeira pergunta formulada em [SC00], mostrando que a hiper-computação (i.e. a computação de funções não-Turing-computáveis) não é uma propriedade inerente aos modelos de computação paraconsistente. Como conseqüência disto, levando em consideração que as MTs classicas são casos especiais de MTPs e de MTPNSs, damos também uma resposta à segunda questão colocada em [SC00]: o problema da parada continua sendo válido nas MTPs e nas MTPNSs, portanto, a não validade do problema da parada não é nenhuma característica subjacente aos modelos de computação paraconsistente.

Antes de descrever o método de axiomatização de computações em MTs (que será usado para definir as MTPS e as MTPNSs), apresentamos uma definição formal das MTs clássicas. Também definimos formalmente os conceitos de MTDs, MTNDs, MTs probabilísticas, as classes de complexidade algorítmica **P**, **NP** e **BPP**, e o conceito de **NP**-completude.

2.1 Definições preliminares

Informalmente, uma MT pode ser descrita como sendo constituída por uma cabeça de leitura/escrita e uma fita unidimensional potencialmente infinita em ambas as

direções (veja por exemplo [CE05]). A fita é dividida em células, cada uma das quais com capacidade de conter um único símbolo. Em cada instante discreto de tempo a máquina se encontra em um determinado *estado*, e com a cabeça lendo uma célula específica da fita, a qual é considerada como sendo a *posição atual* da máquina. A *configuração atual* da máquina consiste na junção do estado atual e do símbolo que está sendo lido na posição atual. O comportamento da máquina está determinado por um conjunto de instruções, as quais indicam a operação a se realizar dependendo da configuração atual. As únicas operações que a máquina pode realizar são: escrever um novo símbolo na posição atual (substituindo o símbolo anterior) ou realizar um movimento da cabeça à esquerda ou direita, passando a uma célula subseqüente da atual. Os estados, os símbolos de leitura/escrita e as instruções são conjuntos finitos.

Formalmente, uma MT pode ser definida da seguinte maneira:

Definição 2.1. Uma *máquina de Turing (MT)* $\mathcal{M}$ é uma estrutura matemática $\langle Q, \Sigma, M, I\rangle$ onde:

- $Q = \{q_1, q_2, \ldots, q_n\}$ (com $Q \neq \emptyset$) é um conjunto finito de *estados* da máquina.
- $\Sigma = \{s_1, s_2, \ldots, s_m\}$ (com $(\Sigma - \{s_1\} \neq \emptyset)$) é um conjunto finito de *símbolos de entrada/saída* (ou *alfabeto de entrada/saída*). Por convenção s_1 representa o símbolo vazio.
- $M = \{R, L\}$ é o conjunto de *movimentos* da máquina, onde R indica movimento a direita e L indica movimento a esquerda.
- $I = \{i_1, i_2, \ldots, i_{\mathsf{k}}\}$ (com $I \neq \emptyset$) é um conjunto finito de *instruções*, onde cada i_{j} é uma quádrupla[3] de alguma das formas:

$$q_i s_j s_k q_l, \tag{I}$$

$$q_i s_j R q_l, \tag{II}$$

$$q_i s_j L q_l. \tag{III}$$

[3]Na definição original de Turing as instruções são definidas por meio de quíntuplas (cf. [Tur36]). Contudo, é possível demonstrar que ambas as definições são equivalentes. Utilizamos aqui a definição por meio de quádruplas para facilitar o processo de axiomatização que será definido a seguir.

O tipo de instrução (I) indica que, quando a máquina estiver no estado q_i, lendo o símbolo s_j, vai escrever o símbolo s_k e vai passar ao estado q_l. O tipo de instrução (II) indica que, quando a máquina estiver no estado q_i, lendo o símbolo s_j, vai se deslocar uma célula a direita e vai passar ao estado q_l. O tipo de instrução (III) é similar ao (II), com a diferença que o movimento se dá a esquerda.

Uma *computação* numa MT começa no instante de tempo $t = 0$, com o dado de entrada (uma cadeia de símbolos de Σ) escrito na fita a partir da posição 0 até a posição $c-1$ (considerando que as posições da fita são enumeradas por números inteiros e que o comprimento do dado de entrada é c). Por convenção, a máquina começa sua execução no estado q_1, lendo a posição 0 da fita. Em cada instante discreto de tempo a máquina executa no máximo uma instrução, dependendo da configuração atual. Se a máquina alcançar uma configuração na qual não tem nenhuma instrução a ser executada, a máquina pára, e o resultado da computação é dado pelos símbolos escritos na fita.

O determinismo/não-determinismo nas MTs é definido da seguinte maneira:

Definição 2.2. Seja $\mathcal{M} = \langle Q, \Sigma, M, I\rangle$ uma MT. $\mathcal{M}$ é uma *MT não-determinística (MTND)* se existir instruções $i_x, i_y \in I$ $(x \neq y)$ tais que $i_x = q_i s_j \mathrm{OP}_x q_l$ e $i_y = q_i s_j \mathrm{OP}_y q_k$ (onde $q_i, q_l, q_k \in Q$; $s_j \in \Sigma$ e $\mathrm{OP}_x, \mathrm{OP}_y \in \Sigma \cup \{R, L\}$). Caso contrário $\mathcal{M}$ é uma *MT determinística (MTD).*

A definição anterior estabelece que uma MT é *não-determinística* se contêm duas instruções com os mesmos dois símbolos iniciais. Levando em conta que esses dois símbolos são os que determinam quando a instrução é executada, as MTNDs são aquelas MTs que podem chegar em configurações para as quais existem múltiplas instruções factíveis de serem executadas, tais configurações serão chamadas de *configurações ambíguas.* Quando uma MTND alcança uma configuração ambígua, escolhe e executa uma única instrução dentre as múltiplas possíveis.

As MTs probabilísticas são casos especiais de MTNDs:

Definição 2.3. Uma *máquina de Turing probabilística (MTPr)* é uma MTND tal

que, ao chegar numa configuração ambígua, escolhe a instrução a ser executada de acordo com uma distribuição de probabilidade.

Para definir a classe de complexidade algorítmica **BPP** é conveniente considerar que as MTPrs escolhem as instruções a serem executadas de acordo com distribuições de probabilidade uniformes, i.e. que no caso de existirem n possíveis instruções a serem executadas qualquer uma delas será executada com probabilidade $\frac{1}{n}$. De agora em diante, sempre que fizermos referência a uma MTPr estaremos considerando que a máquina faz escolha de instruções de acordo com uma distribuição de probabilidade uniforme.

O *tempo de computação* numa MT (de qualquer dos tipos anteriormente descritos) é simplesmente o número de instruções executadas, contadas a partir do começo da computação (tempo $t = 0$). A maneira como o tempo de computação é escalonado em relação ao tamanho do dado de entrada é definido assim:

Definição 2.4. Seja $\mathcal{M}$ uma MT com alfabeto de entrada/saída Σ e seja $f\colon \mathbb{N} \to \mathbb{N}$ uma função. $\mathcal{M}$ *termina em tempo* $f(c)$ se para toda entrada $\alpha \in \Sigma^*$ (onde Σ^* denota o conjunto de todas as possíveis cadeias finitas de símbolos pertencentes a Σ) de comprimento $|\alpha| = c$ o tempo de computação de $\mathcal{M}$ é no máximo $f(c)$, para todo $c \in \mathbb{N}$.

Usualmente as classes de complexidade algorítmica são definidas em termos de 'decisão' de linguagens, onde uma *linguagem* consiste num conjunto de seqüências (ou cadeias) finitas de símbolos pertencentes a um determinado *alfabeto* (ou conjunto de símbolos) Σ. Antes de definir as condições sobre as quais uma MT 'decide uma linguagem', é preciso definir como uma MT 'aceita' ou 'rejeita' cadeias de símbolos. Para isso, vamos partir do suposto de que toda MT tem dois estados distinguidos, um *estado de aceitação* q_s e um *estado de rejeição* q_n, e de que o estado final de toda computação numa MT (se a MT parar) é um desses dois estados. Além disso, vamos denotar por $\mathcal{M}(\alpha)$ o fato da máquina $\mathcal{M}$ começar com o dado de entrada α.

Definição 2.5. Seja $\mathcal{M}$ uma MT com alfabeto de entrada/saída Σ e seja $\alpha \in \Sigma^*$:

- Se $\mathcal{M}$ é uma MTD: $\mathcal{M}$ *aceita* (resp. *rejeita*) α se a computação de $\mathcal{M}(\alpha)$ pára e deixa a máquina no estado q_s (resp. q_n).

- Se $\mathcal{M}$ é uma MTND: $\mathcal{M}$ *aceita* α se existir uma computação de $\mathcal{M}(\alpha)$ (com possíveis escolhas não-determinísticas de instruções a serem executadas) que pare e deixe a máquina no estado q_s; por outro lado, $\mathcal{M}$ *rejeita* α se toda computação de $\mathcal{M}(\alpha)$ pára e deixa a máquina no estado q_n.

- Se $\mathcal{M}$ é uma MTPr: $\mathcal{M}$ *aceita* (resp. *rejeita*) α se todas as computações de $\mathcal{M}(\alpha)$ param e pelo menos $\frac{2}{3}$ dos caminhos de computação terminam no estado q_s (resp. q_n), i.e. se a probabilidade de $\mathcal{M}(\alpha)$ terminar no estado q_s (resp. q_n) é maior ou igual a $\frac{2}{3}$ (nesta definição o valor $\frac{2}{3}$ é arbitrário, de fato, a definição pode ser feita de maneira equivalente usando qualquer valor constante c tal que $\frac{1}{2} < c < 1$, cf. [AB09]).

Podemos agora definir o que significa decidir uma linguagem por meio de uma MT.

Definição 2.6. Seja $\mathcal{M}$ uma MT com alfabeto de entrada/saída Σ e seja $L \subseteq \Sigma^*$ uma linguagem. $\mathcal{M}$ *decide a linguagem* L se, para todo $\alpha \in \Sigma^*$, tem-se que: se $\alpha \in L$ então $\mathcal{M}$ aceita α, caso contrário $\mathcal{M}$ rejeita α.

Note que a definição de decisão de uma linguagem exige que a computação sempre pare. As classes de complexidade algorítmica **P** (tempo polinomial determinístico), **NP** (tempo polinomial não-determinístico) e **BPP** (tempo polinomial probabilístico com erro limitado) são então definidas como as seguintes classes de linguagens:

$$\begin{aligned}
\mathbf{P} =& \{L \mid \text{existe uma MTD que decide } L \text{ e termina em tempo } f(c) = c^k, \\
& \text{para alguma constante } k \in \mathbb{N}\}, \\
\mathbf{NP} =& \{L \mid \text{existe uma MTND que decide } L \text{ e termina em tempo } f(c) = c^k, \\
& \text{para alguma constante } k \in \mathbb{N}\}. \\
\mathbf{BPP} =& \{L \mid \text{existe uma MTPr que decide } L \text{ e termina em tempo } f(c) = c^k, \\
& \text{para alguma constante } k \in \mathbb{N}\}.
\end{aligned}$$

Para definir a classe de linguagens **NP**-completas é necessário definir antes o que significa uma 'redução polinomial'. Nessa definição vamos considerar que uma MTD,

com alfabeto de entrada/saida Σ, *computa a função* $f\colon \Sigma_1^* \to \Sigma_2^*$ (onde $\Sigma_1, \Sigma_2 \subseteq \Sigma$) se para todo $\alpha \in \Sigma_1^*$ o processo de computação de $\mathcal{M}(\alpha)$ sempre termina e deixa na fita a cadeia de símbolos $f(\alpha)$ (a partir da posição em que a máquina parou, para evitar problemas técnicos).

Definição 2.7. Uma linguagem $L_1 \subseteq \Sigma_1^*$ é *reduzível em tempo polinomial* a outra linguagem $L_2 \subseteq \Sigma_2^*$ (o que será denotado por $L_1 \leq_p L_2$) se existir uma MTD que compute, em tempo polinomial, uma função $f\colon \Sigma_1^* \to \Sigma_2^*$ tal que $\alpha \in L_1$ se e somente se $f(\alpha) \in L_2$, para todo $\alpha \in \Sigma_1^*$.

É fácil mostrar que se $L_1 \leq_p L_2$ e $L_2 \in \mathbf{P}$ então $L_1 \in \mathbf{P}$. Vamos agora definir o que é uma linguagem **NP**-completa.

Definição 2.8. Uma linguagem L é ***NP**-completa* se $L \in \mathbf{NP}$ e, para toda linguagem $L_i \in \mathbf{NP}$, $L_i \leq_p L$.

Como conseqüências das definições anteriores temos que $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{NP}$ e que se existir alguma linguagem **NP**-completa em **P** então $\mathbf{NP} \subseteq \mathbf{P}$ (em tal caso teríamos que $\mathbf{P} = \mathbf{BPP} = \mathbf{NP}$); porém é amplamente acreditado que $\mathbf{P} \neq \mathbf{NP}$. Por outro lado, embora ainda não se conheça nenhuma demonstração definitiva de se a inclusão $\mathbf{P} \subseteq \mathbf{BPP}$ é própria ou não, existem evidências para pensar que $\mathbf{P} = \mathbf{BPP}$ (cf. [AB09]).

A seguir, definimos o método para axiomatizar computações em MTs em teorias de primeira ordem.

2.2 Axiomatizando computações de máquinas de Turing

Para axiomatizar a computação de uma MTD $\mathcal{M}$, com uma seqüência de símbolos α como dado de entrada, vamos inicialmente definir uma teoria de primeira ordem $\Delta_{LPC}(\mathcal{M}(\alpha))$.[4] Esta teoria é definida sobre a linguagem $\mathcal{L} =$

[4] Usamos o sub-índice LPC para ressaltar o fato de que a lógica subjacente à teoria Δ é a lógica de primeira ordem clássica (LPC), e acrescentamos o parâmetro $\mathcal{M}(\alpha)$ para indicar a máquina de Turing e o dado de entrada em consideração.

$\{Q_1, \ldots, Q_n, S_1, \ldots, S_m, <, ', 0\}$, onde os símbolos Q_i, S_j e $<$ são símbolos de predicados binários, $'$ é um símbolo de função unária e 0 é um símbolo de constante. Na *interpretação intensional* das sentenças em $\Delta_{LPC}(\mathcal{M}(\alpha))$, que denotaremos por $\mathcal{I}$, as variáveis são interpretadas como números inteiros e os símbolos em $\mathcal{L}$ são interpretados da seguinte maneira:

- $Q_i(t, x)$ indica que, no instante de tempo t, $\mathcal{M}(\alpha)$ está no estado q_i, com a cabeça de leitura/escrita localizada na posição x da fita;
- $S_j(t, x)$ indica que, no instante de tempo t, $\mathcal{M}(\alpha)$ contém o símbolo s_j na célula da posição x da fita;
- $< (x, y)$ indica que x é menor do que y, na ordem padrão dos números inteiros;
- $'(x)$ indica o sucessor de x;
- 0 indica o número 0.

Para simplificar a notação, usaremos $x < y$ em lugar de $< (x, y)$, x' em lugar de $'(x)$ e $x \neq y$ em lugar de $\neg(x = y)$. A teoria $\Delta_{LPC}(\mathcal{M}(\alpha))$ é constituída pelos seguintes axiomas:

- Axiomas para estabelecer as propriedades dos símbolos $'$ e $<$:

$$\forall z \exists x (z = x'), \tag{A1}$$
$$\forall z \forall x \forall y (((z = x') \wedge (z = y')) \rightarrow (x = y)), \tag{A2}$$
$$\forall x \forall y \forall z (((x < y) \wedge (y < z)) \rightarrow (x < z)), \tag{A3}$$
$$\forall x (x < x'), \tag{A4}$$
$$\forall x \forall y ((x < y) \rightarrow (x \neq y)). \tag{A5}$$

- Um axioma por cada instrução i_j de $\mathcal{M}$. A definição de tais axiomas depende do tipo da instrução. Os esquemas apresentados a seguir indicam como construir

os axiomas para instruções de tipo (I), (II) e (III) respectivamente:

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x) \wedge S_k(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Big) \Bigg), \quad (\mathrm{A}i_{\mathrm{j}}\ (\mathrm{I}))$$

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x') \wedge \forall y \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Bigg), \quad (\mathrm{A}i_{\mathrm{j}}\ (\mathrm{II}))$$

$$\forall t \forall x \Bigg(\Big(Q_i(t,x') \wedge S_j(t,x') \Big) \rightarrow \Big(Q_l(t',x) \wedge \forall y \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Bigg). \quad (\mathrm{A}i_{\mathrm{j}}\ (\mathrm{III}))$$

- Um axioma para especificar a configuração[5] inicial da máquina. Considerando o dado de entrada $\alpha = s_{i_0} s_{i_1} \ldots s_{i_{c-1}}$, com comprimento c, este axioma é definido através do seguinte esquema:

$$Q_1(0,0) \quad \wedge \quad \left(\bigwedge_{j=0}^{c-1} S_{i_j}(0,0^j) \right) \quad \wedge \quad \forall y \left(\left(\bigwedge_{j=0}^{c-1} y \neq 0^j \right) \rightarrow S_1(0,y) \right), \quad (\mathrm{A}t_0)$$

  onde 0^j representa a iteração da função sucessor j vezes sobre a constante 0.

Boolos e Jeffrey em [BJ89] definem adicionalmente uma sentença H que representa as condições nas quais $\mathcal{M}(\alpha)$ pára. Eles mostram que $\Delta_{LPC}(\mathcal{M}(\alpha)) \vdash H$ se e somente se a MT $\mathcal{M}$ com a entrada α pára. Dessa maneira, a decidibilidade da LPC implica a computabilidade do problema da parada. Portanto, como é conhecido

[5] De aqui em diante a expressão 'configuração da máquina' será usada num sentido mais amplo, não indicando somente o estado atual e o símbolo que está sendo lido pela máquina mas abrangendo também o conteúdo das outras posições da fita nesse instante de tempo.

que o problema da parada não é computável (em máquinas de Turing), conclui-se por redução ao absurdo que LPC não é decidível (por máquinas de Turing ou por formalismos equivalentes).

Para os objetivos de Boolos e Jeffrey as teorias $\Delta_{LPC}(\mathcal{M}(\alpha))$ são suficientemente fortes, mas nosso objetivo aqui é representar computações em MTDs de uma maneira precisa. A seguir vamos definir formalmente o que significa 'representar computações' em MTDs, e vamos mostrar que novos axiomas precisam ser adicionados às teorias $\Delta_{LPC}(\mathcal{M}(\alpha))$ para atingir nosso objetivo. A definição de representação de computações em MTDs aqui proposta (Definição 2.11) é baseada nas definições de representação de funções e relações (definições 2.9 e 2.10) introduzidas por Alfred Tarski, Andrzej Mostowski e Raphael M. Robinson em [TMR53].

Definição 2.9. Seja f uma função de aridade k, Δ uma teoria arbitrária e $\varphi(x_1, \ldots, x_k, x)$ uma fórmula bem-formada em Δ (com $k+1$ variáveis livres). A função f é *representada* por φ em Δ se $f(m_1, \ldots, m_k) = n$ implica:

1. $\Delta \vdash \varphi(\bar{m_1}, \ldots, \bar{m_k}, \bar{n})$,

2. se $n \neq p$ então $\Delta \vdash \neg\varphi(\bar{m_1}, \ldots, \bar{m_k}, \bar{p})$, e

3. $\Delta \vdash \varphi(\bar{m_1}, \ldots, \bar{m_k}, x) \rightarrow x = \bar{n}$;

onde $\bar{n}$ representa o numeral n.

Definição 2.10. Seja R uma relação de aridade k, Δ uma teoria arbitrária e $\varphi(x_1, \ldots, x_k)$ uma fórmula bem formada em Δ (com $k+1$ variáveis livres). A relação R é *representada* por φ em Δ se:

1. $(m_1, \ldots, m_k) \in R$ implica $\Delta \vdash \varphi(\bar{m_1}, \ldots, \bar{m_k})$, e

2. $(m_1, \ldots, m_k) \notin R$ implica $\Delta \vdash \neg\varphi(\bar{m_1}, \ldots, \bar{m_k})$.

Definição 2.11. Seja $\mathcal{M}$ uma MTD, α o dado de entrada para $\mathcal{M}$, e $\mu(\mathcal{M}(\alpha)) = \langle \mathbb{Z}, Q_1^\mu, Q_2^\mu, \ldots, Q_n^\mu, S_1^\mu, S_1^\mu, \ldots, S_m^\mu, <^\mu, '^\mu, 0^\mu \rangle$ a estrutura determi-

nada pela interpretação intensional $\mathcal{I}$.[6] Uma teoria Δ, na linguagem $\mathcal{L} = \{Q_1, Q_2, \ldots, Q_n, S_0, S_1, \ldots, S_{m-1}, <, ', 0\}$, *representa a computação de* $\mathcal{M}(\alpha)$ se:

1. $<^\mu$ é representada por $\varphi(x, y) := x < y$ em Δ,
2. $'^\mu$ é representada por $\varphi(x, y) := x' = y$ em Δ,
3. Q_i^μ é representada por $\varphi(x, y) := Q_i(x, y)$ em Δ (para $i = 1, \ldots, n$), e
4. S_j^μ é representada por $\varphi(x, y) := S_j(x, y)$ em Δ (para $j = 1, \ldots, m$).

Teorema 2.1. *Seja $\mathcal{M}$ uma MTD e α o dado de entrada para $\mathcal{M}$. A teoria $\Delta_{LPC}(\mathcal{M}(\alpha))$ não representa a computação de $\mathcal{M}(\alpha)$ (no sentido estrito da Definição 2.11).*

Demonstração. As relações Q_i^μ e S_j^μ não são representadas, respectivamente, pelas fórmulas $Q_i(x, y)$ e $S_j(x, y)$ em Δ, pois a condição 2 da Definição 2.10 não é satisfeita para essas relações: quando $\mathcal{M}(\alpha)$ está num estado q_i, no tempo t e na posição x, não está em nenhum outro estado q_j (com $j \neq i$). Nesse caso, temos que $\Delta_{LPC}(\mathcal{M}(\alpha)) \vdash Q_i(\bar{t}, \bar{x})$ (por demonstração em [BJ89, Chap. 10]). Mas, por outro lado temos que $\Delta_{LPC}(\mathcal{M}(\alpha)) \nvdash \neg Q_j(\bar{t}, \bar{x})$: pense por exemplo numa MT 'não-padrão', com as mesmas instruções de $\mathcal{M}$, mas que começa a computação simultaneamente nos estados q_1 e q_2, na posição 0 da fita. Esta MT não-padrão valida todos os axiomas em $\Delta_{LPC}(\mathcal{M}(\alpha))$, mas não valida $\neg Q_2(0, 0)$. Uma situação similar ocorre com as relações S_j. É possível definir MTs não-padrões permitindo múltiplos símbolos nas células da fita, múltiplos estados e posições da máquina de maneira simultânea, e diferentes configurações da máquina em tempos anteriores a 0 e posteriores ao tempo de parada (se $\mathcal{M}(\alpha)$ pára), de maneira que a máquina valide todos os axiomas em $\Delta_{LPC}(\mathcal{M}(\alpha))$. □

O Teorema 2.1 mostra que é necessário estender as teorias $\Delta_{LPC}(\mathcal{M}(\alpha))$, para impedir interpretações não-padrões e permitir representar computações de acordo

[6] $\mathbb{Z}$ representa os números inteiros, as relações Q_i^μ expressam duplas de instantes de tempo e posições da fita nas quais $\mathcal{M}(\alpha)$ esta no estado q_i, as relações S_j^μ expressam duplas de instantes de tempo e posições da fita nas quais $\mathcal{M}(\alpha)$ contem o símbolo s_j, $<^\mu$ é a relação de ordem padrão em $\mathbb{Z}$, $'^\mu$ é a função sucessor e 0^μ é o número 0.

com a Definição 2.11. Conseqüentemente, definimos a seguir as teorias $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ como *teorias intrínsecas às computações* das respetivas $\mathcal{M}(\alpha)$, através da adição de novos axiomas às teorias $\Delta_{LPC}(\mathcal{M}(\alpha))$. Na especificação dos novos axiomas vamos supor que antes de começar a computação (instantes de tempo $t < 0$) a máquina não está em nenhum estado e não tem nenhum símbolo (nem sequer o símbolo vazio) nas células da fita. Também vamos supor que depois de terminar a computação (caso a computação termine) a máquina continua infinitamente na última configuração alcançada. Os novos axiomas são definidos da seguinte maneira:

- Um axioma para estabelecer a configuração de $\mathcal{M}(\alpha)$ antes do começo da computação:

$$\forall t \forall x \left((t < 0) \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(t,x) \right) \wedge \left(\bigwedge_{j=1}^{m} \neg S_j(t,x) \right) \right) \right). \qquad (\mathrm{A}t_{-})$$

- Um axioma para estabelecer a configuração de $\mathcal{M}(\alpha)$ depois de terminar a computação (caso a computação termine):

$$\forall t \forall x \left(\neg \left(\bigvee_{i,j \text{ Inic.}} \left(Q_i(t,x) \wedge S_j(t,x) \right) \right) \rightarrow \forall u \forall y \left(t < u \rightarrow \right. \right.$$
$$\left. \left. \left(\left(\bigwedge_{i=1}^{n} (Q_i(t,y) \rightarrow Q_i(u,y)) \right) \wedge \left(\bigwedge_{j=1}^{m} (S_j(t,y) \rightarrow S_j(u,y)) \right) \right) \right) \right), \quad (\mathrm{A}t_{+})$$

  onde o subíndice 'i,j Inic.', na disjunção, indica que são consideradas somente combinações de i,j tais que $q_i s_j$ coincidem com os dois primeiros símbolos de alguma instrução de $\mathcal{M}$.

- Um axioma por cada símbolo de estado q_i de $\mathcal{M}$, para estabelecer a unicidade de estado e posição da máquina em todo instante de tempo:

$$\forall t \forall x \left(Q_i(t,x) \rightarrow \left(\left(\bigwedge_{j \neq i} \neg Q_j(t,x) \right) \wedge \forall y \left(y \neq x \rightarrow \bigwedge_{i=1}^{n} \neg Q_i(t,y) \right) \right) \right). \qquad (\mathrm{A}q_i)$$

- Um axioma por cada símbolo de leitura/escrita s_i de $\mathcal{M}$, para estabelecer a unicidade de símbolos em cada posição da fita em todo instante de tempo:

$$\forall t \forall x \left(S_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg S_j(t,x) \right). \qquad (\mathrm{A}s_j)$$

Teorema 2.2. *Seja $\mathcal{M}$ uma MTD e α o dado de entrada para $\mathcal{M}$. A teoria $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ representa a computação de $\mathcal{M}(\alpha)$.*

Demonstração. É fácil demonstrar a representação da relação $<^{\mu}$ e da função $'^{\mu}$. A representação das relações Q_i^{μ} e S_j^{μ} segue da prova em [BJ89] e da aplicação direta dos novos axiomas introduzidos nas teorias $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$. □

O seguinte teorema mostra que o método de axiomatização, até aqui definido, quando usado para axiomatizar computações em MTNDs pode levar a teorias contraditórias.

Teorema 2.3. *Seja $\mathcal{M}$ uma MTND e α o dado de entrada para $\mathcal{M}$. Se $\mathcal{M}(\alpha)$ alcança uma configuração ambígua então a teoria $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ é contraditória.*

Demonstração. Pela demonstração em [BJ89], pode se deduzir de $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ uma fórmula que expressa a configuração ambígua. Então, usando os teoremas correspondentes às instruções que podem ser executadas em tal configuração, podem se deduzir fórmulas expressando multiplicidade de estados, posições ou símbolos em algumas células da fita. Depois, usando algum dos axiomas $(\mathrm{A}q_i)$ ou $(\mathrm{A}s_j)$ obtêm-se uma contradição. □

Em [Odi89, p. 48] Piergiorgio Odifreddi estabelece uma condição de 'consistência' na sua definição de MT, indicando com isso que a máquina não pode conter instruções 'contraditórias', i.e., instruções realizando operações diferentes e para ser executadas na mesma configuração da máquina. Desta maneira, a definição de Odiffredi corresponde à definição de MTD, definindo depois as MTNDs através da eliminação da condição de consistência, indicando que uma única instrução deve ser escolhida em caso de se alcançar uma configuração ambígua. O Teorema 2.3 mostra que a idéia de

consistência em MTs, proposta por Odifreddi, corresponde à consistência das teorias intrínsecas das computações (i.e., das teorias $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$).

Note que as contradições nas teorias intrínsecas das computações para MTNDs surgem devido ao uso de múltiplos axiomas da forma (Ai_{j} (I)), (Ai_{j} (II)) ou (Ai_{j} (III)) para o mesmo instante de tempo t. Isto em combinação com o uso de algum axioma do tipo (Aq_i) ou (As_j). O múltiplo uso de axiomas (Ai_{j} (I)), (Ai_{j} (II)) ou (Ai_{j} (III)), para o mesmo instante de tempo t, indica a execução simultânea de múltiplas instruções, o que pode levar (no tempo $t+1$) à multiplicidade de símbolos em algumas células da fita, ou à multiplicidade de estados ou posições da máquina. Por outro lado, os axiomas (Aq_i) e (As_j) estabelecem a unicidade dos elementos anteriormente mencionados, levando portanto a contradição.

As teorias $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ podem ser facilmente adaptadas para lidar com a idéia de que uma única instrução é escolhida quando a máquina alcança uma configuração ambígua, desta maneira obtendo teorias adequadas para representar computações em MTNDs. Porém, como já foi mencionado anteriormente, este não é nosso objetivo aqui. Nosso interesse é generalizar a noção de MT através de lógicas paraconsistentes, e é a isso que nos dedicamos na próxima seção.

2.3 Máquinas de Turing paraconsistentes

Existe uma grande diversidade de lógicas paraconsistentes, propostas a partir de diferentes perspectivas filosóficas, mas com a caraterística comum de se tratar de lógicas que suportam contradições evitando a trivialização dedutiva (i.e. a dedução de qualquer fórmula). Formalmente, uma lógica L é *paraconsistente* se existir uma teoria (ou conjunto de fórmulas) Γ, na linguagem de L, e fórmulas α e β tais que $\Gamma \vdash_L \alpha$, $\Gamma \vdash_L \neg\alpha$ e $\Gamma \nvdash_L \beta$ (nesse caso, o símbolo $\neg$ é chamado de *negação paraconsistene*).

Na definição das MTPs é possível, em princípio, usar qualquer lógica paraconsistente de primeira ordem. Na definição do primeiro modelo de MTPs vamos utilizar a lógica paraconsistente $LFI1^{*}$ (introduzida em [Cdd00]), devido ao fato de se tratar de uma lógica de primeira ordem e com uma semântica simples e bem estabelecida, além de permitir uma interpretação natural das conseqüências das teorias intrínsecas

às computações e dar a possibilidade de controlar a execução das instruções através de condições de 'inconsistência'.

$LFI1^*$ é a versão de primeira ordem da lógica $LFI1$, a qual é uma *lógica de inconsistência formal (LFI)* (cf. [CCM07]) que estende a lógica positiva clássica incluindo conectivos de consistência $\circ$ e inconsistência $\bullet$. Em $LFI1$ (e portanto em $LFI1^*$) as noções de inconsistência e contradição coincidem, através da equivalência $\bullet A \leftrightarrow (A \wedge \neg A)$, e o conectivo de consistência serve para recuperar o caráter explosivo das contradições: se $\Gamma \vdash_{LFI1} \alpha$ e $\Gamma \vdash_{LFI1} \neg\alpha$ então $\Gamma, \circ\alpha \vdash_{LFI1} \beta$, para toda fórmula β. Além disso, é importante ressaltar que $LFI1^*$ permite codificar todas as inferências válidas da lógica de primeira ordem clássica.

Passamos então a definir o primeiro modelo de MTPs e na Seção 2.3.2 analisamos algumas características referentes a seu 'poder' computacional.

2.3.1 Definição do modelo de MTPs

Para definir as MTPs, inicialmente substituímos a lógica subjacente às teorias $\Delta^{\star}_{LPC}(\mathcal{M}(\alpha))$ pela lógica $LFI1^*$, obtendo teorias $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ que permitem axiomatizar computações de MTNDs sem levar a trivialização. Interpretando estas teorias vamos definir em que consiste uma MTP.

Para as teorias $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ a demonstração de [BJ89] continua valendo, pelo fato de $LFI1^*$ ser uma extensão da lógica positiva clássica. Portanto, como descrito anteriormente, a aplicação de múltiplas instâncias de esquemas de axiomas do tipo Ai_{j} (I), (Ai_{j} (II)) ou (Ai_{j} (III)), para o mesmo instante de tempo t, denota a execução simultânea das respetivas instruções, o que pode levar a multiplicidade de símbolos em algumas células da fita, ou a multiplicidade de estados ou posições da máquina. Tal multiplicidade, em conjunção com o uso de axiomas do tipo (Aq_i) ou (As_j), necessariamente levam a contradições, as quais em $LFI1^*$ são identificadas com inconsistências. Portanto, as inconsistências nas teorias $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ estão relacionadas à multiplicidade (de estados, de posições da máquina ou de símbolos em uma célula da fita).

Tirando proveito dessa interpretação de inconsistência como ligada a situações de multiplicidade nas teorias $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$, e do fato destas teorias serem capazes de

suportar contradições, adicionaremos condições de inconsistência nos dois símbolos iniciais das instruções de forma a controlar sua execução. Assim, $q_i^\bullet$ indicará que a instrução será executada unicamente no caso em que a máquina esteja em múltiplos estados ou posições, enquanto $s_j^\bullet$ indicará que a instrução será executada unicamente no caso em que a célula da fita (onde a instrução vai ser executada) contenha múltiplos símbolos. Estas condições correspondem formalmente a colocar o conectivo $\bullet$, respectivamente, na frente do predicado Q_i ou S_j no antecedente dos axiomas que expressam instruções. Estas condições aparentemente inócuas permitem aproveitar o paralelismo das MTPs, como será mostrado adiante.

Note que os axiomas do tipo (Ai_j (I)), (Ai_j (II)) e (Ai_j (III)) não só expressam a ação das instruções, mas também especificam que os símbolos nas células da fita que não são modificadas pela instrução são preservados, i.e. continuam sendo os mesmos do instante de tempo anterior. Portanto, nas MTPs é necessário levar em conta que cada instrução é executada numa posição específica da fita, levando o conteúdo das células não modificadas pela instrução para o próximo instante de tempo; esta tarefa é realizada de maneira independente à execução das outras instruções.

Uma MTP é então definida como:

Definição 2.12. Uma *MTP* é uma MTND tal que:

- quando a máquina alcança uma configuração ambígua executa todas as possíveis instruções *simultaneamente*, o que pode produzir multiplicidade de símbolos em algumas células da fita, e multiplicidade de estados ou posições da máquina;
- cada instrução é executada nas posições em que a configuração atual se corresponde com os dois símbolos iniciais da instrução;
- cada execução de uma instrução preserva os símbolos nas células que não modificada, levando-os para o seguinte instante de tempo;
- podem ser adicionadas condições de *inconsistência* (ou *multiplicidade*) nos primeiros dois símbolos da instrução, para controlar que a instrução seja executada somente em situações de multiplicidade (como descrito acima);

- a máquina pára quando não há instruções a serem executadas; nesse caso, algumas células da fita podem conter múltiplos símbolos, portanto cada escolha desses símbolos representa um resultado da computação.

O seguinte exemplo mostra uma computação numa MTP:

Exemplo 2.1. Seja $\mathcal{M}$ uma MTP com instruções: $i_1 : q_1 00 q_2$, $i_2 : q_1 01 q_2$, $i_3 : q_2 0 R q_3$, $i_4 : q_2 1 R q_4$, $i_5 : q_3 \emptyset 0 q_5$, $i_6 : q_4 \emptyset 1 q_5$, $i_7 : q_5 00 q_6$, $i_8 : q_5 10 q_6$, $i_9 : q_5 0^\bullet * q_6$ e $i_{10} : q_6 * 1 q_6$. A Figura 2.1 esquematiza a computação de $\mathcal{M}$, começando com o dado de entrada $\alpha = 0$. As instruções a serem executadas em cada instante de tempo estão escritas entre parêntese.

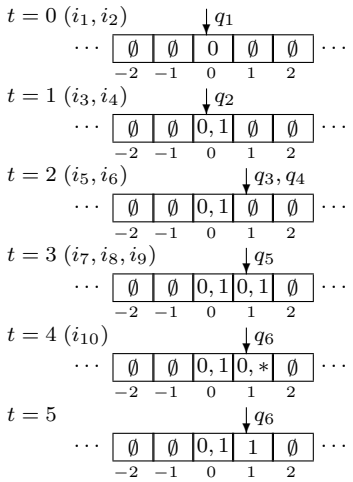

Figura 2.1: Exemplo de computação em uma MTP

As instruções de $\mathcal{M}$ podem ser entendidas da seguinte maneira:

- As instruções i_3 a i_6 correspondem a uma MTD $\mathcal{M}'$ que, começando no estado q_2, computa a função identidade para uma entrada de comprimento 1 (na linguagem $\{0, 1\}$), deixando o resultado na posição 1 da fita.

- As instruções i_1 e i_2 criam uma multiplicidade de símbolos, que representam todos os possíveis dados de entrada, permitindo executar $\mathcal{M}'$ em paralelo para todas as entradas.

- As instruções i_7 a i_{10} determinam se as computações em paralelo de $\mathcal{M}'$ produzem um único resultado (nesse caso, a função computada por $\mathcal{M}'$ é constante), escrevendo 0 na posição 1 da fita; ou múltiplos resultados (nesse caso, a função computada por $\mathcal{M}'$ não é constante), escrevendo 1 na posição 1 da fita.

Portanto, podemos pensar em $\mathcal{M}'$ como uma caixa preta (ou *oráculo*) que pode ser substituída por outra MTD, que compute uma função $f\colon \{0,1\} \to \{0,1\}$ diferente da identidade. Desta maneira, a MTP $\mathcal{M}$ determina se a função computada pelo oráculo é constante ou não.

Até que ponto podemos aproveitar o paralelismo provido pelas MTPs? É isso que vamos discutir a seguir.

2.3.2 Poder computacional das MTPs

Poderíamos ingenuamente pensar que seria fácil definir uma MTP que computasse o problema de *satisfatibilidade de fórmulas do cálculo proposicional clássico (SAT)*[7] em tempo polinomial: poderíamos considerar como oráculo uma MTD $\mathcal{M}'$ tal que, dada uma fórmula α do cálculo proposicional clássico e uma valoração para as variáveis em α, determine o valor de verdade de α para essa valoração em tempo polinomial (o que é relativamente fácil de definir). Acrescentando instruções que gerem multiplicidade de símbolos nas posições correspondentes à valoração (generalizando o que fazem as instruções i_3 a i_6 do exemplo anterior), poderíamos dessa maneira representar todas as possíveis valorações para que $\mathcal{M}'$ realize a valoração da fórmula em paralelo. E por último, acrescentando a instrução $i_{k+1} : q_f 0^\bullet 1 q_{f+1}$ (onde q_f é o estado final de $\mathcal{M}'$), seria possível eliminar a ambigüidade do resultado quando a fórmula não seja nem contradição nem tautologia.

Contudo, o paralelismo permitido por este primeiro modelo de MTPs tem propriedades peculiares que podem levar a 'anomalias' no processo de computação.

[7] O problema SAT consiste em definir um algoritmo que, dada uma fórmula do cálculo proposicional clássico como entrada, determine se essa fórmula é satisfatível ou não, i.e., se existe uma valoração que faz a fórmula tomar o valor verdadeiro. É bem conhecido que SAT é um problema **NP**-completo.

Por exemplo, considere uma MTD $\mathcal{M}'$ com instruções: $i_1 = q_1 00 q_2$, $i_2 = q_1 11 q_3$, $i_3 = q_2 0 R q_4$, $i_4 = q_3 1 R q_4$, $i_5 = q_2 1 R q_5$, $i_6 = q_4 \emptyset 1 q_4$ e $i_7 = q_5 \emptyset 0 q_5$. Se $\mathcal{M}'$ começa a computação com um único símbolo (0 ou 1) como dado de entrada, pode se verificar que $\mathcal{M}'$ computa a função constante 1 (deixando o resultado na posição 1 da fita). Porém, se $\mathcal{M}'$ é considerada como um oráculo que começa a computar com os símbolos 0 e 1 (ambos simultaneamente) como dados de entrada, produz então como saída 0 e 1, como se $\mathcal{M}'$ computasse uma função não constante. Este exemplo mostra que diferentes caminhos de computação do oráculo podem se misturar de maneira indiscriminada, gerando caminhos de computação que não são possíveis na MTD (considerada como oráculo) quando computa de maneira seqüencial, e produzindo resultados indesejados. Portanto somente MTDs que não produzam resultados anômalos (i.e., resultados que não podem ser obtidos se a MTD computar de maneira seqüencial) podem ser consideradas como oráculos. As MTDs com estas propriedades serão chamadas de *paralelizáveis*.

Conseqüentemente, para definir uma MTP que computasse SAT em tempo polinomial precisaríamos que a MTD $\mathcal{M}'$, que determina o valor de verdade da fórmula dada uma valoração, fosse paralelizável. Definir uma MTD com estas propriedades não é nada fácil, e talvez seja mesmo impossível: na verdade, essa tarefa é equivalente a demonstrar que $\mathbf{P} = \mathbf{NP}$ (o que seria uma conseqüência direta do Teorema 2.4, apresentado a seguir).

O paralelismo permitido por estas primeiras MTPs, embora possa ser usado para explicar algumas propriedades da computação quântica (veja o Capítulo 5), pode ser simulado em tempo polinomial por MTDs (Teorema 2.4), e portanto não representa nenhuma vantagem quanto à eficiência.

Teorema 2.4. *Toda MTP pode ser simulada em tempo polinomial por uma MTD.*

Demonstração. Seja $\mathcal{M}$ uma MTP com n estados e m símbolos de entrada/saída. A MTD $\mathcal{M}'$ para simular $\mathcal{M}$ é definida assim:

- Vamos supor que a fita de $\mathcal{M}'$ é dividida em $2n + m$ faixas (estas $2n + m$ faixas podem ser reduzidas a uma faixa mudando a linguagem de $\mathcal{M}'$, sem implicações quanto à eficiência).

- As faixas i $(1 \leq i \leq n)$, na posição p, serão usadas para representar os estados q_i em que $\mathcal{M}$ se encontra nessa posição, representando com 1 a ocorrência do estado e com 0 a não ocorrência.

- As faixas j $(n+1 \leq j \leq n+m)$, na posição p, serão usadas para representar os símbolos s_j presentes na célula de $\mathcal{M}$ nessa posição, representando com 1 a ocorrência do símbolo e com 0 a não ocorrência.

- As faixas $n+m+1$ a $2n+m$ serão usadas para calcular os estados que resultam da execução em paralelo das instruções em $\mathcal{M}$; o valor da faixa $n+m+i$, na posição p, indicará se o estado q_i ocorrera nessa posição no seguinte instante de tempo, representando com 1 a ocorrência e com 0 a não ocorrência.

- O símbolo \$ sera usado na faixa 1 de $\mathcal{M}'$ para delimitar a área na qual $\mathcal{M}$ está em algum estado.

O dado de entrada para $\mathcal{M}'$ é representado de acordo com as convenções anteriores, e os delimitadores \$ são colocados nas posições -1 e 1 da fita (antes e depois da posição 0, que é onde começa a computação de $\mathcal{M}$). Para simular um passo de computação de $\mathcal{M}$, $\mathcal{M}'$ deve percorrer quatro vezes a parte da fita delimitada pelos símbolos \$. No primeiro percurso (de esquerda a direita), $\mathcal{M}'$ simula a execução em paralelo em $\mathcal{M}$ das instruções do tipo (II) (onde a ação consiste em um movimento a direita): em cada posição p, $\mathcal{M}'$ escreve valores nas faixas $n+m+1$ a $2n+m$ de acordo com as instruções de $\mathcal{M}$ e o conteúdo das faixas 1 a $n+m$; além disso, desloca os delimitadores \$ se for necessário. O segundo percurso é similar ao anterior, mas no sentido contrário e simulando a execução de instruções do tipo (III) (onde a ação consiste em um movimento à esquerda). No terceiro percurso (de esquerda para a direita de novo), $\mathcal{M}'$ simula a execução em paralelo de instruções do tipo (I) (onde a ação consiste em modificar o símbolo da célula): em cada posição p, dependendo das instruções de $\mathcal{M}$ e do conteúdo das faixas 1 a $n+m$, $\mathcal{M}'$ escreve valores nas faixas $n+1$ a $n+m$ (correspondentes a símbolos escritos pelas instruções de $\mathcal{M}$) e nas faixas $n+m+1$ a $2n+m$ (correspondentes às mudanças de estados das instruções de escrita). Finalmente, no quarto percurso (de direita a esquerda), $\mathcal{M}'$ copia os

valores das faixas $n+m+1$ a $2n+m$ nas faixas 1 a n, e reinicia as faixas $n+m+1$ a $2n+m$ deixando-as com valores 0.

Vamos mostrar agora que a simulação de n passos de computação é realizada em tempo polinomial: depois de n passos de computação em $\mathcal{M}$, o estado mais à direita e o estado mais à esquerda podem estar separados no máximo por $2n$ células; portanto, essa é a separação máxima entre os delimitadores \$ em $\mathcal{M}'$ na simulação do n-ésimo passo de computação. Como em cada percurso que $\mathcal{M}'$ realiza na simulação de um passo de computação a cabeça da máquina deve se deslocar entre os delimitadores \$, além de realizar operações de escrita em cada posição, cada percurso na simulação do n-ésimo passo toma no máximo $4n$ passos (ignorando os passos devidos a possíveis deslocamentos dos delimitadores \$, que são um número constante de passos). Dessa forma a simulação da n-ésima instrução exige no máximo $16n$ passos. Portanto, para simular os n primeiros passos de computação, $\mathcal{M}'$ vai demorar no máximo $n(16n)$ passos de computação, i.e., $O(n^2)$. □

Note que cada escolha de elementos de uma MTP (em qualquer instante de tempo) representa uma configuração de uma MT clássica, portanto, podemos pensar na configuração de uma MTP como sendo um conjunto de configurações (clássicas). Assim, sob o suposto de que no final da computação de uma MTP deve ser escolhida aleatoriamente uma única configuração das múltiplas possíveis, de maneira a obter um único resultado, e de que cada configuração é escolhida com a mesma probabilidade, podemos adequar a definição de aceitação de cadeias de símbolos (Definição 2.5) para MTPs da seguinte maneira:

Definição 2.13. Seja $\mathcal{M}$ uma MTP com alfabeto de entrada/saída Σ e seja $\alpha \in \Sigma^*$. $\mathcal{M}$ *aceita* (resp. *rejeita*) α se a computação de $\mathcal{M}(\alpha)$ pára e pelo menos $\frac{2}{3}$ das configurações finais[8] tem como estado q_s (resp. q_n), i.e. se a probabilidade de obter uma configuração final com estado q_s (resp. q_n) é maior ou igual a $\frac{2}{3}$.

Nas MTPs o tempo de computação não pode, porém, ser definido como o número de instruções que a máquina executa; pois num mesmo instante de tempo múltiplas

[8]Por 'configurações finais' nos referimos às configurações no instante de tempo em que a computação termina.

instruções podem ser executadas simultaneamente. O tempo de computação numa MTP deve ser definido como o *número de passos* de computação, onde um passo de computação corresponde à execução (simultânea) de todas as instruções que podem ser efetivadas na configuração atual da máquina. Podemos então definir uma nova classe de complexidade algorítmica, a qual chamaremos de **BParP**, que corresponde à classe de linguagens que podem ser decididas por MTPs em tempo polinomial com erro limitado.

$$\begin{aligned}\mathbf{BParP} =&\{L \mid \text{existe uma MTP que decide } L \text{ e termina em tempo}\\ &f(c) = c^k, \text{ para alguma constante } k \in \mathbb{N}\}.\end{aligned}$$

É claro que $\mathbf{P} \subseteq \mathbf{BParP}$ (pois as MTD são casos especiais de MTPs). A inclusão no outro sentido é conseqüência do Teorema 2.4:

Corolário 2.1. ***BParP*** $\subseteq$ ***P***.

Demonstração. Basta acrescentar instruções na MTD $\mathcal{M}'$, definida na demonstração do Teorema 2.4, para determinar se o número de símbolos 1 na faixa correspondente ao estado q_s é pelo menos $\frac{2}{3}$ dos que tem na faixa correspondente ao estado q_n; nesse caso $\mathcal{M}'$ deve passar a seu estado de aceitação, caso contrário deve passar a seu estado de rejeição. □

Portanto, $\mathbf{BParP} = \mathbf{P}$. O que mostra que este primeiro modelo de MTPs introduzido não acrescenta nada quanto à complexidade algorítmica.

2.4 Máquinas de Turing paraconsistentes não-separáveis

No modelo de MTPs definido na seção anterior diferentes caminhos de computação podem se misturar indiscriminadamente, não permitindo aproveitar de maneira adequada o paralelismo intrínseco ao modelo. Isto ocorre devido a que, em situações de multiplicidade, todas as possíveis combinações de estados e símbolos nas células da fita são considerados na execução das instruções. Esta característica é conseqüência direta de algumas propriedades da lógica $LFI1^*$, usada na definição do modelo: a conjunção em $LFI1^*$ valida a *regra de separação* (i.e., se $\vdash_{LFI1^*} A \wedge B$ então $\vdash_{LFI1^*} A$

e $\vdash_{LFI1^*} B$) e a *regra de adjunção* (i.e., se $\vdash_{LFI1^*} A$ e $\vdash_{LFI1^*} B$ então $\vdash_{LFI1^*} A \wedge B$). Assim, se $\Delta^{\star}_{LFI1^*}(\mathcal{M}(n)) \vdash Q_i(\overline{t}, \overline{x}) \wedge S_j(\overline{t}, \overline{x})$ e $\Delta^{\star}_{LFI1^*}(\mathcal{M}(n)) \vdash Q_k(\overline{t}, \overline{x}) \wedge S_l(\overline{t}, \overline{x})$, é também possível deduzir $\Delta^{\star}_{LFI1^*}(\mathcal{M}(n)) \vdash Q_i(\overline{t}, \overline{x}) \wedge S_l(\overline{t}, \overline{x})$ e $\Delta^{\star}_{LFI1^*}(\mathcal{M}(n)) \vdash Q_k(\overline{t}, \overline{x}) \wedge S_j(\overline{t}, \overline{x})$. Conseqüentemente, se se quiser definir um modelo de MTPs onde os diferentes caminhos de computação não se misturem totalmente, deve-se considerar uma lógica paraconsistente onde não seja válida a regra de separação ou a regra de adjunção (ou ambas).

Existem lógicas paraconsistentes com conjunções não-adjuntivas,[9] mas não existem lógicas paraconsistentes onde a regra de separação não seja válida. Seria interessante definir um novo modelo de MTPs usando a extensão a primeira ordem de $D2$, porém novos axiomas teriam que ser propostos para indicar quando diferentes elementos no processo de computação podem ser considerados em conjunto. Preferimos, então, definir uma lógica paraconsistente na qual não seja válida a regra de separação, de forma a permitir interpretar os diferentes elementos da máquina correspondentes a um caminho de computação como estando *inter-relacionados*, o que permitirá pensar em caminhos de computação independentes.

2.4.1 Definição do modelo de MTPNSs

Seguindo as idéias em [Béz02] (veja também [Béz05], [Mar05a] e [Mar05b]) uma negação paraconsistente $\neg_{\diamond}$ é definida dentro da lógica modal $S5$ por $\neg_{\diamond} A \stackrel{\text{def}}{=} \diamond \neg A$ (onde $\neg$ é a negação clássica e $\diamond$ é o operador de possibilidade).[10] Além disso, definimos aqui uma *conjunção não-separável* $\wedge_{\diamond}$ em $S5$ por $A \wedge_{\diamond} B \stackrel{\text{def}}{=} \diamond(A \wedge B)$ (onde $\wedge$ é a conjunção clássica), que servirá para representar propriedades que podem ser

[9] A lógica paraconsistente não adjuntiva mais conhecida é a *lógica discursiva* $D2$, introduzida por Stanisław Jaśkowski em [Jaś48] e [Jaś49], a qual conta com extensões a primeira ordem, cf. [dD95].

[10] Algumas propriedades do operador $\neg_{\diamond}$ são apresentadas em [Béz02] e [Béz05].

conjuntamente válidas. Algumas propriedades desta conjunção são as seguintes:

$$\vdash_{S5} A \wedge_\diamond B \text{ não implica } \vdash_{S5} A \text{ nem } \vdash_{S5} B, \tag{$\wedge_\diamond 1$}$$

$$\vdash_{S5} A \text{ e } \vdash_{S5} B \text{ implica } \vdash_{S5} A \wedge_\diamond B, \tag{$\wedge_\diamond 2$}$$

$$\nvdash_{S5} (A \wedge_\diamond (B \wedge_\diamond C)) \leftrightarrow ((A \wedge_\diamond B) \wedge_\diamond C), \tag{$\wedge_\diamond 3$}$$

$$\vdash_{S5} (A \wedge_\diamond B) \leftrightarrow (B \wedge_\diamond A), \tag{$\wedge_\diamond 4$}$$

$$\nvdash_{S5} ((A \wedge_\diamond B) \wedge (C \wedge_\diamond D)) \rightarrow ((A \wedge_\diamond D) \vee (C \wedge_\diamond B))) \tag{$\wedge_\diamond 5$}$$

$$\vdash_{S5} (A_1 \wedge_\diamond (A_2 \wedge \ldots \wedge A_n)) \leftrightarrow \diamond(A_1 \wedge \ldots \wedge A_n). \tag{$\wedge_\diamond 6$}$$

A propriedade ($\wedge_\diamond 1$) reflete o caráter não separável de $\wedge_\diamond$, enquanto ($\wedge_\diamond 2$) mostra que $\wedge_\diamond$ valida a regra de adjunção e ($\wedge_\diamond 3$) indica a não associatividade de $\wedge_\diamond$. ($\wedge_\diamond 4$) mostra que $\wedge_\diamond$ é comutativa, ($\wedge_\diamond 5$) é uma conseqüência de ($\wedge_\diamond 1$) que será relevante para o inter-relacionamento de elementos, e ($\wedge_\diamond 6$) é uma aplicação simples da definição de $\wedge_\diamond$ que será útil adiante.

Uma lógica paraconsistente 'não-separável', que chamaremos $PNS5$, é obtida a partir da lógica modal $S5$ através da função de tradução $*$: $ForPNS5 \rightarrow ForS5$ definida indutivamente por:[11]

$$A^* = A \text{ se } A \text{ é atômica},$$
$$(\neg_\diamond A)^* = \diamond\neg(A)^*,$$
$$(A \wedge_\diamond B)^* = \diamond(A^* \wedge B^*),$$
$$(A\#B)^* = A^*\#B^* \text{ for } \# \in \{\vee, \rightarrow\};$$

A relação de conseqüência de $PNS5$ é definida com base na tradução $*$ por:

$$\Gamma \vdash_{PNS5} A \text{ se e somente se } \Gamma^* \vdash_{S5} A^*,$$

onde Γ representa um subconjunto de $ForPNS5$ e $\Gamma^* = \{B^* | B \in \Gamma\}$. Dessa maneira a lógica $PNS5$ fica completamente definida, e é uma sublógica de $S5$.

No espirito das LFIs (veja [CCM07]), podemos definir um conectivo $\bullet$ de 'inconsistência' em $PNS5$ por $\bullet A \stackrel{\text{def}}{=} A \wedge_\diamond \neg_\diamond A$ (o que é equivalente a $\diamond A \wedge \diamond\neg A$

[11] $ForPNS5$ é o conjunto de fórmulas proposicionais definida (da maneira usual) sobre a assinatura $\sigma = \{\neg_\diamond, \wedge_\diamond, \vee, \rightarrow\}$ e $ForS5$ é o conjunto de fórmulas de $S5$.

em $S5$), um conectivo $\circ$ de 'consistência' por $\circ A \stackrel{\text{def}}{=} \neg_\diamond \bullet A$ (o que é equivalente a $\Box\neg A \vee \Box A$ em $S5$), uma negação clássica $\neg$ por $\neg A \stackrel{\text{def}}{=} \neg_\diamond A \wedge_\diamond \circ A$ (o que é equivalente a $\diamond\neg A \wedge (\Box\neg A \vee \Box A)$ em $S5$, de onde se deriva $\neg A$) e uma conjunção clássica por $A \wedge B \stackrel{\text{def}}{=} (A \wedge_\diamond B) \wedge_\diamond (\circ A \wedge_\diamond \circ B)$ (o que é equivalente a $\diamond(A \wedge B) \wedge (\Box(A \wedge B) \vee \Box(A \wedge \neg B) \vee \Box(\neg A \wedge B) \vee \Box(\neg A \wedge \neg B))$ em $S5$, de onde se deriva $A \wedge B$). Com estas definições, os 'princípios de explosão' $(A \wedge \neg_\diamond A \wedge \circ A) \rightarrow B$, $(A \wedge_\diamond (\neg_\diamond A \wedge_\diamond \circ A)) \rightarrow B$, $((A \wedge_\diamond \neg_\diamond A) \wedge_\diamond \circ A) \rightarrow B$ e $((A \wedge_\diamond \circ_\diamond A) \wedge_\diamond \neg A) \rightarrow B$ são teoremas de $PNS5$. Tendo em conta tais características $PNS5$ pode legitimamente ser considerada como uma LFI.

Para poder usar $PNS5$ (em lugar de $LFI1^*$) na definição de um segundo modelo de MTPs é necessário primeiro estender esta lógica à primeira ordem (com igualdade). Isso pode ser feito considerando $S5Q^=$ (a versão de primeira ordem com igualdade de $S5$) em lugar de $S5$ na definição da lógica, adaptando a função de tradução $*$ para incluir os quantificadores e a igualdade. Contudo, para simplificar o trabalho, vamos considerar $S5Q^=$ em vez de definir a nova lógica, olhando os conectivos $\neg_\diamond, \wedge_\diamond, \bullet$ e $\circ$ como definições. A definição de $PNS5$, tanto quanto à explicação de como pode ser obtida sua versão de primeira ordem, tem a intenção de mostrar que o fragmento de $S5Q^=$ que vamos usar corresponde a uma lógica paraconsistente, com uma conjunção não-separável.

Para definir o novo modelo de MTPs, o qual chamamos de *máquina de Turing paraconsistente não-separável (MTPNS)*, vamos substituir a lógica subjacente das teorias $\Delta^\star_{LPC}(\mathcal{M}(\alpha))$ pela lógica $S5Q^=$, e vamos interpretar essas teorias através da semântica de mundos possíveis. Para isso, é necessário primeiro indicar o tipo de negação ($\neg$ ou $\neg_\diamond$) e de conjunção ($\wedge$ ou $\wedge_\diamond$) que vamos considerar em cada ocorrência desses operadores nos axiomas de $\Delta^\star_{S5Q^=}(\mathcal{M}(\alpha))$. Além disso, vamos assumir a propriedade de associatividade à direita nos lugares em que o operador $\wedge_\diamond$ aparecer, i.e., $A \wedge_\diamond B\#C$, onde $\#$ é $\wedge$ ou $\wedge_\diamond$, será equivalente a $A \wedge_\diamond (B\#C)$ (isto é necessário devido à não-associatividade do conectivo $\wedge_\diamond$). A interpretação dos conectivos nos axiomas de $\Delta^\star_{S5Q^=}(\mathcal{M}(\alpha))$ será a seguinte:

1. Nos axiomas (A1) a (A5) as negações e conjunções são as clássicas.

2. A negação nos axiomas do tipo (Ai_{j} (I)) (em $y \neq x$) é a negação clássica. Nos axiomas do tipo (Ai_{j} (I)) a (Ai_{j} (III)) a conjunção no antecedente é $\wedge_\diamond$ e, levando em conta a propriedade ($\wedge_\diamond 6$), somente a primeira conjunção do conseqüente é $\wedge_\diamond$, as outras são a conjunção clássica. Além disso, na parte que indica a preservação dos símbolos não modificados pela instrução (i.e., na conjunção $\bigwedge_{i=1}^{m}\big(S_i(t,y) \rightarrow S_i(t',y)\big)$) é preciso mudar o antecedente por $S_i(t,y) \wedge_\diamond (Q_i(t,x) \wedge S_j(t,x))$, para que sejam preservados somente os símbolos relacionados (ou que podem ser conjuntamente válidos) com a configuração que está causando a execução da instrução.

3. Nos axiomas (At_0) e (At_-) as negações e conjunções são as clássicas.

4. No axioma (At_+) somente a conjunção no antecedente é $\wedge_\diamond$, os outros conectivos são os clássicos.

5. Nos axiomas (Aq_i) e (As_j) todas as conjunções são clássicas, mas as negações são $\neg_\diamond$ (exceto em $y \neq x$). Além disso, é necessário adicionar o conectivo $\diamond$ antes dos predicados Q_i e S_i no antecedente dos axiomas (para que este antecedente possa ser satisfeito nos instantes de tempo da computação).

Para interpretar as teorias $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))$, de maneira a definir o modelo de MTPNSs, vamos partir da interpretação Kripkeana[12] dessa teorias e vamos pensar nos mundos possíveis como representando configurações da máquina, de acordo com a seguinte definição:

Definição 2.14. Seja w um mundo possível na interpretação Kripkeana de $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))$. Se $Q_i(\overline{t},\overline{x}), \ldots, S_{j_{-1}}(\overline{t},\overline{-1}), S_{j_0}(\overline{t},\overline{0}), S_{j_1}(\overline{t},\overline{1}), \ldots$ são predicados válidos em w, então w *representa* a configuração da MT $\mathcal{M}$, no instante de tempo $\overline{t}$, dada pela interpretação intensional I (descrita acima) dos respetivos predicados.

[12]Por 'interpretação Kripkeana' nos referimos à semântica relacional (também chamada de 'semântica de Kripke' ou 'semântica de mundos possíveis') introduzida por Saul Kripke para demonstrar a completude e decidibilidade de sistemas modais normais. Para uma descrição desse tipo de semânticas veja [CP08].

De acordo com as considerações anteriores, a interpretação das teorias $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))$ indica a computação em paralelo de todos os possíveis caminhos de computação de $\mathcal{M}$, para o dado de entrada α, pois:

1. Pelo axioma (At_0) existe um mundo w_0 representando a configuração inicial de $\mathcal{M}(\alpha)$.

2. Pelos axiomas do tipo ($\mathrm{A}i_{\mathrm{j}}$ (I)) a ($\mathrm{A}i_{\mathrm{j}}$ (III)), se o mundo w_t representa uma configuração não final (na qual alguma instrução i_j pode ser executada) de $\mathcal{M}(\alpha)$, no instante de tempo t, então existe um mundo $w_{t+1,j}$ que representa a configuração de $\mathcal{M}(\alpha)$ depois da execução de i_j.

Note que, nos casos em que existem diferentes mundos representado configurações da máquina para o mesmo instante de tempo, é possível que uma configuração indique que $\mathcal{M}(\alpha)$ está no estado q_i, lendo o símbolo s_j, na posição x da fita, enquanto uma outra configuração indica que $\mathcal{M}(\alpha)$ está lendo o símbolo s_l, na posição x da fita. Se não existir uma configuração (para o mesmo instante de tempo) na qual o estado na posição x seja q_i, e o símbolo nessa célula seja s_l, dizemos que q_i está *inter-relacionado* com s_j. De maneira análoga se define o inter-relacionamento para os diferentes elementos da máquina.

De maneira similar ao modelo de MTPs anterior, para tirar proveito do paralelismo intrínseco ao modelo, serão introduzidas condições de inconsistência nas instruções das MTPNSs para controlar sua execução. Nesse caso, pela definição do conectivo de inconsistência em $S5Q^{=}$ e sua interpretação em mundos possíveis, a condição $q_i^{\bullet}$ indicará que a instrução será executada somente se duas configurações simultâneas (i.e., para o mesmo instante de tempo) da máquina diferem em seu estado ou em sua posição atual. Por outro lado, a condição $s_j^{\bullet}$ indicará que a instrução será executada somente se duas configurações simultâneas diferem no símbolo da célula onde será executada a instrução.

Uma MTPNS é então definida como:

Definição 2.15. Uma *MTPNS* é uma MTND tal que:

- quando alcança uma configuração ambígua, com n possíveis instruções a ser

executadas, a máquina se divide em n cópias, executando uma instrução diferente em cada cópia;

- *condições de inconsistência* podem ser adicionadas nos dois primeiros símbolos das instruções, permitindo controlar (como indicado acima) as configurações nas quais a instrução deve ser executada;

- a máquina pára quando não há instruções a ser executadas; nesse caso, a máquina pode estar em múltiplas *configurações simultâneas*, cada uma das quais representa um resultado da computação.

Note que uma MTPNS executa de maneira paralela todos os possíveis caminhos de computação de uma MTND, sem misturar caminhos de computação como no modelo anterior. Além disso, as MTPNSs permitem controlar a execução das instruções levando em consideração diferentes configurações simultâneas da máquina, o que possibilita aproveitar o paralelismo do modelo. Mas até que ponto podemos aproveitar tal paralelismo?

2.4.2 Poder computacional das MTPNSs

Nas MTPNSs, de maneira similar ao que acontece com as MTPs anteriormente definidas, os múltiplos resultados obtidos numa computação podem ser considerados como resultados possíveis, dos quais um deles deve ser selecionado de maneira aleatória. Portanto, as noções de aceitação e rejeição de uma cadeia de símbolos neste modelo de máquinas podem ser definida da mesma maneira a como foi feito para as MTPs (Definição 2.13), e o tempo de execução pode também ser definido de maneira similar. Podemos então definir uma nova classe de complexidade algorítmica, a qual chamaremos por **BPNSP**, que corresponde à classe de linguagens que podem ser decididas por MTPNSs em tempo polinomial com erro limitado.

$$\begin{aligned}\mathbf{BPNSP} = \{L \mid &\text{ existe uma MTPNS que decide } L \text{ e termina em tempo} \\ &f(c) = c^k, \text{ para alguma constante } k \in \mathbb{N}\}.\end{aligned}$$

É fácil demonstrar a seguinte inclusão:

Teorema 2.5. ***NP*** $\subseteq$ ***BPNSP***

Demonstração. Seja $\mathcal{M}$ uma MTND, com linguagem de entrada/saida Σ, que aceita a linguagem L. A MTPNS $\mathcal{M}'$ definida acrescentando a $\mathcal{M}$ uma instrução i_{k+j} : $q_n^{\bullet} s_j s_j q_s$ por cada símbolo $s_j \in \Sigma$ aceita L com probabilidade de erro 0 (i.e. aceitando (resp. rejeitando) com probabilidade 1 as cadeias aceitas (resp. rejeitas) por $\mathcal{M}$). □

O Teorema 2.5 mostra as potencialidades quanto à eficiência das MTPNSs. Levando em consideração os teoremas 2.5 e 2.4, e a crença comum de que $\mathbf{P} \neq \mathbf{NP}$, os modelos de MTPs e MTPNSs resultam ser diferentes quanto à eficiência, embora ambos modelos tenham sido definidos através do mesmo método mas usando lógicas diferentes. Esses resultados servem como evidência para mostrar que a complexidade algorítmica é relativa à lógica. O modelo de MTPNSs também mostra como o paralelismo (sem limites) pode ser expresso através de conceitos lógicos, além de mostrar a relevância que tem a possibilidade de estabelecer relações entre os diferentes elementos de uma computação em paralelo (sem essas relações o paralelismo pode ser simulado eficientemente através de modelos seqüenciais, como é mostrado pelo Teorema 2.4).

Não parece muito natural considerar o modelo de MTPNSs como 'razoável', no sentido de poder ser fisicamente construível. Mesmo assim, como mencionado no Capítulo 1, as relações deste modelo com modelos de computação quântica (que serão apresentadas no Capítulo 5) permitirão interpretar algumas das características da computação quântica desde uma perspectiva lógica. No sentido contrário, algumas propriedades quânticas permitirão pensar num possível caminho para a construção efetiva de modelos de computação definidos através de lógicas paraconsistentes.

Na interpretação das teorias paraconsistentes de computação apresentadas neste capítulo as inconsistências são interpretadas em termos de multiplicidade (de estados, de posições da máquina ou de símbolos em células da fita). Por outro lado, a *semântica de traduções possíveis* introduzida por Walter Carnielli em [Car90] (estudadas por João Marcos em [Mar99]) permitem interpretar lógicas paraconsistentes, de maneira adequada, traduzindo fórmulas em múltiplos 'cenários lógicos'. Existe, sem dúvida, uma admirável analogia entre a semântica de traduções possíveis e a

maneira como foram aqui interpretadas as teorias paraconsistentes de computação, a qual sugere uma estreita relação entre os conceitos de inconsistência e multiplicidade, que está em concordância como a interpretação de múltiplos mundos da mecânica quântica. No Capítulo 6 acrescentamos outros comentários a respeito desta relação.

Capítulo 3

Circuitos paraconsistentes

No capítulo anterior foram definidos dois modelos de computação paraconsistente, generalizando o conceito de máquina de Turing, e foram investigadas algumas características desses modelos com respeito à complexidade algorítmica, mostrando que de certa maneira a eficiência dos processos de computação pode ser vista como relativa à lógica. Neste capítulo propomos um caminho diferente para definir modelos de computação paraconsistente e para estudar questões de complexidade algorítmica através de propriedades dos sistemas lógicos. O caminho aqui proposto consiste na generalização do modelo de *circuitos booleanos* para permitir definir circuitos baseados em lógicas não-clássicas.

Um circuito booleano consiste, basicamente, numa coleção finita de *variáveis de entrada* e *portas lógicas* conectadas de maneira direcionada e acíclica, onde as variáveis de entrada tomam valores em $\{0, 1\}$ (0 representando 'falso' e 1 representando 'verdadeiro') e cada porta calcula uma operação booleana (usualmente AND, OR ou NOT). Os circuitos booleanos podem ser vistos como computando funções da forma f: $\{0,1\}^n \to \{0,1\}^m$. Uma computação particular é realizada estabelecendo os valores das variáveis de entrada, calculando nível por nível os valores de saída das portas (de acordo com as tabelas de verdade dos conectivos lógicos) e considerando as saídas das portas finais (não conectadas a outras portas), em alguma ordem preestabelecida, como a saída do circuito.

É claro que o modelo de circuitos booleanos é baseado na *lógica proposicional*

clássica (CP), pois as portas funcionam de acordo com a definição funcional dos conectivos dessa lógica. Para definir circuitos lógicos para uma lógica L é necessário definir o conjunto de valores permitidos como entradas e saídas das portas e as operações que as portas realizam, de acordo com alguma semântica para L.

Uma generalização obvia dos circuitos booleanos, para lógicas finitamente valoradas, consiste em considerar os finitos valores de verdade como os valores de entrada/saída e definir as operações das portas de acordo com as definições funcionais dos respectivos conectivos. Desta maneira, os circuitos baseados em lógicas multi-valoradas, no conjunto de valores de verdade A, computariam funções da forma $f\colon A^n \to A^m$. Contudo, a generalização para lógicas não caraterizáveis por matrizes finitas (como é o caso de varias lógicas paraconsistentes) não é tão trivial.

A generalização de circuitos booleanos aqui proposta é alcançada por meio do *cálculo de anéis de polinômios (CAP)* (que será descrito na Seção 3.2), o qual permite definir as operações das portas através de operações polinomiais, com algumas características interessantes que permitirão definir circuitos para uma ampla variedade de lógicas não-clássicas, incluído algumas não caracterizáveis por matrizes finitas (Seção 3.3).

Como casos particulares são apresentados os modelos de *circuitos paraconsistentes mbC* (Seção 3.3.1) e de *circuitos modais* $S5$ (Seção 3.3.2), mostrando algumas características referentes à eficiência destes modelos. Também é definida uma subclasse de circuitos modais $S5$ (através da definição de uma negação paraconsistente, como foi feito no Capítulo 2, Seção 2.4), nessa classe os circuitos podem ser considerados como sendo paraconsistentes, conservando boas características quanto à eficiência.

Antes de introduzir a generalização do modelo de circuitos booleanos para lógicas não-clássicas apresentamos uma definição formal do modelo clássico, além de algumas definições referentes à complexidade algorítmica.

3.1 Definições preliminares

Formalmente um circuito booleano pode ser definido da seguinte maneira:

Definição 3.1. Um *circuito booleano* é um grafo direcionado acíclico $C = (V, E)$, onde V é o conjunto de *nós* e E é o conjunto de *arestas*. Os nós aos quais não chega nenhuma aresta são chamados *entradas* do circuito e são denotados por variáveis (x_1, x_2, ..., x_n) ou por valores constantes em $\{0, 1\}$; os outros nós são chamados *portas lógicas*, e são associados a conectivos em $\{\neg, \wedge, \vee\}$.[1] As portas lógicas associadas ao conectivo $\neg$ tem uma única aresta de entrada e as portas lógicas associadas aos conectivos $\wedge$ ou $\vee$ tem duas arestas de entrada. As arestas sem nós de saída são chamadas *saídas* do circuito.

Um circuito booleano computa funções da forma $f\colon \{0,1\}^n \to \{0,1\}^m$ da seguinte maneira:

Definição 3.2. Seja C um circuito booleano com n variáveis de entrada e m portas de saída e seja $x \in \{0,1\}^n$ um valor de entrada. C *calcula* o valor de saída atribuindo valores a seus nós de maneira indutiva:

- o nó correspondente à variável de entrada x_i toma como valor o i-ésimo dígito de x; e
- o nó correspondente a uma porta lógica, com arestas de entrada vindo de nós cujos valores já foram atribuídos, toma o valor obtido de aplicar (aos valores de suas entradas) a função correspondente ao respetivo conectivo lógico.

Os valores atribuídos aos nós que são saídas do circuito, ordenados de alguma maneira preestabelecida, representam o valor de *saída* do circuito (denotado por $C(x)$).

Como já foi mencionado no final do Capítulo 1, através de um circuito lógico somente é possível computar uma função da forma $f\colon \{0,1\}^n \to \{0,1\}^m$, onde o tamanho do dado de entrada e do dado de saída (n e m respectivamente) são fixos. Para computar funções onde o dado de entrada pode ser de qualquer tamanho é preciso definir famílias de circuitos. Uma *família de circuitos* é uma seqüência infinita $\{C_n\}_{n\in\mathbb{N}}$ onde C_n é o circuito que computa as entradas de tamanho n. Estamos aqui

[1] De maneira mais geral, os circuitos booleanos podem ser definidos como relativos a uma *base* (ou conjunto de conectivos), sempre que a base seja um conjunto completo de conectivos para CP. O conjunto de conectivos mais comummente usado é $\{\neg, \wedge, \vee\}$.

interessados na decisão de linguagens através de circuitos lógicos, portanto o tamanho do dado de saída será sempre $m = 1$.

O modelo de famílias de circuitos lógicos, como definido acima, é considerado um modelo de computação *não-uniforme*, pelo fato de definir um algoritmo (ou circuito) diferente para cada comprimento do dado de entrada. Com este modelo podem ser computadas funções não-computáveis por MTs: considere, por exemplo, a família de circuitos booleanos $\{C_n\}_{n\in\mathbb{N}}$ tal que C_n é um circuito que computa a função constante 1 se n corresponde ao código de uma MT que não pára para algum dado de entrada (C_n pode consistir na negação da primeira variável de entrada mais a disjunção desse valor com todas as variáveis de entrada), caso contrário C_n é um circuito que computa a função constante 0 (C_n pode consistir na negação da primeira variável de entrada mais a conjunção desse valor com todas as variáveis de entrada). A família de circuitos $\{C_n\}_{n\in\mathbb{N}}$, definida dessa maneira, decide a linguagem $L = \{x \in \{0,1\}^* \mid$ o comprimento de x corresponde ao código de uma MT que não pára para algum dado de entrada$\}$, a qual é uma linguagem não-decidível através de MTs.

A complexidade algorítmica da computação de funções através de famílias de circuitos booleanos é medida, principalmente, com base em dois parâmetros: o *tamanho* (i.e. o número de portas) e a *profundidade* (i.e. o tamanho do maior caminho entre variáveis de entrada e portas de saídas) dos circuitos em relação ao tamanho do valor de entrada. Existem resultados que estabelecem relações entre o tamanho dos circuitos, nas famílias de circuitos booleanos, e o tempo de computação em modelos de máquinas seqüenciais, como é o caso do modelo clássico de MT (alguns desses resultados serão mencionados adiante). Por outro lado, existem também resultados que estabelecem relações entre a profundidade dos circuitos e o tempo de computação de modelos de máquinas em paralelo (cf. por exemplo [CK02, Seção 6.2.5]).

Antes de definir classes de complexidade algorítmica para famílias de circuitos booleanos é preciso definir o que significa aceitar/rejeitar valores de entrada e decidir linguagens através deste modelo de computação.

Definição 3.3. Seja C_n um circuito booleano com n variáveis de entrada, C_n *aceita* (resp. *rejeita*) um valor de entrada $x \in \{0,1\}^n$ se $C_n(x) = 1$ (resp. $C_n(x) = 0$).

Definição 3.4. Seja $\{C_n\}_{n\in\mathbb{N}}$ uma família de circuitos booleanos, a linguagem $L \subseteq \{0,1\}^*$ é *decidida* por $\{C_n\}_{n\in\mathbb{N}}$ se, para todo $x \in \{0,1\}^*$, tem-se que: se $x \in L$ então C_n aceita x, caso contrário C_n rejeita x.

Agora podemos definir a classe de complexidade algorítmica $\mathbf{P}_{/\text{poly}}$, de linguagens decidíveis por famílias de circuitos booleanos de tamanho polinomial:

$$\begin{aligned}\mathbf{P}_{/\text{poly}} = \{&L \subseteq \{0,1\}^* \mid \text{existe uma família de circuitos booleanos que}\\ &\text{decide } L \text{ com circuitos de tamanho no máximo } T(C_n) = n^k,\\ &\text{para alguma constante } k \in \mathbb{N}\}.\end{aligned}$$

Em [AB09] é demonstrado que $\mathbf{P} \subseteq \mathbf{P}_{/\text{poly}}$. Tal inclusão é própria, pelo fato de existir linguagens não decidíveis em $\mathbf{P}_{/\text{poly}}$ (que claramente não pertencem a $\mathbf{P}$). Em [AB09] é também demonstrado que $\mathbf{BPP} \subseteq \mathbf{P}_{/\text{poly}}$.

Usualmente é colocada a restrição de *uniformidade* na definição das famílias de circuitos booleanos, para evitar a computação de funções não computáveis por máquinas de Turing, indicando como isto que a família deve poder ser 'gerada' por uma MT (i.e. que existe uma MT tal que começando com entrada 1^n produz como saída uma descrição do circuito C_n da família). Além disso, para impedir ocultar questões de eficiência na definição da família, é também exigido que a MT que gera os circuitos compute em tempo polinomial (em alguns textos esta condição é chamada de P-*uniformidade*). Adicionando esta condição segue-se que $\mathbf{P} = \mathbf{P}_{/\text{poly}}$, mas não se segue (de maneira direta) que $\mathbf{BPP} = \mathbf{P}_{/\text{poly}}$ (a demonstração de que $\mathbf{BPP} \subseteq \mathbf{P}_{/\text{poly}}$ mostra que é possível definir famílias de circuitos booleanos de tamanho polinomial para todas as linguagens em $\mathbf{BPP}$, mas não indica como gerar tal família através de um algoritmo eficiente).[2]

O modelo de circuitos lógicos é generalizado introduzindo não-determinismo e probabilidade, de maneira análoga a como foi feito com as MTs. Assim, um *circuito booleano não-determinístico* é definido como um circuito no qual a entrada é dividida em dois valores x e y, sendo x o valor de entrada a ser computado e sendo y um

[2]Se se demonstrar que $\mathbf{BPP} \subseteq \mathbf{P}_{/\text{poly}}$ ainda com a condição de P-uniformidade teria-se que $\mathbf{P} = \mathbf{BPP}$, o que não foi demonstrado até agora e pode ainda ser falso.

valor obtido de maneira aleatória. Um circuito booleano não-determinístico C *aceita* o valor de entrada x se existir um valor y tal que $C(x, y) = 1$, e *rejeita* x em caso contrário. Conseqüentemente, um *circuito booleano probabilístico* é definido como um circuito booleano não-determinístico no qual todos os valores y são gerados com a mesma probabilidade, *aceitando* (resp. *rejeitando*) a entrada x se $C(x, y) = 1$ (resp. $C(x, y) = 0$) para $\frac{2}{3}$ dos valores que pode tomar y (ou, equivalentemente, se a probabilidade de que $C(x, y) = 1$ (resp. $C(x, y) = 0$) é maior ou igual a $\frac{2}{3}$).[3] A definição de decisão de linguagens em estas generalizações continua sendo a mesma (Definição 3.4), mas considerando as novas definições de aceitação/rejeição.

Com as definições acima, e considerando somente o tamanho da entrada x (desconsiderando o tamanho de y), são introduzidas classes de complexidade algorítmica análogas às classes **NP** e **BPP**: $\mathbf{NP}_{/\text{poly}}$ (a classe de linguagens decidíveis por famílias de circuitos booleanos não-determinísticos com tamanho polinomial com respeito ao tamanho de x) e $\mathbf{BPP}_{/\text{poly}}$ (a classe de linguagens decidíveis por famílias de circuitos booleanos probabilísticos com tamanho polinomial com respeito ao tamanho de x e com erro limitado). Obviamente, $\mathbf{P}_{/\text{poly}} \subseteq \mathbf{BPP}_{/\text{poly}} \subseteq \mathbf{NP}_{/\text{poly}}$. Um resultado não obvio é que $\mathbf{BPP}_{/\text{poly}} \subseteq \mathbf{P}_{/\text{poly}}$ (portanto $\mathbf{BPP}_{/\text{poly}} = \mathbf{P}_{/\text{poly}}$), sem se considerar a condição de **P**-uniformidade.[4]

A seguir descrevemos o cálculo de anéis de polinômios, para depois generalizar os circuitos booleanos para lógicas não-clássicas.

3.2 O cálculo de anéis de polinômios

O CAP, introduzido em [Car05], consiste basicamente em traduzir fórmulas de uma lógica em polinômios com coeficientes num corpo finito (ou *corpo de Galois*), e realizar deduções através de operações sobre esses polinômios. Denotando os corpos de Galois, como é usual, por $GF(p^n)$ (onde p é a *característica* do corpo, sendo necessariamente um número primo, e n é um número natural), as operações do CAP são

[3] Nesta definição, igual ao que acontece na definição de aceitação através de MTs probabilísticas, o valor $\frac{2}{3}$ pode ser substituido por qualquer outra constante c tal que $\frac{1}{2} < c < 1$.

[4] Este resultado é enunciado em [SM02, pág. 778], como um caso especial de um resultado publicado em [BG81].

governadas pelas seguintes regras:

- *Regras do anel*, que correspondem às propriedades de adição e multiplicação num anel: a adição é associativa e comutativa, existe um elemento 'zero' e todo elemento tem 'inverso aditivo'; a multiplicação é associativa, existe um elemento 'um' e a multiplicação é distributiva respeito à adição.

- *Regras polinomiais*, que permitem as seguintes reduções de polinômios: a adição de p vezes x pode ser reduzida ao polinômio constante 0, e elementos da forma $x^i \cdot x^j$ podem ser reduzidos a $x^k (\text{mod } q(x))$, onde $k \equiv i + j (\text{mod } (p^n - 1))$ e $q(x)$ é um polinômio primitivo (i.e., um polinômio de grau n não reduzível, com coeficientes em $\mathbb{Z}_p$).

- A meta-regra de *substituição uniforme*, que consiste na substituição de variáveis por polinômios (em todas as ocorrências da variável).

- A *regra de Leibniz*, que permite substituir polinômios por polinômios equivalentes.

Definir um CAP para uma lógica específica consiste, portanto, nos seguintes passos: (1) selecionar um corpo finito F, para representar os valores de verdade, escolhendo um subconjunto de elementos como *valores designados*; (2) definir a função de tradução de fórmulas em polinômios com coeficientes em F; (3) em alguns casos, definir 'restrições polinomiais', estabelecendo equivalências e relações de conseqüência entre diferentes polinômios. Estas definições devem ser propostas de tal maneira que as operações sobre os polinômios permitam realizar deduções válidas.

O CAP pode ser visto como um método de prova ou como uma semântica algébrica na qual a estrutura dos polinômios reflete a estrutura das fórmulas da lógica. Para entender o CAP como uma semântica é preciso adaptar as noções de *valoração* (Definição 3.5) e *relação de conseqüência* (Definição 3.6) da semântica de valorações. As noções de *satisfação* e *validade* são definidas de igual maneira que na semântica de valorações, mas levando em consideração a definição de valoração para o CAP.

Definição 3.5. Seja F o corpo algébrico e X o conjunto de variáveis algébricas de um CAP para uma lógica L; uma *L-CAP-valoração* é uma função v: $X \to F$. i.e., uma L-CAP-valoração é uma atribuição de valores do corpo às variáveis algébricas.

Para simplificar a notação, a atribuição de valores às variáveis em X dada por uma valoração v será denotada por $\overrightarrow{X}_v$, e o valor do polinômio P para a valoração v será denotada por $P[\overrightarrow{X}_v]$.

Definição 3.6. Seja F o corpo algébrico e X o conjunto de variáveis algébricas de um CAP para uma lógica L, e seja $*$ a função de tradução de fórmulas de L a polinômios em $F[X]$. Considerando $D \subset F$ $(D \neq \emptyset)$ como sendo o conjunto de *valores designados*. Uma fórmula α de L é uma *L-CAP-conseqüência* do conjunto de fórmulas Γ de L (o que será denotado por $\Gamma \mid\!\approx_L \alpha$) se, para toda L-CAP-valoração v, tem-se que $\alpha^*[\overrightarrow{X}_v] \in D$ sempre que $\gamma^*[\overrightarrow{X}_v] \in D$ para toda fórmula $\gamma \in \Gamma$.

Nos casos em que D é um conjunto unitário $(D = \{d\})$ temos que $\mid\!\approx_L \alpha$ se e somente se α^* é redutível, através das regras do CAP para L, ao polinômio constante d.

Vamos agora apresentar, consecutivamente, CAPs para a lógica CP, para a lógica paraconsistente mbC (a qual é uma lógica fundamental na hierarquia das *lógicas da inconsistência formal (LFIs)*, que estende a lógica positiva clássica incluindo um conectivo de consistência e definindo um conectivo de inconsistência através da negação da consistência, mas onde inconsistência e contradição não são equivalentes, cf. [CCM07]) e para a lógica modal $S5$.[5] Nos três casos o corpo escolhido é $\mathbb{Z}_2$ (os inteiros módulo 2), portanto, as regras polinomiais são $x + x \approx 0$ e $x \cdot x \approx x$, onde a relação $\approx$ indica que o polinômio da esquerda pode ser reduzido ao polinômio da direita. Consideraremos o 1 de $\mathbb{Z}_2$ como o único valor designado.

[5] O CAP para CP e mbC foram introduzidos em [Car05]. No caso de mbC, uma pequena correção foi feita em [AC07]. O CAP para $S5$ foi um resultado obtido no desenvolvimento da idéia dos $S5$-circuitos. Com esse resultado escrevemos o artigo [AC09] onde, além da definição do CAP para $S5$, são estabelecidas relações deste cálculo com uma semântica algébrica para essa lógica, com lógica equacional e com o estilo equacional de provas de Dijkstra-Scholten; indicando como esses resultados podem ser estendidos a outras lógicas modais. O artigo está sendo submetido a publicação numa revista internacional arbitrada.

Sob a assinatura $\sigma = \{\neg, \rightarrow\}$ (que é um conjunto completo de operadores para CP), uma *valoração clássica* é uma função v: $ForCP \rightarrow \{0, 1\}$ (onde $ForCP$ é o conjunto de fórmulas de CP) tal que:

$$v(\neg\alpha) = 1 \text{ se e somente se } v(\alpha) = 0,$$
$$v(\alpha \rightarrow \beta) = 1 \text{ se e somente se } v(\alpha) = 0 \text{ ou } v(\beta) = 1.$$

O CAP para CP é definido de maneira que os polinômios em que são traduzidas as fórmulas representam as condições de uma valoração clássica:

Definição 3.7. Seja $X = \{x_{p_1}, x_{p_2}, \ldots\}$ um conjunto de variáveis algébricas. O *CAP para CP* é determinado pela função de tradução $*$: $ForCP \rightarrow \mathbb{Z}_2[X]$, definida recursivamente por:[6]

$$(p_i)^* = x_{p_i} \text{ se } p_i \text{ é uma variável proposicional},$$
$$(\neg\alpha)^* = \alpha^* + 1,$$
$$(\alpha \rightarrow \beta)^* = \alpha^*(\beta^* + 1) + 1.$$

Através das definições $\alpha \vee \beta \stackrel{\text{def}}{=} \neg\alpha \rightarrow \beta$ e $\alpha \wedge \beta \stackrel{\text{def}}{=} \neg(\alpha \rightarrow \neg\beta)$, são obtidas as traduções para os conectivos $\vee$ e $\wedge$, dadas por $(\alpha \vee \beta)^* = \alpha^*\beta^* + \alpha^* + \beta^*$ e $(\alpha \wedge \beta)^* = \alpha^*\beta^*$.

O seguinte exemplo mostra como são realizadas as deduções no CAP para CP.

Exemplo 3.1. $\not\approx_{CP} p \vee \neg p$:

$$\begin{aligned}(p \vee \neg p)^* &= p^*(\neg p)^* + p^* + (\neg p)^* \\ &= x_p(x_p + 1) + x_p + x_p + 1 \\ &\approx x_p x_p + x_p + 1 \\ &\approx x_p + x_p + 1 \\ &\approx 1.\end{aligned}$$

Como conseqüência de que as valorações clássicas constituem uma semântica correta e completa para CP, e da definição da função *, obtém-se o seguinte teorema (cf. [Car05]):

[6] Os produtos serão denotados por concatenação (eliminando o símbolo $\cdot$) como é usual.

Teorema 3.1. *$\vdash_{CP} \alpha$ se e somente se $\approx_{CP} \alpha$.*

Vamos agora definir o CAP para *mbC*. Nessa lógica as fórmulas são definidas, da maneira usual, sobre a assinatura $\sigma = \{\wedge, \vee, \rightarrow, \neg, \circ\}$, onde $\circ$ denota o operador unário de 'consistência' e tem-se a propriedade de que os operadores não são inter-definíveis. É possível mostrar que *mbC* não é caracterizável por matrizes finitas, mas é caracterizável por uma semântica bi-valorada não vero-funcional (cf. [CCM07]). Nessa semântica, uma *mbC-valoração* é definida como uma função v: $FormbC \rightarrow \{0, 1\}$ (onde $FormbC$ é o conjunto de fórmulas de *mbC*) tal que:

$$
\begin{aligned}
&v(\alpha \wedge \beta) = 1 \text{ se e somente se } v(\alpha) = 1 \text{ e } v(\beta) = 1,\\
&v(\alpha \vee \beta) = 1 \text{ se e somente se } v(\alpha) = 1 \text{ ou } v(\beta) = 1,\\
&v(\alpha \rightarrow \beta) = 1 \text{ se e somente se } v(\alpha) = 0 \text{ ou } v(\beta) = 1,\\
&v(\neg\alpha) = 0 \text{ implica } v(\alpha) = 1,\\
&v(\circ\alpha) = 0 \text{ implica } v(\alpha) = 0 \text{ ou } v(\neg\alpha) = 0.
\end{aligned}
$$

Note que as duas últimas condições introduzem um certo 'indeterminismo' na definição de v, pois não determinam completamente o valor das fórmulas complexas em termos dos valores das subformulas.

Para capturar o indeterminismo presente na definição das *mbC*-valorações, o CAP para *mbC* é definido introduzindo um elemento adicional: *variáveis ocultas*. As variáveis ocultas são variáveis diferentes daquelas definidas para as variáveis proposicionais, que tomam valores (pertencentes ao corpo) de uma maneira aleatória. O CAP para *mbC* é definido da seguinte maneira:

Definição 3.8. Sejam $X = \{x_{p_1}, x_{p_2}, \ldots\}$ e $X' = \{x'_{\alpha_1}, x''_{\alpha_1}, x'_{\alpha_2}, x''_{\alpha_2}, \ldots\}$ conjuntos disjuntos de variáveis algébricas, indexadas por variáveis proposicionais p_i e por fórmulas de *mbC* denotadas por α_i, respectivamente. As variáveis em X' são chamadas de *variáveis ocultas*. O *CAP para mbC* é determinado pela função de tradução

$*$: $FormbC \to \mathbb{Z}_2[X \cup X']$, definida recursivamente por:

$$
\begin{aligned}
&(p_i)^* = x_{p_i} \text{ se } p_i \text{ é uma variável proposicional,}\\
&(\alpha \wedge \beta)^* = \alpha^*\beta^*,\\
&(\alpha \vee \beta)^* = \alpha^*\beta^* + \alpha^* + \beta^*,\\
&(\alpha \to \beta)^* = \alpha^*(\beta^* + 1) + 1,\\
&(\neg\alpha)^* = \alpha^* x'_\alpha + 1,\\
&(\circ\alpha)^* = (\alpha^*(x'_\alpha + 1) + 1)x''_\alpha.
\end{aligned}
$$

O seguinte exemplo mostra uma dedução com o CAP para mbC:

Exemplo 3.2. $\approx_{mbC} (p \wedge \neg p \wedge \circ p) \to q$:

$$
\begin{aligned}
&((p \wedge \neg p \wedge \circ p) \to q)^*\\
&\qquad = (p \wedge \neg p \wedge \circ p)^*(q^* + 1) + 1\\
&\qquad = p^*(\neg p)^*(\circ p)^*(q^* + 1) + 1\\
&\qquad = x_p(x_p x'_p + 1)(x_p(x'_p + 1) + 1)x''_p(x_q + 1) + 1\\
&\qquad \approx (x_p x'_p + x_p)(x_p x'_p + x_p + 1)x''_p(x_q + 1) + 1\\
&\qquad \approx (x_p x'_p + x_p x'_p + x_p x'_p + x_p x'_p + x_p + x_p)x''_p(x_q + 1) + 1\\
&\qquad \approx 1.
\end{aligned}
$$

Como conseqüência de que as mbC-valorações constituem uma semântica correta e completa para mbC, e da definição da função *, obtém-se o seguinte teorema (cf. [Car05]):

Teorema 3.2. *$\vdash_{mbC} \alpha$ se e somente se $\approx_{mbC} \alpha$.*

No caso da lógica modal $S5$, pode ser demonstrado que essa lógica não é caracterizável por matrizes finitas (cf. [CP08]) e também não é conhecida uma semântica bi-valorada não vero-funcional para essa lógica. Portanto, para definir o CAP para $S5$ é necessário explorar outro caminho. O caminho que nós seguimos (cf. [AC09]) é baseado numa definição sintática de $S5$. Como é bem conhecido, $S5$ pode ser definido sobre a assinatura $\sigma = \{\neg, \to, \Box\}$ (onde $\Box$ denota o operador unário de

'necessitação'), definindo as fórmulas da maneira usual e estendendo CP mediante a adição dos seguintes axiomas e regra de inferência:

$$\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta), \tag{K}$$

$$\Box\alpha \to \alpha, \tag{T}$$

$$\alpha \to \Box\neg\Box\neg\alpha, \tag{B}$$

$$\Box\alpha \to \Box\Box\alpha, \tag{4}$$

$$\vdash \alpha \text{ implica } \vdash \Box\alpha. \tag{Nec}$$

Em $S5$ é usualmente definido um operador unário de 'possibilidade' $\Diamond$ por $\Diamond\alpha \stackrel{\text{def}}{=} \neg\Box\neg\alpha$. Os conectivos $\wedge$ e $\vee$ são definidos da mesma forma que em CP.

Na definição do CAP para $S5$, além de incluir variáveis ocultas, são introduzidas *restrições polinomiais*, as quais são regras adicionais a ser levadas em conta na redução dos polinômios. O CAP para $S5$ é definido por:

Definição 3.9. Seja $ForS5$ o conjunto de fórmulas de $S5$, e sejam $X = \{x_{p_1}, x_{p_2}, \ldots\}$ e $X' = \{x_{\Box\alpha_1}, x_{\Box\alpha_2}, \ldots\}$ conjuntos disjuntos de variáveis algébricas, indexadas por variáveis proposicionais p_i e por fórmulas de $S5$ da forma $\Box\alpha_i$. O *CAP para* $S5$ é determinado pela função de tradução $*$: $ForS5 \to \mathbb{Z}_2[X \cup X']$, definida por:

$$\begin{aligned}
&(p_i)^* = x_{p_i} \text{ se } p_i \text{ é uma variável proposicional,}\\
&(\neg\alpha)^* = \alpha^* + 1,\\
&(\alpha \to \beta)^* = \alpha^*(\beta^* + 1) + 1,\\
&(\Box\alpha)^* = x_{\Box\alpha}.
\end{aligned}$$

As variáveis em X' são chamadas de *variáveis ocultas* e estão condicionadas pelas seguintes restrições polinomiais:

$$x_{\Box(\alpha\to\beta)}(x_{\Box\alpha}(x_{\Box\beta} + 1)) \approx 0, \tag{cK}$$

$$x_{\Box\alpha}(\alpha^* + 1) \approx 0, \tag{cT}$$

$$\alpha^*(x_{\Box\neg\Box\neg\alpha} + 1) \approx 0, \tag{cB}$$

$$x_{\Box\alpha}(x_{\Box\Box\alpha} + 1) \approx 0, \tag{c4}$$

$$\alpha^* \approx 1 \text{ implica } x_{\Box\alpha} \approx 1. \tag{cNec}$$

Pelas definições dos operadores $\vee$ e $\wedge$, as traduções para estes conectivos são as mesmas que no caso de CP; e pela definição de $\Diamond$, a tradução para este conectivo é dada por $(\Diamond\alpha)^* = x_{\Box\neg\alpha} + 1$.

Algumas propriedades das variáveis ocultas (demonstradas em [AC09]) são:

Lema 3.1.

$$x_{\Box\bot} \approx 0, \tag{a}$$

$$x_{\Box\alpha}x_{\Box\neg\alpha} \approx 0, \tag{b}$$

$$x_{\Box\neg\neg\alpha} \approx x_{\Box\alpha}, \tag{c}$$

$$x_{\Box\neg\Box\neg\neg\alpha} \approx x_{\Box\neg\Box\alpha}, \tag{d}$$

$$x_{\Box\alpha} \approx 1 \text{ } or \text{ } x_{\Box\beta} \approx 1 \text{ } implica \text{ } x_{\Box(\alpha\vee\beta)} \approx 1, \tag{e}$$

$$x_{\Box(\alpha\wedge\beta)} \approx x_{\Box\alpha}x_{\Box\beta}, \tag{f}$$

$$x_{\Box\Box\alpha} \approx x_{\Box\alpha}, \tag{g}$$

$$x_{\Box\neg\Box\Box\alpha} \approx x_{\Box\neg\Box\alpha}, \tag{h}$$

$$x_{\Box\neg\Box\alpha} + 1 \approx x_{\Box\alpha}. \tag{i}$$

Um exemplo de dedução realizada com o CAP para $S5$, que mostra o uso das restrições polinomiais, é o seguinte:

Exemplo 3.3. $\not\approx_{S5} \Box(\Box(p \to \Box p) \to p) \to \Box(\Diamond\Box p \to p)$:

$$\begin{aligned}
&(\Box(\Box(p \to \Box p) \to p) \to \Box(\Diamond\Box p \to p))^* \\
&= (\Box(\Box(p \to \Box p) \to p))^*((\Box(\Diamond\Box p \to p))^* + 1) + 1 \\
&= x_{\Box(\Box(p\to\Box p)\to p)}(x_{\Box(\Diamond\Box p\to p)} + 1) + 1.
\end{aligned}$$

Por outro lado temos que:

$$\begin{aligned}
(\Diamond\Box p \to p)^* &= (\Diamond\Box p)^*(p^* + 1) + 1 \\
&= (x_{\Box\neg\Box p} + 1)(p^* + 1) + 1 \\
&\approx (x_{\Box\Diamond\neg p} + 1)(\neg p)^* + 1 \\
&\approx 1 \text{ (pela restrição polinomial (cB)).}
\end{aligned}$$

Logo, pela restrição polinomial (cNec), obtém-se que $x_{\Box(\Diamond\Box p\to p)} \approx 1$. Portanto, $(\Box(\Box(p \to \Box p) \to p) \to \Box(\Diamond\Box p \to p))^* \approx 1$.

As propriedades (a) e (b) do Lema 3.1 permitem eliminar algumas ocorrências de variáveis ocultas nos polinômios, enquanto as outras propriedades (com exceção da propriedade (e)) permitem reduzir a complexidade das formulas que indexam às variáveis ocultas. Porém, essas não são todas as reduções possíveis. Outras reduções válidas são as seguintes:

Lema 3.2.

$x_{\Box\neg(\alpha\wedge\beta)} \approx x_{\Box(\neg\alpha\vee\neg\beta)}$, $x_{\Box\neg(\alpha\vee\beta)} \approx x_{\Box\neg\alpha}x_{\Box\neg\beta}$,

$x_{\Box\alpha}x_{\Box(\alpha\vee\beta)} \approx x_{\Box\alpha}$, $x_{\Box(\alpha\vee\neg\alpha)} \approx 1$,

$x_{\Box(\alpha\vee\alpha)} \approx x_{\Box\alpha}$, $x_{\Box(\alpha\vee\beta)} \approx x_{\Box(\beta\vee\alpha)}$,

$x_{\Box(\alpha\vee(\beta\vee\delta))} \approx x_{\Box((\alpha\vee\beta)\vee\delta)}$, $x_{\Box(\alpha\vee\neg\neg\beta)} \approx x_{\Box(\alpha\vee\beta)}$,

$x_{\Box(\alpha\vee\neg(\beta\wedge\delta))} \approx x_{\Box(\alpha\vee(\neg\beta\vee\neg\delta))}$, $x_{\Box(\alpha\vee\neg(\beta\vee\delta))} \approx x_{\Box(\alpha\vee\neg\beta)}x_{\Box(\alpha\vee\neg\delta)}$,

$x_{\Box(\alpha\vee(\beta\vee\neg\beta))} \approx 1$, $x_{\Box(\alpha\vee(\beta\wedge\delta))} \approx x_{\Box(\alpha\vee\beta)}x_{\Box(\alpha\vee\delta)}$,

$x_{\Box(\alpha\vee\Box\beta)} \approx x_{\Box\alpha}x_{\Box\beta} + x_{\Box\alpha} + x_{\Box\beta}$, $x_{\Box(\alpha\vee\Diamond\beta)} \approx x_{\Box\alpha}x_{\Box\neg\beta} + x_{\Box\neg\beta} + 1$.

Considerando que $\to$ pode ser definido por $\alpha \to \beta \stackrel{\text{def}}{=} \neg\alpha \vee \beta$ em $S5$, as propriedades do Lema 3.1 (desconsiderando a propriedade (e)) e do Lema 3.2 permitem reduzir as variáveis $x_{\Box\alpha}$ a casos em que α é um literal (i.e. uma variável proposicional ou a negação de uma variável proposicional) ou é uma seqüência de disjunções de literais não contraditórios (i.e. uma seqüência de disjunções de literais onde não aparecem p_i e $\neg p_i$, para alguma variável proposicional p_i).

A restrição polinomial (cNec) pode ser substituída pelas propriedades dos lemas 3.1 e (e), devido a que o polinômio α^* não é reduzível ao polinômio constante 1 para os casos nos quais α é uma seqüência de disjunções de literais não contraditórios. Isso permite realizar deduções de uma maneira mais simples e totalmente equacional.

Em [AC09] é demonstrado o seguinte teorema:

Teorema 3.3. $\vdash_{S5} \alpha$ *se e somente se* $\models\!\!\!\approx_{S5} \alpha$.

A seguir, vamos generalizar o modelo de circuitos booleanos para lógicas não-clássicas, aproveitando as definições anteriores.

3.3 Circuitos baseados em lógicas não-clássicas

Havendo descrito o CAP, fica agora fácil estender a noção de circuitos booleanos para lógicas caracterizáveis por esse tipo de semântica:

Definição 3.10. Seja L uma lógica proposicional provida de um CAP sobre o corpo finito $GF(p^k)$. Um *L-circuito* é um grafo direcionado acíclico $C = (V, E)$, onde V é o conjunto de *nós* e E é o conjunto de *arestas*. Os nós aos quais não chega nenhuma aresta são chamados *entradas* do circuito e são denotados por variáveis (x_1, x_2, ..., x_n) ou por constantes correspondentes a elementos em $\{0, 1, \ldots, p^k - 1\}$; os outros nós são chamados *portas lógicas*, e são associados a conectivos de L. As portas lógicas têm um número de arestas de entradas igual à aridade do respectivo conectivo. As arestas sem nós de saída são chamadas *saídas* do circuito.

Um L-circuito computa funções da forma $f\colon \{0, 1, \ldots, p^k - 1\}^n \to \{0, 1, \ldots, p^k - 1\}^m$ da seguinte maneira:

Definição 3.11. Seja L uma lógica proposicional provida de um CAP sobre o corpo finito $GF(p^k)$ e seja C um L-circuito com n variáveis de entrada e m portas de saída. Para o valor de entrada $x \in \{0, 1, \ldots, p^k - 1\}^n$ o circuito C *calcula* o valor de saída atribuindo valores a seus nós de maneira indutiva:

- o nó correspondente à variável de entrada x_i toma como valor o i-ésimo dígito de x; e
- o nó correspondente a uma porta lógica, com arestas de entrada vindo de nós cujos valores já foram atribuídos, toma o valor obtido de aplicar (sobre os valores de entrada) o polinômio definido no CAP para o respetivo conectivo lógico; o que pode introduzir variáveis ocultas no processo de computação. Tais variáveis ocultas tomam valores em $\{0, 1, \ldots, p^k - 1\}$, atribuídos de maneira aleatória e levando em consideração as restrições polinomiais (se existirem).

Os valores atribuídos aos nós que são saídas do circuito, ordenados de alguma maneira preestabelecida, representam o valor de *saída* do circuito (denotado por $C(x)$).

Note que, devido ao possível uso de variáveis ocultas, existem L-circuitos não determinísticos mesmo considerando que as entradas são determinísticas, i.e., o indeterminismo neste modelo é introduzido pelas operações lógicas, e não por considerar as entradas como sendo não determinísticas (como é o caso na definição de circuitos booleanos não-determinísticos). O indeterminismo é então uma conseqüência da lógica subjacente.

É importante também notar que, tendo em conta que os corpos usados na definição do CAP são finitos, a computabilidade clássica (i.e., através de máquinas de Turing ou formalismos equivalentes) das raízes dos polinômios definidos sobre esses corpos, assegura a computabilidade das funções computadas através de famílias uniformes de L-circuitos, para qualquer lógica L. Portanto, os L-circuitos não representam vantagens quanto à computabilidade (isto é, quanto á extensão da classe de funções computáveis), mas, como será mostrado a seguir, podem representar interessantes vantagens quanto à eficiência.

A seguir descrevemos casos específicos de L-circuitos, para as lógicas mbC e $S5$, esclarecendo como o indeterminismo introduzido pelas variáveis ocultas pode levar a computações eficientes.

3.3.1 Circuitos paraconsistentes mbC

O modelo de mbC-circuitos é completamente determinado pelas definições 3.8, 3.10 e 3.11. Uma primeira pergunta que surge, de maneira natural, é se a classe de funções computáveis através de famílias uniformes de mbC-circuitos é igual à classe de funções computáveis através de famílias uniformes de circuitos booleanos (ou, equivalentemente, através de MTs). Por um lado, pelo argumento apresentado acima para L-circuitos em geral, toda função computada por uma família uniforme de mbC-circuito é Turing-computável (i.e. computável em MTs). Na outra direção, devido ao fato de mbC ser uma sublógica de CP, podemos ser levados a pensar que a classe de funções computáveis através de famílias uniformes de mbC-circuitos seria menor

que a classe de funções Turing-computáveis. Porém, como mostrado em [CCM07, Teorema 74], CP pode ser traduzida em mbC de maneira conservativa através da função f: $ForCP \rightarrow FormbC$ definida por:

$$t(p) = p \text{ para toda variável proposicional } p,$$
$$t(\alpha \# \beta) = t(\alpha) \# t(\beta) \text{ para } \# \in \{\wedge, \vee, \rightarrow\},$$
$$t(\neg\alpha) = t(\alpha) \rightarrow (p \wedge (\neg p \wedge \circ p)).$$

Portanto CP pode também ser vista como uma sublógica de mbC. Através da tradução conservativa t (que assegura que $\Gamma \vdash_{CP} \alpha$ se e somente se $t(\Gamma) \vdash_{mbC} t(\alpha)$, onde $t(\Gamma) = \{t(\gamma) | \gamma \in \Gamma\}$), famílias uniformes de CP-circuitos podem ser convertidas em famílias uniformes de mbC-circuitos computando as mesmas funções.

As vantagens dos mbC-circuitos dizem respeito à complexidade algorítmica. A seguir, daremos algumas indicações de como as variáveis ocultas podem ser usadas neste modelo.

Considere inicialmente o mbC-circuito definido pela fórmula $\neg p \wedge \neg q$, representado graficamente pela Figura 3.1 (denotamos as variáveis de entrada colocando como subíndices as variáveis proposicionais respetivas). Se as variáveis de entrada associadas às variáveis proposicionais, tomam ambas o valor 1, o valor de saída do circuíto representa a conjunção da negação das duas variáveis ocultas (x'_p e x'_q). Portanto o circuito computa essa conjunção aleatoriamente, embora as entradas tomem o valor fixo 1.

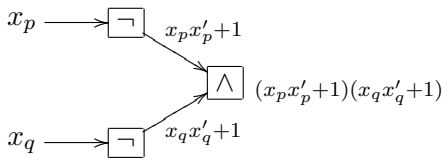

Figura 3.1: mbC-circuito para a fórmula $\neg p \wedge \neg q$

Em mbC pode ser definido um operador de negação clássico $\sim$ por $\sim \alpha \stackrel{\text{def}}{=} \alpha \rightarrow (p \wedge (\neg p \wedge \circ p))$ (pode se verificar que $(\sim \alpha)^* = \alpha^* + 1$, tal como para a negação

em CP). O circuito para calcular a negação clássica é representado graficamente na Figura 3.2.

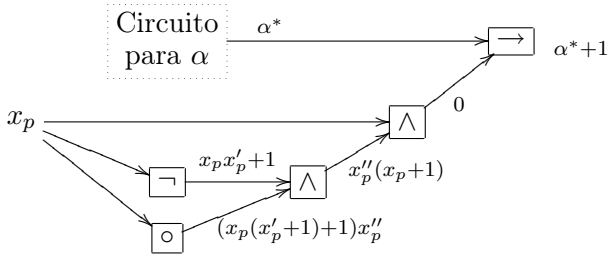

Figura 3.2: mbC-circuito para a fórmula $\sim \alpha = \alpha \rightarrow (p \wedge (\neg p \wedge \circ p))$

Se quisermos computar aleatoriamente a conjunção das variáveis ocultas (diferentemente da conjunção de suas negações), isso pode ser feito através do mbC-circuito associado à fórmula $\sim \neg p \wedge \sim \neg q$: se as variáveis de entrada tomam o valor fixo 1, o circuito tem como saída $x'_p x'_q$.

Generalizando o procedimento anterior, substituindo uniformemente as variáveis proposicionais p_i de uma fórmula α de CP por $\sim (\neg p_i)$, obtém-se uma fórmula de mbC cujo circuito associado, ao fixar os valores de entrada com o valor 1, computa aleatoriamente a função computada deterministicamente pelo circuito booleano correspondente a α. Desta maneira, podem-se traduzir computações determinísticas (em circuitos booleanos) a computações não-determinísticas (em mbC-circuitos) sem considerar entradas aleatórias.

Considerando que as variáveis ocultas tomam os valores 0 e 1 com a mesma probabilidade (i.e., com probabilidade $1/2$), o mbC-circuito definido para a fórmula α, como especificado anteriormente, determina a satisfatibilidade de α com probabilidade dada pelo número de valorações que fazem α tomar o valor verdadeiro. Agora a pergunta é a seguinte: podem ser definidos mbC-circuitos tais que, através das operações com variáveis ocultas, calculem deterministicamente (ou pelo menos probabilisticamente, com probabilidade maior ou igual a $2/3$) a satisfatibilidade das respetivas fórmulas α?

Variáveis ocultas podem aparecer nos níveis intermediários dos mbC-circuitos

(ou dos L-circuitos em geral) e depois desaparecer devido a reduções polinomiais na relação com outros conectivos. Isso acontece, por exemplo, no mbC-circuito da Figura 3.2. Isso sugere que as variáveis ocultas podem ser usadas de alguma maneira 'inteligente' para resolver problemas de forma que os resultados obtidos sejam corretos com uma alta probabilidade, permitindo assim resolver problemas de maneira eficiente.

Para especificar de maneira mais ampla a questão colocada acima, vamos introduzir a classe de complexidade algorítmica $\mathbf{BPmbC}_{/\mathrm{poly}}$ como sendo a classe de linguagens decidíveis por famílias de mbC-circuitos com número polinomial de portas (de acordo com o tamanho do dado de entrada) e com erro limitado (definida de maneira análoga a como foi definida a classe $\mathbf{BPP}_{/\mathrm{poly}}$, mas sem necessidade de dividir a entrada em dois, pois o indeterminismo nos mbC-circuitos é introduzido pelos operadores lógicos). A questão acima corresponde então a determinar a relação de $\mathbf{BPmbC}_{/\mathrm{poly}}$ com outras classes de complexidade algorítmica (em particular, com as que foram mencionadas na Seção 3.1), é isso que fazemos no seguinte teorema.

Teorema 3.4. $\boldsymbol{BPmbC_{/poly} = BPP_{/poly}}$.

Demonstração. Primeiro vamos provar que $\mathbf{BPP}_{/\mathrm{poly}} \subseteq \mathbf{BPmbC}_{/\mathrm{poly}}$: circuitos booleanos probabilísticos podem ser traduzidos em mbC-circuitos trocando as entradas não determinísticas por subcircuitos correspondentes a mbC-fórmulas da forma $\neg p_i$ (com a condição de que x_{p_i} não pode ser uma entrada do circuito original) e fixando o valor dos x_{p_i} com o valor 1. Essa tradução acrescenta o número de portas lógicas em uma quantidade igual ao número de variáveis de entradas não determinísticas no circuito original, que tem de ser menor ao tamanho do circuito, portanto o tamanho do circuito obtido continua sendo polinomial com respeito ao tamanho do dado de entrada.

Agora vamos provar que $\mathbf{BPmbC}_{/\mathrm{poly}} \subseteq \mathbf{BPP}_{/\mathrm{poly}}$: mbC-circuitos podem ser traduzidos em circuitos booleanos probabilísticos trocando portas de negação e de consistência ($\circ$) por subcircuitos correspondentes a CP-fórmulas, respectivamente, da forma $\neg(\alpha \wedge p'_\alpha)$ e $\neg(\alpha \wedge \neg p'_\alpha) \wedge p''_\alpha$; onde α é a fórmula que descreve o subcircuito cuja saída é a entrada da porta a ser trocada, enquanto p'_α e p''_α são novas variáveis

proposicionais que correspondem a novas entradas $x_{p'_\alpha}$ e $x_{p''_\alpha}$ que devem ser consideradas como não-determinísticas. O número de portas acrescentadas na troca das portas de negação e de consistência é fixo, e o número total de trocas é igual ao número desses conectivos no circuito original (que é obviamente menor ao tamanho do circuito), portanto o tamanho do circuito obtido continua sendo polinomial com respeito ao tamanho do dado de entrada. □

Pelos resultados mencionados na Seção 3.1 e o teorema anterior, $\mathbf{BPmbC}_{/\mathrm{poly}} = \mathbf{P}_{/\mathrm{poly}} = \mathbf{P}$, se não se considerar a condição de **P**-uniformidade. Contudo, é ainda possível que famílias **P**-uniformes de $S5$-circuitos consigam decidir linguagens de maneira mais eficiente que as famílias **P**-uniformes de circuitos booleanos (i.e. que $\mathbf{BPmbC}_{/\mathrm{poly}} \not\subseteq \mathbf{P}_{/\mathrm{poly}}$, com a condição de **P**-uniformidade). Nesse caso, os mbC-circuitos mostrariam, de uma outra maneira, que a complexidade algorítmica é relativa às lógicas subjacentes aos modelos de computação.

3.3.2 Circuitos modais $S5$

De maneira similar aos mbC-circuitos, os $S5$-circuitos estão completamente determinados pela definição geral de L-circuito (definições 3.10 e 3.11) e pela definição do CAP para a lógica modal $S5$ (Definição 3.9). É fácil mostrar também que a classe de funções computáveis por famílias uniformes de $S5$-circuitos é a mesma classe das funções computáveis por famílias uniformes de circuitos booleanos. Vamos mostrar a seguir, contudo, como os $S5$-circuitos podem ser usados para calcular funções e resolver problemas de maneira eficiente.

Diferentemente do que acontece no CAP para mbC, no CAP para $S5$ as variáveis ocultas nem sempre tomam valores de maneira totalmente aleatória. As restrições polinomiais, presentes na definição do CAP para $S5$, estabelecem certas inter-relações entre variáveis ocultas e variáveis associadas a variáveis proposicionais. Por exemplo, pela restrição polinomial (cT), se x_p toma o valor 0 então $x_{\Box p}$ também toma o valor 0. Esta característica impede pensar nas variáveis ocultas, introduzidas pelos conectivos lógicos, como simples entradas aleatórias, permitindo um melhor aproveitamento dessa variáveis no processo de computação.

De maneira análoga ao que fizemos com os *mbC*-circuitos, vamos introduzir a classe de complexidade algorítmica $\mathbf{BPS5}_{/\text{poly}}$ como sendo a classe de linguagens decidíveis por *mbC*-circuitos com número polinomial de portas e com erro limitado. Para mostrar as potencialidades dos $S5$-circuitos quanto à eficiência vamos estabelecer algumas relações entre $\mathbf{BPS5}_{/\text{poly}}$ e outras classes de complexidade algorítmica.

Teorema 3.5. $\boldsymbol{BPP}_{/poly} \subseteq \boldsymbol{BPS5}_{/poly}$

Demonstração. Circuitos booleanos probabilísticos podem ser traduzidos em $S5$-circuitos trocando as entradas aleatórias por subcircuitos correspondentes a fórmulas $\Box p_i$ (com a condição de que x_{p_i} não pode ser uma entrada do circuito original) e fixando o valor das entradas x_{p_i} com o valor 1, o que acrescenta o número de portas do circuito original no máximo em um número polinomial respeito ao tamanho do dado de entrada. □

Vamos voltar ao problema que estávamos tentando resolver na seção anterior, que consiste em determinar satisfatibilidade de fórmulas de CP de maneira eficiente aproveitando as variáveis ocultas presentes nos L-circuitos. Vamos considerar α como sendo uma fórmula de CP e vamos denotar por $\alpha[p_i/\beta_i]$ à fórmula obtida ao substituir em α as variáveis p_i por β_i. Através do $S5$-circuito C definido por $\Diamond\alpha[p_i/\Box p_i] \vee \Diamond\alpha[p_i/\Box q_i]$ (Figura 3.3), fixando os valores das entradas x_{p_i} e x_{q_i} com o valor 1, pode-se determinar a satisfatibilidade de α de maneira probabilística com erro limitado, acrescentando somente um número polinomial de portas respeito ao tamanho da fórmula α (denotaremos por C_1 o circuito descrito pela fórmula $(\Diamond\alpha[p_i/\Box p_i])$ e por C_2 o circuito descrito pela fórmula $(\Diamond\alpha[p_i/\Box q_i])$):

- Se α é tautologia $(\Diamond\alpha[p_i/\Box p_i])^* = x_{\Box\neg\alpha[p_i/\Box p_i]} + 1 \approx 1$ (pelo Teorema 3.3, a restrição polinomial (cNec) e a propriedade (b) do Lema 3.1), portanto $C_1(x) = 1$ independentemente do valor de entrada x. Se α é contradição $(\Diamond\alpha[p_i/\Box p_i])^* \approx 0$ (pelo Teorema 3.3 e a restrição polinomial (cNec)), portanto $C_1(x) = 0$ independentemente do valor de entrada x. Se α não é tautologia nem contradição, a probabilidade de que $C_1(1^n) = 1$ é igual a soma da probabilidade de que $(\alpha[p_i/\Box p_i])^* = 1$ (com as entradas x_{p_i} fixadas no valor 1),

que é maior do que 0, mais a probabilidade de que $x_{\Box\neg\alpha[p_i/\Box p_i]} = 0$ dado que $(\alpha[p_i/\Box p_i])^* = 0$, que por pressuposto é $\frac{1}{2}$. Portanto, a probabilidade de que $C_1(1^n) = 1$ é estritamente maior do que $\frac{1}{2}$.

- Obviamente C_2 tem o mesmo comportamento de C_1, conseqüentemente temos que $C(1^{2n}) = 1$ com probabilidade 1 se α for tautologia, $C(1^{2n}) = 0$ com probabilidade 1 se α for contradição e $C(1^{2n}) = 1$ com probabilidade maior do que $\frac{3}{4}$ se α não é contradição nem tautologia.

- O número de portas de C é duas vezes o número de conectivos em α, mais duas vezes o número de variáveis proposicionais em α, mais 3 (três).

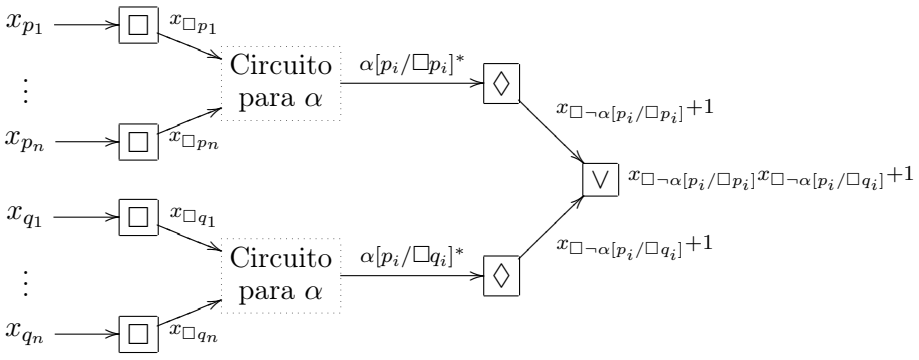

Figura 3.3: $S5$-circuito para determinar a satisfatibilidade de α

O resultado acima sugere que todas as linguagens em **NP** podem ser decididas por famílias de $S5$-circuitos, com número polinomial de portas em relação ao tamanho do dado de entrada, de maneira probabilística e com erro limitado (i.e. que $\mathbf{NP}_{/\text{poly}} \subseteq \mathbf{BPS5}_{/\text{poly}}$); pois indica como determinar satisfatibilidade de fórmulas de CP (que é um problema **NP**-completo) de maneira eficiente. Porém, a solução proposta define um $S5$-circuito diferente para cada fórmula de CP (não uma família com um único circuito para cada tamanho do dado de entrada) e a generalização não parece fácil, pois ao considerar circuitos que permitam representar fórmulas arbitrárias (de um certo tamanho) perde-se a possibilidade de que a fórmula que descreve tal circuito seja uma tautologia ou uma contradição, não permitindo aproveitar de

maneira direta o condição (cNec). Contudo, o subcircuito associado à fórmula α pode ser considerado como se fosse um 'oráculo', que calcula o valor de verdade de α recebendo como entrada os valores de suas variáveis proposicionais. Deste modo, a inclusão $\mathbf{NP}_{/\text{poly}} \subseteq \mathbf{BPS5}_{/\text{poly}}$ pode ser obtida como relativa a oráculos que correspondem a circuitos na classe $\mathbf{P}_{/\text{poly}}$.

É evidente que a validação da condição (cNec) envolve uma operação complexa: a determinação de se $\alpha^* \approx 1$. Contudo, os resultados que apresentamos são evidências a favor da eficiência dos $S5$-circuitos, o que indica uma vez mais que a complexidade algorítmica pode ser vista como sendo relativa às lógicas subjacentes aos modelos de computação.

De maneira similar ao que acontece com os modelos de MTPs, é difícil aceitar, de maneira direta, que o modelo de L-circuitos (em particular para o caso de $S5$) seja um modelo de computação 'razoável' (no sentido de ser fisicamente construtível). É interessante, portanto, pensar em L-circuitos definidos através de outras lógicas não-clássicas ou em subclasses de $S5$-circuitos que permitam ainda realizar tarefas de maneira eficiente, mas que não envolvam operações complexas de maneira tão evidente. Nesta direção, retomando idéias que já foram apresentadas no capítulo anterior, pode se considerar a lógica paraconsistente $PNS5$ (definida na Seção 2.4.1). Na subclasse de $PNS5$-circuitos é possível ainda computar a satisfatibilidade de uma CP-fórmula α, de maneira probabilística com erro limitado, através do circuito definido pela fórmula $\alpha[p_i/\Box p_i] \vee (\bullet\alpha[p_i/\Box q_i] \vee \bullet\alpha[p_i/\Box r_i])$. O que pode ser vantajoso se existir um CAP para $PNS5$ que não envolva operações complexas.

No Capítulo 5 serão apresentadas relações do modelo de L-circuitos (para os casos particulares das lógicas aqui consideradas) com modelos de computação quântica, procurando, desta maneira, explicar algumas características da computação quântica a partir de outra perspectiva lógica e, na outra direção, dando evidências para que se possa considerar este modelo (para certas lógicas específicas) como 'razoável'.

Capítulo 4

Computação quântica

A *computação quântica* investiga os fundamentos de uma teoria da computação baseada nos princípios da mecânica quântica (i.e., da teoria física que trata da estrutura da matéria). Embora a mecânica quântica tenha sido formulada na década de 1920 e a teoria da computabilidade tenha surgido na década de 1930, foi só na década de 1980 que foram definidos os primeiros modelos de computação através da mecânica quântica, e a partir daí começaram a ser estudadas as vantagens que esta nova abordagem dos processos de computação pode trazer.

Numa conferência sobre 'física e computação' que teve lugar no MIT em 1981, Richard Feynman formulou uma pergunta crucial: seriam os fenômenos da mecânica quântica eficientemente simuláveis numa máquina de Turing clássica? Feynman apresentou boas razões para acreditar que a resposta fosse negativa, argumentando que tal simulação parece impossível sem incorrer num retardo exponencial. Além disso, sugeriu que para poder simular eficientemente a evolução dos sistemas quânticos seriam necessários computadores que funcionassem de acordo com as leis da mecânica quântica; não definiu, porém, um modelo apropriado para tal objetivo. Estas idéias foram publicadas em [Fey82].[1]

Poucos anos depois David Deutsch formalizou a idéia de Feynman, definindo

[1] É importante indicar que Yuri Manin, em seu livro [Man80], já tinha apontado as possíveis vantagens de construir computadores sob os princípios da mecânica quântica, mas nesse momento a idéia não foi suficientemente divulgada.

modelos de computação baseados na mecânica quântica. Tendo inicialmente generalizado o modelo de MTs através de um modelo de *máquinas de Turing quânticas (MTQs)* (em [Deu85]), posteriormente Deutsch formulou o modelo de circuitos lógicos através de um modelo de *circuitos quânticos (CQs)* (em [Deu89]). Esses modelos serão descritos na Seção 4.2.

Em [Deu85] Deutsch, além de definir o modelo de MTQs, mostrou a existência de uma MTQ universal, a qual tem a limitação de estar submetida a um retardo exponencial na simulação da MTQ que recebe como entrada. Essa limitação foi superada por Ethan Bernstein e Umesh Vazirani, em [BV93] (uma versão melhorada desse artigo é [BV97]), definindo um modelo de MTQ universal que permite simular qualquer MTQ em tempo polinomial.

Outro resultado importante, que vale a pena mencionar aqui, é a demonstração da equivalência entre os modelos de MTQs e CQs obtida por Andrew Yao (publicada em [Yao93]). Yao demonstrou que qualquer função computável em tempo polinomial numa MTQ pode ser computada por um CQ de tamanho polinomial. Esse resultado legitima a definição de algoritmos quânticos através de CQs, ao invés de MTQs, o que parece facilitar o trabalho de se especificar algoritmos quânticos.

Desde a definição dos modelos de MTQs e CQs, são vários os algoritmos quânticos que têm sido definidos (na Seção 4.3 serão descritos alguns deles), mostrando algumas vantagens quanto à eficiência desses modelos de computação com respeito a seus análogos clássicos. Entre esses algoritmos, o mais influente, que levou muitos pesquisadores a acreditar nas potencialidades da computação quântica e impulsou fortemente a pesquisa nessa área, é o algoritmo de fatoração de inteiros em tempo polinomial, proposto por Peter Shor em [Sho94] (uma versão mais recente desse artigo é [Sho97]). Não existe nenhum algoritmo clássico que permita fatorar inteiros em tempo polinomial, e a complexidade para realizar esta tarefa é chave para o bom funcionamento dos protocolos de encriptação de dados. Porém, pelo fato de que não se conhece a verdadeira complexidade deste problema (é um problema na classe **NP**, o qual se acredita que não pertence a **P** e que não é NP-Completo), a existência deste algoritmo não implica, dentro do quadro teórico atual, a superioridade em eficiência dos modelos de computação quântica.

Antes de descrever os modelos de computação quântica e de explicar alguns algoritmos, vamos apresentar uma breve introdução à mecânica quântica, limitando-nos aos elementos necessários para uma compreensão dos modelos de MTQs e CQs.

4.1 Uma breve introdução à mecânica quântica

A mecânica quântica surgiu, a começos do século XX, como resposta ao descobrimento de vários fenômenos que estavam em contradição com as leis postuladas pela física clássica, principalmente o chamado problema de 'radiação de corpo negro'. Os cálculos realizados, através das teorias físicas predominantes nessa época, para determinar a intensidade de radiação de corpos negros quando aquecidos (em função dos comprimentos de onda), eram totalmente divergentes com os dados obtidos experimentalmente. Para resolver esse problema, Max Planck em 1900 propôs considerar a amissão de radiações como ocorrendo em pacotes (ou 'quanta') de energia. Com esse pressuposto, que segundo ele mesmo era um simples artifício matemático sem valor físico, conseguiu dar uma formulação teórica que permitia calcular valores muito próximos aos obtidos experimentalmente. A idéia de 'quantização' foi depois usada por Albert Einstein, em 1905, para explicar o chamado 'efeito fotoelétrico'. Porém, foi só na década de 1920 que se chegou à formalização da teoria quântica como o resultado do trabalho conjunto de vários físicos, entre eles Niels Bohr, Werner Heisenberg, Max Born, Louis de Broglie, Erwin Schrödinger e Paul Dirac.

Inicialmente, em 1925, Heisenberg propôs a *mecânica matricial* na qual são representados os níveis de energia através de auto-valores de matrizes infinitas. Depois, em 1926, Schrödinger introduz a *mecânica ondulatória*, representando a evolução dos sistemas quânticos através de uma equação de onda, hoje chamada de 'equação de Schrödinger'. Nesse mesmo ano, o próprio Schrödinger demonstrou a equivalência entre essas duas formulações. Posteriormente, a mecânica matricial e a mecânica ondulatória foram imersas numa teoria mais geral, em boa parte graças ao esforço de Paul Dirac e John von Neuman. Foi então que essa teoria começou ser chamada de *mecânica quântica*.

Na formalização de von Neumann-Dirac a mecânica quântica é apresentada por

meio de postulados. Tais postulados indicam como representar sistemas físicos microscópicos através de estruturas matemáticas chamadas *espaços de Hilbert*, como será descrito a seguir (tomando como referência [NC00] e [Gru99]):[2]

Postulado 1. Todo sistema físico isolado tem associado um espaço de Hilbert (Definição B.15), o qual é chamado de *espaço de estados* do sistema. O sistema é completamente descrito por seu *vetor de estado*, o qual é um vetor unitário (Definição B.20) no espaço de estados do sistema.

Os espaços de Hilbert considerados na mecânica quântica são espaços de Hilbert complexos, i.e., espaços de Hilbert definidos sobre o corpo dos números complexos; portanto, os espaços de Hilbert de dimensão n (onde n é um número natural) podem ser considerados como sendo $\mathbb{C}^n$ (o espaço de Hilbert sobre o corpo $\mathbb{C}$ dos números complexos onde os vetores são n-uplas de elementos em $\mathbb{C}$). O sistema quântico mais simples e de maior relevância para a computação quântica é o *qubit* (ou *bit quântico*), utilizado como a unidade básica de informação, análogo ao *bit* na computação clássica. Formalmente, um qubit é um vetor unitário num espaço de estados bidimensional (para uma descrição matemática mais precisa ver Apêndice B) .

O estado geral de um qubit é usualmente definido usando a notação de Dirac, e tal notação é utilizada no formalismo da mecânica quântica com o objetivo de simplificar o conteúdo notacional e facilitar o entendimento das equações. Como apresentado na Definição B.18, na notação de Dirac os vetores são denotados por $|\cdot\rangle$ e são chamados de *ket* e os vetores duais (i.e., os vetores transpostos conjugados dos kets, veja Definição B.17) são denotados por $\langle\cdot|$ e são chamados de *bra*. Utilizando a notação de Dirac e tomando $|0\rangle$ e $|1\rangle$ como os vetores base do espaço de estados bidimensional, o estado geral de um qubit é definido pela equação:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{4.1}$$

onde α e β são números complexos. A condição de que o vetor ψ seja unitário (Definição B.20) é equivalente à condição de que $|\alpha|^2 + |\beta|^2 = 1$, onde $|\alpha|$ denota o

[2]No Apêndice B são apresentados elementos de álgebra linear necessários para entender a mecânica quântica no nível que aqui precisamos. As definições no apêndice serão usadas nos postulados e na descrição de conceitos a seguir.

módulo de α;[3] tal condição é comumente chamada de *condição de normalização*. Os coeficientes α e β são denominados *amplitudes de probabilidade*.

Comumente $|\,0\,\rangle$ e $|\,1\,\rangle$ são tomados como os vetores-coluna:

$$|\,0\,\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |\,1\,\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.2}$$

Um qubit é diferente de um bit clássico pelo fato de que o qubit pode estar numa *superposição* de estados (dada pela equação (4.1)), o qual pode ser interpretado fisicamente como a existência simultânea do qubit nos estados $|\,0\,\rangle$ e $|\,1\,\rangle$.

Existem vários referentes físicos que podem ser usados na implementação de um qubit; um deles é o estado de *spin*, ou direção do momento angular, de uma partícula de spin $1/2$.

A função que descreve o estado de um sistema quântico, atribuindo amplitudes de probabilidade aos diferentes estados base do sistema (que no caso de um qubit é dada pela equação 4.1), é usualmente chamada de *função de onda*.

Postulado 2. A evolução de um sistema quântico fechado é descrita por uma *transformação unitária*, i.e., o estado $|\,\psi\,\rangle$ de um sistema no tempo t_1 se relaciona com o estado $|\,\psi'\,\rangle$ do sistema no tempo t_2 por um operador unitário U (Definição B.26), chamado *operador de evolução*, o qual depende somente dos tempos t_1 e t_2,

$$|\,\psi'\,\rangle = U|\,\psi\,\rangle. \tag{4.3}$$

O postulado 2 descreve como os estados de um sistema quântico em dois instantes de tempo diferentes estão relacionados, enquanto o sistema não é observado. Uma versão mais refinada deste postulado pode ser expressa através da equação de Schrödinger, a qual descreve a evolução de um sistema quântico em tempo continuo.[4] Os modelos de MTQs e CQs são modelos discretos de computação, portanto é suficiente apresentar o Postulado 2 na sua versão mais simples.[5]

[3] O módulo de um número complexo $\alpha = a + bi$ é definido por $|\alpha| = \sqrt{\alpha\overline{\alpha}} = \sqrt{a^2 + b^2}$, onde $\overline{\alpha} = a - bi$ é chamado o *conjugado* de α.

[4] A equação de Schrödinger e a derivação do postulado 2 (a partir dessa versão mais refinada) são apresentadas em [NC00, p. 82-83].

[5] Existem modelos de computação quântica contínua, como os descritos em [LB99], [Bra98] e [VS00]; mas esses modelos não serão aqui apresentados.

Um operador unitário U pode ser representado, com respeito a uma base B, por meio de uma matriz M (Definição B.10) tal que, para estados $|\,i\,\rangle$, $|\,j\,\rangle \in B$, o elemento na fila i e coluna j de M representa a amplitude de transição do estado $|\,j\,\rangle$ ao estado $|\,i\,\rangle$. Além disso, como U é um operador unitário, tem a propriedade de que se $U|\,\psi\,\rangle = |\,\psi'\,\rangle$ então $|\,\psi\,\rangle = U^\dagger|\,\psi'\,\rangle$ (onde $U^\dagger$ representa a transposta conjugada de U), o que expressa o fato físico de que os processos quântico são *reversíveis* (desde que não seja realizada nenhuma medição sobre o sistema) e que $U^\dagger$ representa o processo inverso a U.

Exemplos de matrizes unitárias para operar sobre qubits são as chamadas *matrizes de Pauli*:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{4.4}$$

A linearidade (Definição B.9) dos operadores unitários permite especificar as transformações que realizam indicando somente as mudanças sobre os estados base. Assim, as matrizes de Pauli realizam as seguintes transformações:

$$\begin{aligned} X{:}|\,0\,\rangle &\mapsto |\,1\,\rangle, \\ |\,1\,\rangle &\mapsto |\,0\,\rangle. \end{aligned} \tag{4.5}$$

$$\begin{aligned} Y{:}|\,0\,\rangle &\mapsto i|\,1\,\rangle, \\ |\,1\,\rangle &\mapsto -i|\,0\,\rangle. \end{aligned} \tag{4.6}$$

$$\begin{aligned} Z{:}|\,0\,\rangle &\mapsto |\,0\,\rangle, \\ |\,1\,\rangle &\mapsto -|\,1\,\rangle. \end{aligned} \tag{4.7}$$

Na evolução de um sistema quântico pode ocorrer um fenômeno denominado *interferência quântica*. Esse fenômeno é representado pela soma (*interferência construtiva*) ou a subtração (*interferência destrutiva*) das amplitudes de probabilidade para o mesmo estado base. Considere, por exemplo, o estado de um qubit $|\,\psi\,\rangle = {}^1\!/\!\sqrt{2}(|\,0\,\rangle + |\,1\,\rangle)$ e o operador unitário:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.8}$$

O operador H realiza as seguintes transformações:

$$\begin{aligned} H{:}|\,0\,\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|\,0\,\rangle + |\,1\,\rangle\right), \\ |\,1\,\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|\,0\,\rangle - |\,1\,\rangle\right). \end{aligned} \tag{4.9}$$

Ao aplicar H sobre o estado $|\,\psi\,\rangle$ obtém-se:

$$\begin{aligned} H|\,\psi\,\rangle &= \frac{1}{\sqrt{2}}(H|\,0\,\rangle + H|\,1\,\rangle) \text{ (pela linearidade de } H) \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|\,0\,\rangle + |\,1\,\rangle) + \frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) \\ &= \frac{1}{2}|\,0\,\rangle + \frac{1}{2}|\,1\,\rangle + \frac{1}{2}|\,0\,\rangle - \frac{1}{2}|\,1\,\rangle \\ &= |\,0\,\rangle. \end{aligned}$$

Note que as amplitudes de probabilidade do estado $|\,0\,\rangle$ se somam (o que indica que houve interferência construtiva), enquanto as amplitudes de probabilidade do estado $|\,1\,\rangle$ se subtraem (o que indica que houve interferência destrutiva).

Postulado 3. As propriedades dos sistemas quânticos que podem ser medidas são chamadas *observáveis* e são representadas por operadores auto-adjuntos (Definição B.25). A saída numérica de uma medição, de um estado $|\,\psi\,\rangle$ com respeito a um observável A, é um dos auto-valores de A (os conceitos de auto-valor, auto-vetor e auto-espaço são apresentados na Definição B.22). Ao se realizar a medição, o auto-valor m_i é obtido com probabilidade:

$$Pr(m_i) = \langle\,\psi\,|P_i|\,\psi\,\rangle, \tag{4.10}$$

onde P_i é o projetor no auto-espaço de A com respeito ao auto-valor m_i (i.e., $P_i = |\,m_i\,\rangle\langle\,m_i\,|$, sendo $|\,m_i\,\rangle$ o auto-vetor associado ao auto-valor m_i).

O Postulado 3 descreve um tipo especial de medições em sistemas quânticos chamadas *medidas projetivas*, o que é suficiente para nossos propósitos. Uma versão mais geral desse postulado pode ser encontrada em [NC00].

A medição tem um efeito colateral, que consiste no *colapso* do estado do sistema (representado pela função de onda) ao estado representado pelo auto-vetor $|\,\psi'\,\rangle$ associado ao auto-valor obtido na medição.

Para o caso de um qubit pode ser construído um observável A, o qual é chamado de *observável padrão*, usando a representação diagonal (Definição B.23) e tomando como auto-vetores $|\,0\,\rangle$ e $|\,1\,\rangle$, com os respectivos auto-valores 0 e 1; assim:

$$A = 0|\,0\,\rangle\langle\,0\,| + 1|\,1\,\rangle\langle\,1\,| = 0\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} + 1\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} = \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}. \tag{4.11}$$

Para o estado $|\,\psi\,\rangle = \alpha|\,0\,\rangle + \beta|\,1\,\rangle$, numa medição com respeito a A obtém-se o valor 0 como saída com probabilidade $Pr(0) = \langle\,\psi\,||\,0\,\rangle\langle\,0\,||\,\psi\,\rangle = |\alpha|^2$, ou valor 1 com probabilidade $Pr(1) = \langle\,\psi\,||\,1\,\rangle\langle\,1\,||\,\psi\,\rangle = |\beta|^2$. O estado do sistema depois da medição colapsa ao estado $|\,0\,\rangle$ ou $|\,1\,\rangle$, dependendo de se o valor obtido na medição é 0 ou 1, respectivamente.

Diferentemente da mecânica clássica, na mecânica quântica a observação ou medição de um sistema influi nele de maneira radical, fazendo-o mudar de estado de uma maneira não determinística, perdendo irreversivelmente o estado anterior à medição. É esta uma das grandes diferenças entre a mecânica clássica e a mecânica quântica: a medição é o ato que introduz indeterminismo e probabilidade na teoria quântica.

Postulado 4. O espaço de estados de um *sistema físico composto* é o produto tensorial (Definição B.27) dos espaços de estados dos sistemas físicos que o compõem. Além disso, se se tem os sistemas numerados de 1 a n nos estados representados por $|\,\psi_1\,\rangle$ a $|\,\psi_n\,\rangle$, respectivamente, então, o estado do sistema composto por esses n subsistemas é representado por $|\,\psi_1\,\rangle \otimes |\,\psi_2\,\rangle \otimes \ldots \otimes |\,\psi_n\,\rangle$ (Definição B.28).

O espaço de estados bidimensional associado a um qubit é comumente denotado por H_2. De acordo com o postulado 4, um registro composto por dois qubits, usualmente chamado de *2-qubit*, tem associado um espaço de estados $H_4 = H_2 \otimes H_2$ (um espaço vetorial de dimensão 4). A base padrão de tal espaço é $\{|\,0,0\,\rangle, |\,0,1\,\rangle, |\,1,0\,\rangle, |\,1,1\,\rangle\}$, onde $|\,i,j\,\rangle$ denota o produto tensorial $|\,i\,\rangle \otimes |\,j\,\rangle$. O estado geral de um 2-qubit é da forma:

$$|\,\psi\,\rangle = \alpha_0|\,0,0\,\rangle + \alpha_1|\,0,1\,\rangle + \alpha_2|\,1,0\,\rangle + \alpha_3|\,1,1\,\rangle, \tag{4.12}$$

e a condição de normalização é:

$$|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 = 1. \tag{4.13}$$

De maneira mais geral, para um registro composto por n qubits, chamado de *n-qubit*, o espaço de estados é $H_{2^n} = H_2^{\otimes n}$ (onde $H_2^{\otimes n}$ representa o produto tensorial $\overbrace{H_2 \otimes \ldots \otimes H_2}^{n-vezes}$. O espaço H_{2^n} é de dimensão 2^n). Para uma base $B = \{|\,\psi_i\,\rangle | 0 \leq i < 2^n\}$ de H_{2^n}, o estado geral de um n-qubit é da forma:

$$|\,\psi\,\rangle = \sum_{i=0}^{2^n-1} \alpha_i |\,i\,\rangle, \tag{4.14}$$

e a condição de normalização é:

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \tag{4.15}$$

Na mecânica quântica os sistemas compostos têm uma propriedade particular: alguns estados de sistemas compostos não podem ser expressos através do produto tensorial dos estados dos sistemas que o compõem, esses estados são chamados *estados emaranhados*. Um exemplo de estado emaranhado de um 2-qubit é o seguinte:

$$|\,\psi\,\rangle = \frac{1}{\sqrt{2}}(|\,0,0\,\rangle + |\,1,1\,\rangle). \tag{4.16}$$

Se se supuser que existem qubits $|\,\varphi\,\rangle$ e $|\,\gamma\,\rangle$ com estados $|\,\varphi\,\rangle = \alpha_0|\,0\,\rangle + \alpha_1|\,1\,\rangle$ e $|\,\gamma\,\rangle = \beta_0|\,0\,\rangle + \beta_1|\,1\,\rangle$, tais que $|\,\psi\,\rangle = |\,\varphi\,\rangle \otimes |\,\gamma\,\rangle$, então $|\,\psi\,\rangle = \alpha_0\beta_0|\,0,0\,\rangle + \alpha_0\beta_1|\,0,1\,\rangle + \alpha_1\beta_0|\,1,0\,\rangle + \alpha_1\beta_1|\,1,1\,\rangle$, logo $\alpha_0\beta_0 = \alpha_1\beta_1 = 1/\sqrt{2}$ e $\alpha_0\beta_1 = \alpha_1\beta_0 = 0$; mas isso não é possível, pois $\alpha_0\beta_1 = 0$ implica que $\alpha_0 = 0$ ou $\beta_1 = 0$, logo, no primeiro caso ($\alpha_0 = 0$) obtém-se que $\alpha_0\beta_0 = 0$, o que contradiz que $\alpha_0\beta_0 = 1/\sqrt{2}$, e no segundo caso ($\beta_1 = 0$) obtém-se que $\alpha_1\beta_1 = 0$, o que contradiz que $\alpha_1\beta_1 = 1/\sqrt{2}$. Portanto, $|\,\psi\,\rangle$ não pode ser expresso como produto tensorial de dois qubits.

Note que no estado $|\,\psi\,\rangle$, se se realizar uma medição sobre qualquer um dos qubits, baseado no observável padrão descrito acima, obtém-se o valor 0 com probabilidade $1/2$ ou o valor 1 também com probabilidade $1/2$. Porém, após a realização da medição sobre um dos dois qubits o estado do sistema composto colapsa a $|\,0,0\,\rangle$ (se

o valor 0 for obtido na medição) ou a $|1,1\rangle$ (se o valor 1 for obtido na medição). Portanto, ao medir o outro qubit vai-se obter o valor 0 com probabilidade 1, ou o valor 1 com probabilidade 1, respectivamente. i.e., a medição do estado de um dos qubits determina o estado do outro qubit, e o mais assombroso é que isso de fato acontece, mesmo em situações em que os dois qubits estão distantes um do outro. Os estados emaranhados não têm equivalente na física clássica, e sua existência é uma das características mais contra-intuitivas da mecânica quântica. Sua relevância na definição de algoritmos quânticos eficientes não é ainda clara.

Os postulados 1 a 4 determinam completamente como deve ser descrito um sistema físico microscópico, com evolução temporal discreta, o que é suficiente para a descrição dos modelos de MTQs e CQs.

Embora a mecânica quântica tenha se mostrado experimentalmente válida, ainda hoje, oitenta anos após o surgimento dessa teoria, não existe um consenso entre os físicos de como interpretá-la para explicar de maneira unificada o que acontece a nível microscópico. A partir da primeira formulação da mecânica quântica surgiram várias interpretações dessa teoria, interpretações essas com pontos de divergência muito fortes, e algumas delas com elementos bastante discutíveis.

Entre as diferentes interpretações da mecânica quântica a de maior aceitação é a chamada *interpretação de Copenhagen* (embora recentemente outras interpretações estejam ganhando aceitação, segundo [Teg98]). Tal interpretação, contudo, não consiste em um conjunto de idéias únicas e claramente definidas, mostrando-se como um denominador comum de vários pontos de vista divergentes em múltiplos aspectos, mas baseada principalmente na noção de *complementaridade* de Niels Bohr.

Como descrito por Max Jammer em [Jam74, p. 104], mesmo que não seja fácil se definir a noção de complementaridade sugerida por Bohr, as idéias expostas por ele podem ser descritas indicando quando uma teoria pode ser interpretada via complementariedade. Uma teoria T admite uma interpretação via complementaridade se as seguintes condições são satisfeitas:

1. T contém (pelo menos) duas descrições D_1 e D_2 de seus aspectos substanciais.

2. D_1 e D_2 se referem ao mesmo universo de discurso U (no caso de Bohr, a

microfísica).

3. Nem D_1 nem D_2, isoladamente, dão conta exaustiva dos fenômenos de U.

4. D_1 e D_2 são mutuamente excludentes, no sentido em que sua combinação, numa descrição única, leva a contradições lógicas.

No caso da mecânica quântica, D_1 e D_2 podem ser tomadas como as descrições que interpretam os objetos microfísicos como ondas e como partículas, respectivamente, levando à chamada *dualidade onda-partícula*.

A noção de interpretação via complementaridade sugere, naturalmente, a possibilidade de se utilizar alguma lógica paraconsistente como lógica subjacente a teorias que admitam tal tipo de interpretação, em particular à mecânica quântica. Essa é precisamente a idéia que trabalham Newton da Costa e Decio Krause em [dK04] (cf. também [dK06]), onde eles chamam de *c-theories* as teorias com estas características. Ainda de uma maneira mais específica, a noção de interpretação via complementaridade sugere que se lance mão da idéia de semântica de traduções possíveis (já mencionada no final do Capítulo 2): aspectos complementares poderiam ser pensados como diferentes traduções do mesmo estado de coisas.

Ainda em relação à interpretação de Copenhagen, é importante ressaltar que nessa interpretação é possível afirmar que existem grandezas físicas que não têm valores determinados em certos estados quânticos. É só através da medição que se determinam tais grandezas. Portanto, a função de onda só pode ser considerada como uma ferramenta para calcular as probabilidades dos valores a ser obtidos nas medições.

Outro tipo de interpretação da mecânica quântica conhecida como *interpretação de múltiplos mundos*, da qual existem varias versões (cf. [Vai02]), consiste de variações, reinterpretações ou aprimoramentos da *interpretação de estados relativos* proposta por Hugh Everett em [Eve57]. Na interpretação de múltiplos mundos a função de onda nunca colapsa. Antes da medição todos os possíveis estados coexistem no 'mundo' descrito pela função de onda $|\,\psi\,\rangle$. A medição divide o mundo em múltiplos ramos ou múltiplos mundos igualmente reais, cada um dos quais corresponde a um componente clássico da superposição $|\,\psi\,\rangle$. A interpretação de múltiplos mundos

baseia-se no postulado de que todo sistema isolado evolui deterministicamente, de acordo com a equação de Schrödinger, e portanto, como o universo é um sistema isolado, deve necessariamente evoluir de acordo com tal equação. O colapso da função de onda violaria tal postulado.

Um outro tipo de interpretação da mecânica quântica, que nos interessa aqui mencionar, é a chamada *interpretação de variáveis ocultas*[6] (também referida como *teorias de variáveis ocultas*, pois algumas dessas 'interpretações' não somente interpretam a mecânica quântica mas também introduzem novos elementos na formalização). Uma característica dessas teorias é a tentativa de formalizar a idéia, já sugerida por Einstein, de que a natureza age de maneira totalmente determinística e portanto a mecânica quântica só pode ser vista como uma aproximação probabilística à realidade física, sendo o indeterminismo o produto do desconhecimento dos valores de certas propriedades ou 'variáveis ocultas'. Neste tipo de interpretação também não há colapso da função de onda, pois o estado do sistema é sempre totalmente determinado pelos valores das variáveis ocultas. A chamada 'teoria de de Broglie-Bohm' é talvez a mais representativa dentro deste tipo de interpretações.[7]

4.2 Modelos de computação quântica

Como mencionado acima, nesta seção vamos apresentar os modelos de MTQs e CQs, modelos estes que podem ser considerados como os modelos 'padrões' da computação quântica. O modelo de MTQs será apresentado seguindo [Oza98] e [ON00], e o modelo de CQs será apresentado seguindo [NC00] e [Gru99].

4.2.1 Máquinas de Turing quânticas

O modelo de MTQ é uma generalização natural do modelo de MTs, definido mediante a suposição de que os elementos da máquina correspondem a observáveis de um

[6] Em princípio, as variáveis ocultas introduzidas no cálculo de anéis de polinômios (Seção 3.2) são diferentes das variáveis ocultas da interpretação da mecânica quântica. Contudo, o uso de variáveis ocultas no cálculo de anéis de polinômios foi inspirado nas teorias quânticas de variáveis ocultas. De fato, essas variáveis foram inicialmente chamadas de *variáveis quânticas* em [Car05].

[7] Para maior informação a respeito de teorias de variáveis ocultas veja [Gen05].

sistema físico microscópico.

Na generalização vamos considerar $Q = \{q_1, \ldots, q_n\}$ como sendo o conjunto de estados da máquina e $\Sigma = \{s_1, \ldots, s_m\}$ como sendo o alfabeto de entrada/saída, vamos definir uma *configuração de fita* T como uma seqüência infinita de símbolos em Σ, que representa o conteúdo da fita num determinado instante de tempo (portanto deve conter um número finito de símbolos diferentes de s_1, que por convenção é o símbolo vazio), e vamos denotar por $T(j)$ o símbolo na posição j da fita.

Pelo Postulado 1, o estado de uma MTQ $\mathcal{M}$, no instante de tempo t, é representado por um vetor unitário $|\,\psi(t)\,\rangle$, no espaço de Hilbert $\mathcal{H}(Q, \Sigma)$ gerado pelo espaço de configurações $C(Q, \Sigma) = Q \times \Sigma^{\#} \times \mathbb{Z}$, onde $\Sigma^{\#}$ é o conjunto de todas as possíveis configurações de fita T. Cada elemento $|\,q_i, T, n\,\rangle$ de $C(Q, \Sigma)$ representa uma possível configuração de uma MT clássica, sendo q_i o estado atual, T o conteúdo da fita e n a posição atual da máquina. Portanto, o estado de uma MTQ pode ser considerado como uma superposição de configurações de uma MT clássica.

Pelo Postulado 3 e a discussão subseqüente, os observáveis correspondentes ao estado atual da máquina, à posição atual da máquina e ao símbolo na posição j da fita são representados, respectivamente, pelos operadores:

$$\hat{q} = \sum_{i=1}^{n} i|\,q_i\,\rangle\langle\,q_i\,|,\ \hat{p} = \sum_{p\in\mathbb{Z}} p|\,p\,\rangle\langle\,p\,|,\ \hat{T}(j) = \sum_{i=1}^{m} i|\,s_i\,\rangle\langle\,s_i\,|. \tag{4.17}$$

Uma computação numa MTQ $\mathcal{M}$ começa no instante de tempo $t = 0$; nesse instante, $\mathcal{M}$ é preparada (através de alguma ação experimental) num estado inicial, que segundo o Postulado 4 é representado pelo produto tensorial:

$$|\,\psi(0)\,\rangle = |\,q_0, T_{inicial}, 0\,\rangle. \tag{4.18}$$

A computação é realizada em passos de duração fixa τ. Portanto, como estabelecido pelo Postulado 2, a evolução de $\mathcal{M}$ em cada passo de computação é descrita por um operador unitário U em $\mathcal{H}(Q, \Sigma)$. Assim, depois de n passos de computação o estado de $\mathcal{M}$ é descrito por:

$$|\,\psi(n\tau)\,\rangle = U^n|\,\psi(0)\,\rangle, \tag{4.19}$$

onde U^n denota a aplicação n vezes do operador U. Esse operador, como indicado

acima, pode ser representado como uma matriz M, onde o elemento na fila i e coluna j representa a amplitude de transição da configuração C_j à configuração C_i.

As transições de configurações no processo de computação de uma MTQ, de maneira similar a como se faz para as MTs probabilísticas, podem ser representadas por meio de uma árvore (Figura 4.1), onde os nós correspondem a configurações (de MT clássicas) C_i, e as aresta entre os nós C_i e C_j são assinaladas com α_{ij}, indicando a amplitude de probabilidade de passar da configuração C_i à configuração C_j. A raiz da árvore corresponde à configuração inicial $C_0 = (q_0, T_{inicial}, 0)$.

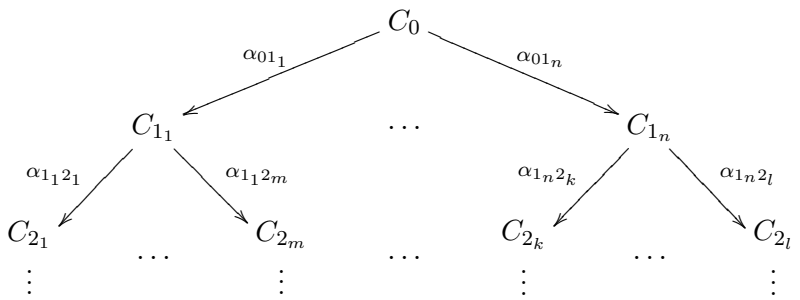

Figura 4.1: Árvore de computação numa MTQ

Diferentemente de uma MT probabilística, onde cada computação particular percorre somente um caminho da árvore (de acordo com as probabilidades dadas), nas MTQs a computação percorre todos os possíveis caminhos simultaneamente. Como o número de nós vai crescendo exponencialmente em cada nível da árvore, isso significa que a MTQ pode estar, num momento da computação, numa superposição de um número exponencial de configurações, de acordo com o número de passos da computação. Dessa maneira, as MTQs expressam um tipo de paralelismo no processo de computação, o que é chamado de *paralelismo quântico*. Porém, para se obter o resultado da computação deve se realizar uma medição ao final do processo e, devido ao Postulado 3, somente uma configuração da superposição é obtida de maneira probabilística, perdendo irreversivelmente as outras configurações.[8] Por-

[8]Para determinar quando termina o processo de computação numa MTQ, para realizar a medição do resultado, é acrescentado um qubit de *indicador de parada* à configuração da máquina; esse

tanto, o paralelismo deve ser aproveitado, através da interferência quântica, antes da computação terminar, de maneira a aumentar a amplitude de probabilidade dos resultados desejados e diminuir a dos não desejados. Alguns cominhos da arvore podem desaparecer devido à interferência destrutiva.

É importante indicar que a interpretação das computações em MTQs, como proposta em [Deu85], é realizada através da interpretação de múltiplos mundos da mecânica quântica. Esse tipo de interpretação permite pensar os estados superpostos das MTQs como a coexistência de configurações de MTs clássicas, e o paralelismo quântico como a execução simultânea das diferentes possíveis computações. Em [Deu85], Deutsch chega a afirma que não é possível explicar a computação quântica através de qualquer outra interpretação da mecânica quântica sem se perder totalmente o poder explicativo. No Capítulo 5 (Seção 5.2) vamos apresentar alguns argumentos contra esta afirmação de Deutsch.

Além de substituir os elementos da MT por observáveis de um sistema microscópico, Deutsch, na sua definição do modelo de MTQs, exige condições de 'finitude' similares às impostas por Turing para suas máquinas abstratas. Assim, as operações de uma MTQ estão restritas pelas seguintes condições:

1. Somente uma parte finita do sistema pode mudar durante cada passo de computação.

2. A mudança deve depender somente do estado de um subsistema finito.

3. As regras que especificam a mudança devem ser dadas finitamente.

Para satisfazer estas condições, os elementos da matriz que representa o operador de evolução U devem ser da forma:

$$\langle\, q', T', p' \,|U|\, q, T, p \,\rangle = \begin{cases} \delta\,(q, T(p), q', T'(p), 1) & \text{se } p' = p + 1 \\ \delta\,(q, T(p), q', T'(p), 0) & \text{se } p' = p \\ \delta\,(q, T(p), q', T'(p), -1) & \text{se } p' = p - 1 \end{cases} \tag{4.20}$$

qubit deve ser modificado adequadamente pelo operador de evolução e é medido em todo instante de tempo, até se obter o valor 1 como saída, o que indica que a computação terminou (cf. [Oza98]).

sempre que $T'(m) = T(m)$ para todo $m \neq p$, e $\langle\, q', T', p' \,|U|\, q, T, p \,\rangle = 0$ em outro caso. A função δ é da forma δ: $Q \times \Sigma \times Q \times \Sigma \times \{-1, 0, 1\} \to \tilde{\mathcal{C}}$, e é chamada *função de transição local*. Os valores em $\{-1, 0, 1\}$ representam os movimentos da cabeça de leitura/escrita (à esquerda, não movimento e à direita, respectivamente) e $\tilde{\mathcal{C}}$ representa o conjunto dos números complexos computáveis. Portanto, $\delta(q, s, q', s', d') = c$ é interpretado assim: se $\mathcal{M}$ está no estado q, lendo o símbolo s, então, com uma amplitude de probabilidade c, a máquina $\mathcal{M}$ escreve o símbolo s', faz o movimento d' e passa ao estado q'.

Para assegurar que o operador U seja unitário, e portanto $\mathcal{M}$ seja reversível, δ deve validar as seguintes condições:

1. Para toda $(q, s) \in Q \times \Sigma$:

$$\sum_{q', s', d'} |\delta(q, s, q', s', d')|^2 = 1. \tag{4.21}$$

2. Para toda $(q, s), (t, r) \in Q \times \Sigma$, com $(q, s) \neq (t, r)$:

$$\sum_{q', s', d'} \delta(q, s, q', s', d') \cdot \delta(t, r, q', s', d') = 0. \tag{4.22}$$

3. Para toda $(q, s, r'), (t, r, s') \in Q \times \Sigma^2$:

$$\sum_{q' \in Q, d' \in \{0,1\}} \delta(t, r, q', s', d' - 1) \cdot \delta(q, s, q', r', d') = 0. \tag{4.23}$$

4. Para toda $(q, s, r'), (t, r, s') \in Q \times \Sigma^2$:

$$\sum_{q' \in Q} \delta(t, r, q', s', -1) \cdot \delta(q, s, q', r', 1) = 0. \tag{4.24}$$

O operador de evolução U é totalmente determinado pela função de transição local, conseqüentemente, as MTQs podem ser definidas por meio dessa função de transição sem fazer referência ao operador de evolução como noção primitiva.

4.2.2 Circuitos quânticos

O modelo de CQs é uma generalização, também natural, do modelo de circuitos lógicos clássicos (ou circuitos booleanos), definido mediante a suposição de que o circuito é implementado através de um sistema microscópico e, portanto, condicionado pelas leis da mecânica quântica.

Um CQ é então uma coleção finita de *variáveis de entrada* e *portas* conectadas de maneira acíclica e direcionada, onde as variáveis de entrada, em lugar de bits clássicos, são qubits (de acordo com o Postulado 1), e onde as portas (chamadas *portas quânticas*), em lugar de realizar operações booleanas, realizam transformações unitárias (de acordo com o Postulado 2).

Devido a propriedades da mecânica quântica, diferentemente dos circuitos lógicos clássicos, onde as operações de *fanin* e *fanout* são permitidas,[9] nos CQs estas operações não são possíveis. A primeira é impossível por se tratar de uma operação não reversível, o que vai em contradição com a reversibilidade da evolução dos sistemas quânticos (enquanto não são realizadas medições), e a segunda é impedida pelo *Teorema de Não-clonagem*:

Teorema 4.1. *Não existe uma transformação unitária que copie ou clone qubits arbitrários; i.e., não existe transformação unitária U tal que $U|\,a,0\,\rangle = |\,a,a\,\rangle$ para qualquer qubit $|\,a\,\rangle$.*

Demonstração. Suponha que exista uma transformação unitária U tal que $U|\,a,0\,\rangle = |\,a,a\,\rangle$ para todo $|\,a\,\rangle$. Sejam $|\,a\,\rangle$ e $|\,b\,\rangle$ dois estados ortogonais (portanto podem ser tomados como base para o espaço dos qubits). Pela definição de U, tem-se que $U|\,a,0\,\rangle = |\,a,a\,\rangle$ e $U|\,b,0\,\rangle = |\,b,b\,\rangle$. Considere $|\,c\,\rangle = \frac{1}{\sqrt{2}}(|\,a\,\rangle + |\,b\,\rangle)$; pela linearidade de U:

$$\begin{aligned} U|\,c,0\,\rangle &= \frac{1}{\sqrt{2}}(U|\,a,0\,\rangle + U|\,b,0\,\rangle) \\ &= \frac{1}{\sqrt{2}}(|\,a,a\,\rangle + |\,b,b\,\rangle). \end{aligned} \tag{4.25}$$

[9] A operação de *fanin* consiste na união de duas conexões em uma só, contendo a disjunção do conteúdo das duas conexões; enquanto a operação de *fanout* consiste na divisão de uma conexão em várias, cada uma contendo uma cópia do conteúdo da conexão inicial.

Mas, por outro lado, pela definição de U tem-se que:

$$U|\,c,0\,\rangle = |\,c,c\,\rangle = \frac{1}{2}(|\,a,a\,\rangle + |\,a,b\,\rangle + |\,b,a\,\rangle + |\,b,b\,\rangle), \tag{4.26}$$

o que contradiz o resultado obtido acima. □

Nos CQs o número de saídas das portas deve ser igual ao número de entradas, de forma a assegurar a reversibilidade das operações. Porém na saída do circuito podem ser descartados alguns valores, de maneira que o resultado da computação corresponde aos valores obtidos na medição de um subconjunto específico de qubits de saída.

As medições num CQ são realizadas ao final do circuíto.[10] Tais medições são regidas pelo Postulado 3, e portanto obtém-se um valor de maneira probabilística, dependendo das amplitudes de probabilidade do estado do sistema ao momento de realizar a medição; nesse momento, como consequência desse postulado, o estado do sistema colapsa ao autovetor correspondente ao autovalor obtido na medição.

De maneira análoga ao que acontece nas MTQs, nos CQs os estados superpostos e a linearidade dos operadores unitários permitem realizar processamento em paralelo, o que corresponde ao paralelismo quântico neste modelo (que também é explicado através da interpretação de múltiplos mundos da mecânica quântica). Esse paralelismo, tal como no caso das MTQs, deve ser aproveitado antes de se realizar a medição e perder-se irreversivelmente a superposição de estados.

Como já foi mencionado no começo deste capítulo, Yao demonstrou em [Yao93] que qualquer função computável em tempo polinomial numa MTQ pode ser computada por um CQ de tamanho polinomial (usando somente portas quânticas 'elementares', que correspondem a um conjunto de portas quânticas 'universais', o que será descrito adiante). Esse resultado legitima a definição de algoritmos quânticos através de CQs, ao invés de MTQs, o que parece facilitar o trabalho. Na próxima seção vamos descrever alguns algoritmos, definidos através de CQs, que ilustrarão

[10] Medições em passos intermédios dos CQs são às vezes realizadas, e os resultados são usados para controlar outras portas. Contudo, estas medições podem ser substituídas por portas quânticas controladas (que serão descritas adiante), postergando a medição para o final do circuito e obtendo o mesmo resultado (cf. [NC00, p. 186]).

como funciona o paralelismo nesse modelo; mas antes disso vamos apresentar algumas portas quânticas que são de uso comum e vamos descrever um conjunto de portas quânticas universais.

As matrizes de Pauli (Equação 4.4) e o operador H (Equação 4.8) são exemplos de portas quânticas que operam sobre um qubit. A matriz X de Pauli é chamada de *porta de negação*, pois inverte os valores base $|\,0\,\rangle$ e $|\,1\,\rangle$. O operador H é chamado de *porta de Hadamard* e é comumente usado no começo do circuito para gerar *superposições perfeitas* de estados (i.e. superposições onde todas as amplitudes de probabilidade são iguais), para depois, aproveitando a linearidade dos operadores quânticos, realizar processamento em paralelo sobre todos os elementos da base.

Outras portas comuns, que operam sobre um qubit, são as chamadas *porta de fase* (denotada por S) e *porta* $\pi/8$ (denotada por T)[11]:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix},\ T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{4.27}$$

Um tipo especial de portas quânticas são as chamadas *portas controladas*; nessas portas tem-se um qubit de controle que determina se a operação sobre os outros qubits é ou não realizada. Um exemplo destas portas é a chamada *porta de negação controlada*:

$$X_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.28}$$

Esta porta opera sobre um 2-qubit, realizando as transformações:

$$\begin{aligned} X_c{:}|\,0,0\,\rangle &\mapsto |\,0,0\,\rangle \\ |\,0,1\,\rangle &\mapsto |\,0,1\,\rangle \\ |\,1,0\,\rangle &\mapsto |\,1,1\,\rangle \\ |\,1,1\,\rangle &\mapsto |\,1,0\,\rangle. \end{aligned} \tag{4.29}$$

[11] Esta porta é denominada $\pi/8$, embora na definição apareça o exponente $\pi/4$, devido a que anteriormente esta porta era expressa por uma matriz equivalente onde aparecia o exponente $\pi/8$ (cf. [NC00, p. 174]).

Note que o segundo qubit é negado somente se o primeiro qubit é $|\,1\,\rangle$ (se o primeiro qubit é $|\,0\,\rangle$ não se realiza nenhuma operação). A representação gráfica da porta X_c num CQ é:

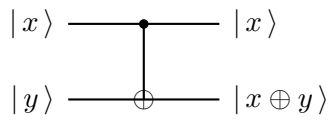

Figura 4.2: Representação gráfica da porta X_c

Em geral, a representação gráfica das portas U-controladas é:

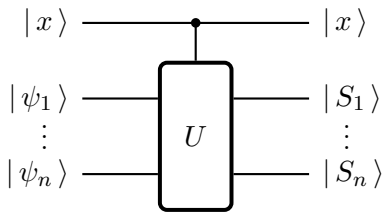

Figura 4.3: Representação gráfica de uma porta U-controlada

De maneira similar à existência de conjuntos de *portas clássicas universais*, os quais são conjuntos de portas booleanas através das quais qualquer função $f\colon \{0,1\}^n \to \{0,1\}^m$ pode ser calculada, existem também conjuntos de *portas quânticas universais*, os quais são conjuntos de portas quânticas tais que qualquer transformação unitária pode ser aproximada, com precisão arbitrária, por um circuito quântico contendo somente portas desse conjunto. No caso dos circuitos quânticos é exigida somente uma 'aproximação' com precisão arbitrária, pelo fato do conjunto de transformações ser continuo. Existem múltiplos conjuntos de portas quânticas universais, um deles consiste nas portas H, S, X_c e $\pi/8$ apresentadas acima (cf. [NC00, Seção 4.5.2]). Outro conjunto´de portas universais consiste da porta de Toffoli T (que é uma negação duplamente controlada) e da porta H, este conjunto é interessante por mostrar, de certa maneira, o mínimo que deve ser acrescentado à computação clássica para obter todas as potencialidades da computação quântica

(cf. [Aha03]).

4.3 Algoritmos quânticos

Vamos agora apresentar alguns algoritmos quânticos: o algoritmo para resolver o chamado *problema de Deutsch* será apresentado em detalhe, para ilustrar como são realizados os cálculos nos algoritmos quânticos e como funciona o paralelismo e a interferência quântica. O algoritmo para resolver o chamado *problema de Deutsch-Jozsa* e os algoritmos de Daniel Simon (para calcular o período de uma função), de Peter Shor (para fatorar números inteiros em seus componentes primos) e de Lov Grover (para procurar dados numa base de dados desordenada) serão brevemente descritos.

4.3.1 Algoritmo para o problema de Deutsch

Dada uma função arbitraria f: $\{0,1\} \rightarrow \{0,1\}$ e um oráculo[12] U_f que realiza a operação $U_f|\, x, y\,\rangle = |\, x, y \oplus f(x)\,\rangle$ (onde $\oplus$ representa adição módulo 2), o problema de Deutsch consiste em determinar, usando U_f uma única vez, se f é *constante* ($f(0) = f(1)$) ou *balanceada* ($f(0) \neq f(1)$). Classicamente parece inevitável usar o oráculo que computa f duas vezes,[13] para o valor 0 e para o valor 1, para depois comparar os valores e determinar se f é constante ou balanceada. Com um CQ, aproveitando o paralelismo intrínseco a este modelo, é possível obter $f(0)$ e $f(1)$ mediante uma única aplicação de U_f, produzindo um estado superposto, com ambos valores da função, que pode depois ser modificado de maneira a determinar se f é constante ou balanceada.

Uma solução probabilística para este problema foi proposta por Deutsch em [Deu85] e a primeira solução determinística foi proposta por Cleve, Ekert, Macchiavello e Mosca em [CEMM98]. A figura a seguir representa um CQ que resolve,

[12] Um oráculo é um dispositivo que trabalha como uma caixa preta, computando uma função sem especificar de que maneira.

[13] Em [Cal07], Christian Calude propõe uma de-quantização da solução do problema de Deutsch, mostrando como esse problema pode ser resolvido classicamente. Porém essa proposta não é aceita em geral.

de maneira determinística, o problema de Deutsch:[14]

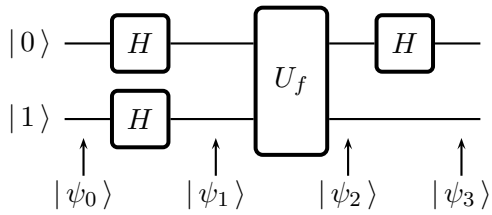

Figura 4.4: Circuito quântico para resolver o problema de Deutsch

Os kets $|\,\psi_i\,\rangle$ na parte baixa do desenho serão usados para explicar como funciona o circuito.

No começo do circuito, as entradas são representadas pelo produto tensorial:

$$|\,\psi_0\,\rangle = |\,0,1\,\rangle. \tag{4.30}$$

Depois de aplicar as primeiras duas portas de Hadamard obtém-se o estado superposto:

$$\begin{aligned} |\,\psi_1\,\rangle &= H|\,0\,\rangle \otimes H|\,1\,\rangle \\ &= \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle + |\,1\,\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) \\ &= \frac{1}{2}\left(|\,0,0\,\rangle - |\,0,1\,\rangle + |\,1,0\,\rangle - |\,1,1\,\rangle\right). \end{aligned} \tag{4.31}$$

A porta U_f, aplicada sobre o estado $|\,\psi_1\,\rangle$, computa a função f em paralelo produzindo:

$$\begin{aligned} |\,\psi_2\,\rangle &= U_f\left(\frac{1}{2}\left(|\,0,0\,\rangle - |\,0,1\,\rangle + |\,1,0\,\rangle - |\,1,1\,\rangle\right)\right) \\ &= \frac{1}{2}\left(|\,0,0\oplus f(0)\,\rangle - |\,0,1\oplus f(0)\,\rangle + |\,1,0\oplus f(1)\,\rangle - |\,1,1\oplus f(1)\,\rangle\right) \\ &= \frac{1}{2}\left((-1)^{f(0)}(|\,0\,\rangle \otimes (|\,0\,\rangle - |\,1\,\rangle)) + (-1)^{f(1)}(|\,1\,\rangle \otimes (|\,0\,\rangle - |\,1\,\rangle))\right) \\ &= \begin{cases} \pm\left(\frac{1}{\sqrt{2}}(|\,0\,\rangle + |\,1\,\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) & \text{se } f(0) = f(1), \\ \pm\left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) & \text{se } f(0) \neq f(1). \end{cases} \end{aligned} \tag{4.32}$$

[14]Este CQ foi tomado de [NC00, pp. 33], e corresponde a uma pequena variação do circuito proposto em [CEMM98].

A porta de Hadamard, no final do circuito, produz uma interferência de tal maneira que gera o estado:

$$|\,\psi_3\,\rangle = \begin{cases} \pm|\,0\,\rangle \otimes \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) & \text{se } f(0) = f(1), \\ \pm|\,1\,\rangle \otimes \left(\frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)\right) & \text{se } f(0) \neq f(1). \end{cases} \tag{4.33}$$

Portanto, ao fazer uma medição sobre o primeiro qubit de $|\,\psi_3\,\rangle$ obtém-se 0 (com probabilidade 1) se f é constante, ou obtém-se 1 (com probabilidade 1) se f é balanceada. Note que o algoritmo é determinístico, e efetua uma única aplicação de U_f.

4.3.2 Algoritmo para o problema de Deutsch-Jozsa

O problema de Deutsch-Jozsa consiste na generalização do problema de Deutsch para funções f: $\{0,1\}^n \to \{0,1\}$, onde f é balanceada se $f(x) = 0$ para a metade dos valores $x \in \{0,1\}^n$ e $f(x) = 1$ para a outra metade dos valores, e onde se parte da premissa de que f é constante ou balanceada (cf. [DJ92]). Um CQ para resolver este problema, representado na figura a seguir, é uma generalização natural do CQ anterior:

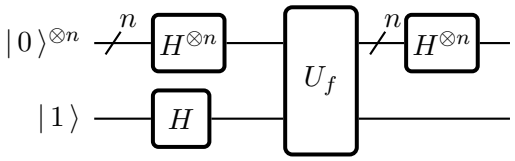

Figura 4.5: Circuito quântico para resolver o problema de Deutsch-Jozsa

O registro de entrada é agora constituído por $n+1$ qubits e toma o valor $|\,0\,\rangle^{\otimes n} \otimes |\,1\,\rangle$. Novas portas de Hadamard são adicionadas ao circuito nas entradas e saídas do oráculo U_f, o qual opera sobre os $n+1$ qubits calculando a função $U_f(|\,x_1,\ldots,x_n,y\,\rangle) = |\,x_1,\ldots,x_n,y \oplus f(x_1,\ldots,x_n)\,\rangle$. Neste caso, quando são medidos os primeiros n qubits no final da computação, se todos os valores obtidos são 0 então f é constante (com probabilidade 1), em outro caso f é balanceada (com probabilidade 1).[15]

[15] Os cálculos são apresentados em detalhe em [NC00].

Note que a solução quântica do problema de Deutsch-Jozsa efetua uma única aplicação de U_f, e determina se f é constante ou balanceada de maneira determinística. Classicamente parece inevitável que, no pior caso, o oráculo que computa f deve ser usado $2^{n-1}+1$ vezes para determinar se f é constante ou balanceada (deve-se calcular f com diferentes entradas até encontrar dos valores diferentes ou até calcular a metade dos valores mais um). Como a complexidade de um algoritmo é medida pelos recursos usados no pior caso, a solução clássica determinística para determinar se f é balanceada ou não tem complexidade exponencial, enquanto o algoritmo quântico para solucionar o mesmo problema tem complexidade polinomial. Porém, pode se mostrar que o problema de Deutsch-Jozsa pode ser resolvido probabilisticamente (com probabilidade de erro pequena) em tempo polinomial em um modelo de computação clássico.

4.3.3 Algoritmo de Simon

Dada uma função $f\colon \{0,1\}^n \to \{0,1\}^n$ tal que, para todo $x \neq x'$, $f(x) = f(x')$ se e somente se $x' = x \oplus s$ (onde $\oplus$ denota a operação, bit a bit, do ou-exclusivo), o problema resolvido por Simon (cf. [Sim97]) consiste em determinar o período s de f. Para este problema não existe uma solução clássica eficiente (i.e. com tempo polinomial).

Em sua solução Simon define um CQ que realiza um procedimento ao qual chama *Fourier-twice* (e que em outros lugares é denominado *Hadamard-twice*). Esse circuito recebe como entrada um registro de $2n$-qubits inicializado em $|\, 0^n, 0^n \,\rangle$, aplica transformações de Hadamard nos primeiros n-qubits, em seguida aplica a transformação U_f (que funciona como um oráculo similar ao do algoritmo anterior) e por último aplica de novo transformações de Hadamard sobre os primeiros n-qubits. Analisando esse circuito fica claro que, se a função computada por U_f satisfaz as condições acima mencionadas, ao medir os primeiros n-qubits obtêm-se um valor y_i tal que $y_i \cdot s \equiv 0 \pmod 2$. Para calcular s o procedimento *Fourier-twice* é executado um número de vezes $O(n)$, produzindo, com uma alta probabilidade, uma quantidade k de valores y_i, linearmente independentes (se vistos como vetores), os quais são suficientes para determinar s resolvendo o sistema de equações lineares $y_i \cdot s \equiv 0 \pmod 2$, $1 \leq i \leq k$.

A validação da independência linear dos y_i e a solução do sistema de equações $y_i \cdot s \equiv 0 \pmod 2$, $1 \leq i \leq k$, é realizada classicamente, pois existem algoritmos clássicos que resolvem esses problemas em tempo polinomial. Assim, a solução proposta por Simon, combinando procedimentos clássicos e quânticos, é mais eficiente que os algoritmos puramente clássicos (i.e., sem componentes quânticas) conhecidos.

4.3.4 Algoritmo de Shor

Inspirado no algoritmo de Simon e em resultados da teoria dos números, Shor (em [Sho94]) definiu um algoritmo que aproveita as propriedades quânticas para fatorar números inteiros de maneira eficiente, reduzindo o problema de encontrar um fator primo ao problema de encontrar o período de uma função, da seguinte maneira:

1. Se o número n a ser fatorado é par, então o algoritmo retorna o fator 2.

2. Se n é um número primo, o que pode ser verificado eficientemente através do algoritmo clássico denominado AKS (apresentado em [AKS04]), o algoritmo retorna n.

3. Através de algoritmos clássicos eficientes decide-se se n é uma potência de um número primo (i.e. se existem $p, k > 1$ tais que $n = p^k$ e p é primo), e nesse caso o algoritmo retorna p. O algoritmo apresentado em [Ber98] serve para verificar eficientemente se n é uma potência perfeita e calcular p e k, e através do algoritmo AKS se determina se p é primo.

4. Se n não corresponde a nenhum dos casos anteriores, i.e., se n é um número composto ímpar que não é potência de primo, o fator primo é calculado da seguinte maneira:

   (a) Escolhe-se aleatoriamente um número x tal que $1 < x < n$.

   (b) Se $mdc(x, n) > 1$ o algoritmo retorna $mdc(x, n)$.

   (c) Se $mdc(x, n) = 1$ se calcula, através de um CQ o período r da função $f(a) = x^a \bmod n$. Se r for ímpar ou $x^{r/2} \equiv -1 \pmod n$ (o que acontece com baixa probabilidade) o procedimento falha e deve ser executado de

novo, em caso contrário, o algoritmo retorna $mdc(x^{r/2}+1, n)$. O fato de a função $f(a) = x^a \bmod n$ ser periódica (sob as condições especificadas para x e n) e de que $mdc(x^{r/2}+1, n)$ corresponda a um fator primo de n (sob as condições especificadas para r) são teoremas da teoria dos números (consequências do chamado 'Pequeno Teorema de Fermat').

Note que, no algoritmo de Shor, o único calculo realizado quanticamente é o calculo do período da função $f(a) = x^a \bmod n$, quando n é um número composto ímpar que não é potência de primo e x é co-primo de n. O calculo desse período através de um algoritmo clássico tem a mesma complexidade do que achar um fator primo de n. O CQ para calcular tal período consiste, basicamente, em calcular a função $f(a) = x^a \bmod n$ aproveitando o paralelismo quântico para depois extrair o período através do CQ que computa a denominada *transformada de Fourier quântica*, mais algumas operações adicionais. Shor demonstra que o número total de portas quânticas necessárias para construir esse circuito é polinomial com respeito ao tamanho do dado de entrada. O CQ que computa a função $f(a) = x^a \bmod n$ no algoritmo de Shor é especificado em detalhe, diferentemente dos algoritmos anteriores onde são supostos oráculos que computam as respetivas funções f.

4.3.5 Algoritmo de Grover

Em [Gro96], Grover define um algoritmo quântico que serve para buscar um elemento (que satisfaz uma determinada propriedade) dentro de uma lista desordenada de dados. Pode se pensar, por exemplo, no problema de buscar o nome de uma pessoa numa lista telefônica através do seu número de telefone. Numa lista de N elementos um algoritmo clássico precisa verificar $\frac{N}{2}+1$ elementos, no pior dos casos, para obter o elemento procurado com probabilidade maior do que $\frac{1}{2}$; i.e. o algoritmo toma $O(N)$ passos para realizar a busca. O algoritmo proposto por Grover consegue encontrar o elemento procurado, com probabilidade maior do que $\frac{1}{2}$, em $O(\sqrt{N})$ passos; o que representa uma melhora quadrática com respeito ao tempo de computação.

O problema de se encontrar um elemento dentro de uma lista de dados pode ser especificado abstratamente como o problema de inverter uma função, i.e. dada uma

função f e um valor fixo y tal que $f(x) = y$ (para um único valor x), o problema consiste em determinar x.

O algoritmo de Grover pressupõe a existência de um oráculo U_f que computa a função f (de maneira similar aos algoritmos anteriores, exceto no algoritmo de Shor), e é também pressuposto que $N = 2^n$ para algum $n \in \mathbb{N}$. As entradas do circuito são $n+1$ qubits, os primeiros n qubits no estado $|\,0\,\rangle$ e o último qubit no estado $|\,1\,\rangle$. No primeiro passo do algoritmo cria-se uma superposição uniforme de estados através de portas de Hadamard, e em seguida é realizada a chamada *iteração de Grover* um número $r = \left\lceil \pi\sqrt{2^n}/4 \right\rceil$ de vezes (o que corresponde a $O(\sqrt{N})$). A iteração de Grover utiliza o oráculo U_f, e a partir daí são usadas portas de Hadamard e de mudança de fase para fazer girar o vetor de entrada de maneira a fazê-lo se aproximar cada vez mais do vetor que representa o valor x.

Michel Boyer, Gilles Brassard, Peter Høyer e Alain Tapp, em [BBHT98], generalizam o algoritmo de Grover para casos em que o número m de elementos que satisfazem a propriedade de busca é maior do que 1, mas com a condição de que m deve ser conhecido com antecedência. Além disso, eles esboçam a maneira como pode ser realizado o algoritmo quântico para contar, numa lista desordenada, o número de elementos que satisfazem uma certa propriedade. Esse algoritmo foi depois definido em detalhe por Brassard, Høyer e Tapp em [BHT98]. Esses resultados implicam a possibilidade de resolver problemas NP-completos em tempo $O(\sqrt{2^n})$, que continua ainda sendo de ordem exponencial com respeito ao comprimento n da entrada, mas que representa uma melhora quadrática com respeito aos algoritmos clássicos. Finalmente, Charles Bennett, Ethan Bernstein, Gilles Brassard e Umesh Vazirani, em [BBBV97], demonstram que o algoritmo de Grover é ótimo, no sentido em que não pode ser definido um algoritmo quântico de busca, baseado num oráculo, que consuma menos recursos. Isso implica que problemas NP-completos não podem ser resolvidos eficientemente através de CQ baseados em métodos de busca 'ingênuos',[16] i.e., em listas desprovidas de estrutura.

[16] Para uma apresentação detalhada destes resultados veja [NC00].

4.4 Sobre o poder computacional dos modelos quânticos

Para estudar o poder computacional dos modelos quânticos foram definidas classes de complexidade algorítmica e foram estabelecidas algumas relações com classes de complexidade existentes. Entre essas classes talvez a mais representativa é **BQP**, contendo as linguagens que podem ser decididas por famílias uniformes de CQs de tamanho polinomial (ou, equivalentemente, por MTQs em tempo polinomial), com probabilidade de erro limitado. **BQP** é definida de maneira análoga a **BPP** (Seção 2.1) e caracteriza as linguagens que podem ser decididas eficientemente através de CQs e MTQs.

Devido a que as máquinas de Turing probabilísticas podem ser vistas como casos particulares de MTQs, tem-se que $\mathbf{BPP} \subseteq \mathbf{BQP}$. O algoritmo de Shor para fatorar números inteiros com um número polinomial de portas quânticas é uma evidência a favor de que a inclusão $\mathbf{BPP} \subseteq \mathbf{BQP}$ é própria, pois esse é um problema que muitos pesquisadores acreditam não ter solução polinomial probabilística, com erro limitado, nos modelos clássicos. Determinar, de maneira definitiva, se a inclusão $\mathbf{BPP} \subseteq \mathbf{BQP}$ é própria ou não é um dos grandes desafios atuais da pesquisa em computação quântica, pois tal resultado mostraria se realmente esses modelos quânticos permitem realizar computações de maneira mais eficiente.

São muitos os trabalhos e as conjeturas a respeito das potencialidades quanto à eficiência dos modelos quânticos: alguns pesquisadores acreditam que as potencialidades são devidas aos estados emaranhados (cf. por exemplo [Joz98]), enquanto outros acreditam que são devidas à existência de amplitudes de probabilidade negativas, que permitem diminuir a probabilidade de obter resultados não desejados e aumentar a probabilidade dos desejados (cf. [Aha98] e [For03]). Por outro lado, simulações clássicas eficientes de algumas subclasses importantes de CQs já foram demostradas (cf. [AB06] e [TD02]). Propostas que ajudem a esclarecer as verdadeiras potencialidades da computação quântica são de grande relevância e é isso que pretendemos fazer no seguinte capítulo, propondo uma maneira de entender algumas características da computação quântica através de noções lógicas.

Capítulo 5

Computação paraconsistente versus computação quântica

Embora os modelos de computação paraconsistente introduzidos nos capítulos 2 e 3 sejam generalizações dos modelos de MTs e circuitos booleanos através da lógica, e as MTQs e os CQs sejam generalizações dos mesmos modelos de computação realizadas a partir de uma perspectiva totalmente diferente (através dos princípios da mecânica quântica), os modelos de computação paraconsistente e quântica possuem várias características semelhantes que serão descritas neste capítulo. Essas semelhanças abrem um novo caminho para interpretar a computação quântica, desvelando características importantes de uma lógica subjacente a tais modelos, o que pode ser útil no esclarecimento de suas potencialidades. Na outra direção, as semelhanças também abrem possibilidades de pensar em implementações físicas de certas propriedades lógicas presentes em modelos de computação abstratos, que à primeira vista parecem não razoáveis.

Este capítulo é organizado da seguinte maneira: inicialmente mostramos como podem ser simuladas algumas computações de MTQs através de MTPs (Seção 5.1), apontando as semelhanças e diferenças entre estes dois modelos. A seguir indicamos algumas relações entre os modelos de CQs e L-circuitos (Seção 5.2) para as lógicas mbC e $S5$. Finalmente, propomos uma maneira de representar computações quânticas através de noções paraconsistentes e modais (Seção 5.3).

Como afirmado por Christos Papadimitriou em [Pap94, p. 45], o problema **P** =? **NP** consiste em determinar se o retardo exponencial na simulação de uma MTND, através de uma MTD, é inerente a esses modelos de computação ou é produto de nosso limitado entendimento do indeterminismo. As relações entre computação paraconsistente e quântica aqui apresentadas mostram uma interessante relação entre paralelismo e indeterminismo: o paralelismo pode ser obtido através de um indeterminismo 'parcialmente controlado', como será explicado principalmente na Seção 5.2. Assim, nossa abordagem lógica permite um melhor entendimento do papel do indeterminismo nos processos de computação, e mostra um novo caminho para estudar questões de complexidade algorítmica.

5.1 Simulação de MTQs através de MTPs

No primeiro modelo de MTPs, descrito na Seção 2.3, cada função de escolha de valores dos diferentes elementos da máquina (i.e. estado e posição atual da máquina e símbolos em cada célula da fita) representa uma configuração de uma MT clássica; portanto, a configuração de uma MTP pode ser interpretada como uma 'superposição' de configurações de uma MT. Sob esta interpretação, e supondo que no final da computação deve ser realizada só uma escolha de valores dos elementos da máquinas para obter um único resultado (assumindo que toda combinação de valores tem a mesma probabilidade de ser escolhida), as MTPs são similares às MTQs: as MTPs podem ser consideradas como MTQs nas quais são permitidas somente superposições *uniformes*, i.e. superposições onde os estados com amplitudes de probabilidade diferentes de 0 tem a mesma magnitude.[1]

Contudo, não é possível representar estados emaranhados nas MTPs através da mera interpretação da multiplicidade de valores como superposição de estados: note que qualquer superposição uniforme emaranhada de estados exclui alguma combinação de valores dos elementos constituintes,[2] enquanto, nas MTPs, qualquer combina-

[1] A magnitude de uma amplitude de probabilidade é seu módulo elevado ao quadrado.

[2] Por exemplo, no estado emaranhado $|\,\psi\,\rangle = \frac{1}{\sqrt{2}}(|\,0,0\,\rangle + |\,1,1\,\rangle)$, somente as combinações do valor 0 no primeiro qubit com o valor 0 no segundo qubit, e do valor 1 no primeiro qubit com o valor 1 no segundo qubit, são permitidas; as outras combinações são excluídas.

ção de valores na configuração da máquina pode ser obtida (i.e. nenhuma combinação é excluída).

Outra característica importante na representação de estados quânticos é a noção de *fase relativa*. Considere, por exemplo, os seguintes estados de um qubit: $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. As magnitudes das amplitudes para o estado $|1\rangle$ em ambos os estados são as mesmas (a saber $\frac{1}{2}$), mas as amplitudes têm sinal diferente. Diz-se que duas amplitudes de probabilidade a e b *diferem por uma fase relativa* θ se $a = e^{i\theta}b$, onde θ é um número real e $e^{i\theta}$ é chamado *fator de fase relativa*. Os estados $|\psi\rangle$ e $|\varphi\rangle$ acima diferem por uma fase relativa de π nas amplitudes do estado $|1\rangle$, e por um fator de fase relativa $e^{i\pi} = -1$. A noção de fase relativa nas MTQs também não é diretamente representável através das MTPs, e essa característica é essencial para a interferência quântica, que é o mecanismo provido pelas MTQs para aproveitar o paralelismo no processo de computação. O mecanismo provido pelas MTPs para aproveitar o paralelismo é bastante diferente da interferência quântica: consiste na adição de condições de inconsistência na definição das instruções (como explicado na Seção 2.3).

Apesar das diferenças entre MTQs e o primeiro modelo de MTPs, o modelo de MTPs permite vislumbrar os estados superpostos como estados contraditórios, propondo uma interpretação lógica para esse fenômeno quântico e, na outra direção, indicando como modelos de computação com estados contraditórios podem ser fisicamente implementados. Por outro lado, com os estados contraditórios e as condições de inconsistência, as MTPs permitem resolver os problemas de Deutsch e Deutsch-Jozsa com a mesma eficiência que no caso quântico, mostrando como esses problemas podem ser de certa forma 'de-quantizados'[3] (com as restrições impostas aos oráculos nesse modelo):

- O Exemplo 2.1 (p. 47) mostra como o problema de Deutsch pode ser resolvido através de MTPs. Note que a MTP do exemplo determina se uma função $f\colon \{0,1\} \to \{0,1\}$ é constante ou balanceada computando $f(0)$ e $f(1)$ em paralelo.

[3] De forma totalmente distinta da que propõe Christian Calude em [Cal07].

- A solução para o problema de Deutsch-Jozsa é uma generalização simples da MTP do Exemplo 2.1: o dado de entrada deve agora ser uma seqüencia de n símbolos 0; as instruções i_1 e i_2 devem ser substituídas pelas instruções $i_1 = q_1 00 q_2$, $i_2 = q_1 01 q_2$, $i_3 = q_1 0 R q_1$, $i_4 = q_1 \emptyset L q_3$, $i_5 = q_3 0 L q_3$ e $i_6 = q_3 \emptyset R q_4$ (que trocam os zeros da seqüência de entrada escrevendo os símbolos 0 e 1 em cada uma das n posições, para representar desta maneira todos os possíveis valores de entrada), e o oráculo $\mathcal{M}'$ deve ser uma MT paralelizável que compute uma função f: $\{0,1\}^n \rightarrow \{0,1\}$.[4]

Pelo Teorema 2.4, a solução dos problemas de Deutsch e Deutsch-Jozsa através de MTPs implica a existência de MTDs para resolver esses problemas, o que mostra como alguns algoritmos quânticos podem ser 'de-quantizados', sob as restrições impostas aos oráculos nas MTPs.

As MTPs também mostram a importância dos estados emaranhados no processo de computação: os estados emaranhados correspondem a inter-relações entre diferentes propriedades ou elementos da máquina (no sentido definido na Seção 2.4.1), sem tais inter-relações o paralelismo das MTPs pode ser simulado de maneira eficiente por MTDs (como mostrado no Teorema 2.4). Isso explica, em certa medida, a relevância dos estados emaranhados na computação quântica.

O segundo modelo de MTPs, que chamamos de MTPNSs, é um pouco mais próximo ao modelo de MTQs. Esse novo modelo permite representar estados emaranhados de maneira direta, pois as configurações das múltiplas cópias criadas na execução de uma MTPNSs não se misturam (como explicado na Seção 2.4), ao contrário do que acontece no primeiro modelo de MTPs. Considere, por exemplo, o estado emaranhado $|\,\psi\,\rangle = \frac{1}{\sqrt{2}}(|\,0,0\,\rangle + |\,1,1\,\rangle)$ como representando os símbolos nas células das posições 0 e 1 da fita de uma MTQ $\mathcal{M}$. Se se pensar que o estado dos outros elementos de $\mathcal{M}$ corresponde a um estado clássico (i.e. a um estado base, sem superposição de diferentes configurações), o estado de $\mathcal{M}$ é equivalente ao estado

[4]Neste caso a função não precisa ser constante ou balanceada, e a MTP determina se a função é constante ou não, o que corresponde a um problema mais geral do que o problema de Deutsch-Jozsa. Porém, é importante ter em conta que o oráculo deve ser uma MT paralelizável, o que torna difícil resolver problemas NP-Completos desta maneira, como explicado na Seção 2.3.

de uma MTPNS $\mathcal{M}'$ cuja execução dividiu a máquina em duas cópias: uma copia contendo o símbolo 0 nas posições 0 e 1 da fita, a outra copia contendo o símbolo 1 nas posições 0 e 1 da fita e os outros elementos de $\mathcal{M}'$ sendo iguais aos de $\mathcal{M}$. Assim, estados emaranhados de MTQs podem ser representados pelas configurações das múltiplas cópias criadas na execução das MTPNSs. Contudo, igual ao que acontece com o primeiro modelo de MTPs, as MTPNSs também não permitem representar fatores de fase relativa de uma maneira direta, o que dificulta a simulação da interferência quântica. Por outro lado, as condições de inconsistência nas instruções das MTPNSs parecem ser um mecanismo mais eficiente do que a interferência quântica para o aproveitamento do paralelismo, pois através do paralelismo e a interferência quântica não é possível definir um algoritmo, baseado num método de busca 'ingênuo', que resolva um problema NP-completo em tempo polinomial (cf. [NC00, Cap. 6]), enquanto nas MTPNSs é possível (é uma conseqüência imediata do Teorema 2.5).

Através do modelo de MTPNSs propomos também uma interpretação lógica de um fenômeno quântico: estados emaranhados podem ser considerados como conjunções não separáveis, pois é através da conjunção não separável, definida na Seção 2.4.1, que foram definidas as inter-relações entre os diferentes elementos das MTPNSs, o que possibilita pensar em diferentes cópias ou configurações independentes da máquina. Na outra direção, modelos de computação com elementos com valores não separáveis poderiam ser implementadas através de estados quânticos emaranhados.

Outro modelo de MTPs pode ser definido através da extensão da lógica considerada na definição das MTPNSs com um novo operador de possibilidade. Um operador de possibilidade seria usado para representar configurações 'positivas' e o outro configurações 'negativas'. Estabelecendo, por meio de axiomas, o comportamento dos dois operadores de possibilidade (de maneira que o comportamento desses operadores seja análogo ao comportamento do $+$ e o $-$), é possível representar a noção de fator de fase relativa para superposições uniformes. Seguindo este caminho, talvez seja possível definir um modelo de MTPs muito mais próximo ao modelo de MTQs, desvelando cada vez mais uma lógica adequada para interpretar os modelos

de computação quântica.

5.2 Relações entre CQs e L-circuitos

As variáveis ocultas usadas na definição dos L-circuitos permitem também estabelecer algumas relações entre esse modelo de computação e os CQs. Tais relações mostram, em certa medida, que a computação quântica poderia ser frutiferamente explicada através de uma interpretação de variáveis ocultas, em oposição à idéia de Deutsch de que algumas características da computação quântica, como o paralelismo quântico, não podem ser adequadamente explicadas através de uma interpretação diferente da de mundos possíveis (cf. [Deu85]).

Com o intuito de relacionar L-circuitos e CQs propomos simular estados superpostos através de variáveis ocultas. Nos mbC-circuitos, por exemplo, o circuito correspondente à fórmula $\neg p$ produz a saída $(\neg p)^* = x_p x'_p + 1$. Logo, se o valor de entrada é $x_p = 1$ então a saída é aleatória, dependendo do valor que tomar a variável oculta x'_p. O valor obtido na saída do circuito se comporta de maneira análoga ao valor obtido na medição de um qubit no estado $|\,\psi\,\rangle = |\,0\,\rangle + |\,1\,\rangle$. Assim, a variável oculta x'_p pode ser interpretada como o estado superposto $|\,\psi\,\rangle$, e a porta de negação serviria para produzir tal estado superposto, de maneira análoga ao valor computado pela porta de Hadamard com a entrada $|\,0\,\rangle$ nos CQs.

O paralelismo nos mbC-circuitos é atingido, então, substituindo as variáveis de entrada por variáveis ocultas (através de portas de negação, como explicado na Seção 3.3.1) e operando sobre essas variáveis ocultas. Por exemplo, o mbC-circuito correspondente à formula $(\sim \neg p) \wedge (\sim \neg q)$ produz a saída $x'_p x'_q$, o que corresponde à computação em paralelo da conjunção de duas variáveis proposicionais, sob a interpretação que estamos propondo. O problema consiste em definir estratégias para aproveitar os estados superpostos que resultam das computações em paralelo, eliminando variáveis ocultas, de maneira a obter (ou, pelo menos, aumentar a probabilidade de obter) um valor desejado na saída. Por exemplo, no caso de computar em paralelo os diferentes valores de verdade que pode tomar uma fórmula proposicional, o valor desejado poderia ser 1 se a fórmula fosse satisfatível ou 0 em caso contrário. Definir

uma tal estratégia para o caso dos mbC-circuitos parece impossível, devido ao fato que as variáveis ocultas tomam valores de maneira totalmente independente, mas é por outro lado perfeitamente possível em L-circuitos construídos sob outras lógicas (como no caso de $S5$), onde os valores de algumas variáveis estão inter-relacionados.

Nos $S5$-circuitos, o operador de inconsistência $\bullet$ pode ser usado para simular a porta de Hadamard de maneira mais aproximada que a negação nos mbC-circuitos. Neste caso, como $(\bullet p)^* = (x_{\Box p} + 1)(x_{\Box \neg p} + 1)$, se o valor de entrada é $x_p = 0$ então a saída é $x_{\Box \neg p}+1$ (pela restrição polinomial (cT)), e se o valor de entrada é $x_p = 1$ então a saída é $x_{\Box p} + 1$ (também pela restrição polinomial (cT)). Portanto, dependendo do valor de entrada se produz estados de indeterminação diferentes, que podem ser interpretados como superposições diferentes. Esse comportamento é similar ao que acontece com a porta de Hadamard, que produz a superposição $|\,\psi\,\rangle = \frac{1}{\sqrt{2}}(|\,0\,\rangle + |\,1\,\rangle)$ para o valor de entrada $|\,0\,\rangle$, e a superposição $|\,\psi\,\rangle = \frac{1}{\sqrt{2}}(|\,0\,\rangle - |\,1\,\rangle)$ para o valor de entrada $|\,1\,\rangle$. A dupla aplicação do operador de inconsistência produz como saída o valor constante 0, o que se corresponde com o que faz a porta de Hadamard para a entrada $|\,0\,\rangle$, mas não para a entrada $|\,1\,\rangle$.

O parágrafo anterior indica como podem ser simuladas diferentes superposições de estados através de diferentes variáveis ocultas, e como as variáveis ocultas podem desaparecer mediante a aplicação de outros conectivos. Por outro lado, no CAP para $S5$, existem variáveis ocultas indexadas por fórmulas complexas que podem ser decompostas em termos de variáveis ocultas indexadas por fórmulas de complexidade menor, como é o caso de $x_{\Box(p \wedge q)}$ que é igual a $x_{\Box p} x_{\Box q}$; e existem outras variáveis ocultas que não podem ser decompostas dessa maneira, como por exemplo $x_{\Box(p \vee q)}$. As variáveis ocultas que não podem ser decompostas em termos de variáveis indexadas por fórmulas de complexidade menor podem servir para representar estados emaranhados.

Embora os mbC-circuitos e os $S5$-circuitos, através da interpretação que estamos propondo, não consigam representar fielmente as características dos CQs, mostram de maneira clara que há uma possibilidade de interpretar a computação quântica por meio de variáveis ocultas, sem necessidade de aludir a mundos possíveis. Também mostram como o paralelismo nos processos de computação pode ser obtido medi-

ante a introdução de indeterminismo, parcialmente controlado de acordo com certas restrições.

5.3 Expressando computações quânticas através de noções modais paraconsistentes

Inicialmente, seguindo a idéia sugerida no final da Seção 5.1, vamos propor uma maneira de representar registros de qubits (considerando somente superposições uniformes) através de uma lógica com dois operadores de possibilidade, mais um operador de 'produto tensorial'. Indicamos a partir daí como podem ser definidos operadores sobre essas representações, permitindo assim expressar computações quânticas.

Na representação de qubits usamos literais para expressar os valores que podem ser obtidos no processo de medição: as variáveis proposicionais representarão o valor 1 e a negação das variáveis proposicionais representarão o valor 0. Dessa forma, retomando a definição do operador $\bullet$ na lógica modal $S5$ (apresentada na Seção 2.4), a fórmula $\bullet p$ (que é equivalente a $\Diamond p \wedge \Diamond \neg p$ em $S5$) expressa a possibilidade de obter os valores 0 e 1 na medição do qubit representado por p. Essa fórmula será então usada para representar o estado superposto $|\,\psi\,\rangle = |\,0\,\rangle + |\,1\,\rangle$.

Para representar a noção de fase relativa estendemos $S5$ acrescentando um novo operador de possibilidade, o qual vai interagir com o operador de possibilidade alética de $S5$ imitando as regras dos sinais algébricos de $+$ e $-$. A lógica resultante será chamada *lógica das modalidades pró-possíveis e anti-possíveis (PN)*. É claro que PN é uma lógica legitimamente bi-modal, no sentido em que as modalidades atuam de maneira independente, e contudo podem se combinar (cf. [CP08]). Nessa lógica o novo operador de possibilidade, chamado de *anti-possibilidade*, será denotado por $\Diamond\!\!\!\!-$ (definindo o operador dual de anti-necessidade $\boxdot$ por $\boxdot A \stackrel{\text{def}}{=} \neg \Diamond\!\!\!\!- \neg A$), e os operadores usuais (aléticos) de possibilidade e necessidade de $S5$ serão denotados por $\Diamond\!\!\!\!+$ e $\boxplus$, e denominados de *pró-possibilidade* e *pró-necessidade*, respectivamente. Vamos supor que $\Diamond\!\!\!\!-$ obedece às mesmas propriedades que o operador de possibilidade na lógica modal K. Além disso, os seguintes axiomas regularão a interação entre as diferentes

modalidades (usamos $\alpha \equiv \beta$ para denotar $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$):

$$\Diamond\Diamond A \equiv \Diamond\!\!\!\!-\, A, \tag{AX1}$$

$$\Diamond\!\!\!\!-\, A \rightarrow \Diamond A, \tag{AX2}$$

$$\Diamond\!\!\!\!-\, \Diamond A \rightarrow \Diamond A, \tag{AX3}$$

$$\Diamond\!\!\!\!-\, \boxdot A \rightarrow \boxdot A, \tag{AX4}$$

$$\Diamond \boxplus A \equiv \boxdot A, \tag{AX5}$$

$$\Diamond \boxdot A \equiv \boxplus A. \tag{AX6}$$

Em PN são validas as seguintes reduções de modalidades:

Teorema 5.1.

$$\Diamond\!\!\!\!-\, \Diamond\!\!\!\!-\, A \equiv \Diamond\!\!\!\!-\, A, \qquad \Diamond\Diamond A \equiv \Diamond\!\!\!\!-\, A, \qquad \boxplus \Diamond\!\!\!\!-\, A \equiv \Diamond\!\!\!\!-\, A, \qquad \boxdot \Diamond A \equiv \Diamond\!\!\!\!-\, A,$$

$$\boxplus\boxplus A \equiv \boxplus A, \qquad \boxdot\boxdot A \equiv \boxplus A, \qquad \Diamond\!\!\!\!-\, \boxplus A \equiv \boxplus A, \qquad \Diamond \boxdot A \equiv \boxplus A,$$

$$\Diamond\!\!\!\!-\, \Diamond A \equiv \Diamond A, \qquad \Diamond\Diamond\!\!\!\!-\, A \equiv \Diamond A, \qquad \boxplus \Diamond A \equiv \Diamond A, \qquad \boxdot \Diamond\!\!\!\!-\, A \equiv \Diamond A,$$

$$\boxplus\boxdot A \equiv \boxdot A, \qquad \boxdot\boxplus A \equiv \boxdot A, \qquad \Diamond\!\!\!\!-\, \boxdot A \equiv \boxdot A, \qquad \Diamond \boxplus A \equiv \boxdot A.$$

Seguindo as idéias apresentadas na Seção 2.4, podem ser definidas duas negações paraconsistentes ($\neg_{\Diamond\!\!\!\!-}$ e $\neg_{\Diamond}$) e duas conjunções não separáveis ($\wedge_{\Diamond\!\!\!\!-}$ e $\wedge_{\Diamond}$) em PN: $\neg_{\Diamond\!\!\!\!-} A \stackrel{\text{def}}{=} \Diamond\!\!\!\!-\, \neg A$, $\neg_{\Diamond} A \stackrel{\text{def}}{=} \Diamond \neg A$, $A \wedge_{\Diamond\!\!\!\!-} B \stackrel{\text{def}}{=} \Diamond\!\!\!\!-\, (A \wedge B)$ e $A \wedge_{\Diamond} B \stackrel{\text{def}}{=} \Diamond (A \wedge B)$. Desse modo, adicionando a PN o esquema axiomático:

$$\Diamond_i (A \wedge \Diamond_j B) \equiv \Diamond_i A \wedge \Diamond_i \Diamond_j B, \tag{AX7}$$

onde $\Diamond_i, \Diamond_j \in \{\Diamond\!\!\!\!-, \Diamond\}$, podem ser definidos quatro conectivos de inconsistência: $\bullet_+ A \stackrel{\text{def}}{=} A \wedge_{\Diamond\!\!\!\!-} \neg_{\Diamond\!\!\!\!-} A$ (que é equivalente a $\Diamond\!\!\!\!-\, \neg A \wedge \Diamond\!\!\!\!-\, A$ em PN), $\bullet_- A \stackrel{\text{def}}{=} A \wedge_{\Diamond} \neg_{\Diamond\!\!\!\!-} A$ (que é equivalente a $\Diamond \neg A \wedge \Diamond A$ em PN), $\bullet_{\mp} A \stackrel{\text{def}}{=} A \wedge_{\Diamond\!\!\!\!-} \neg_{\Diamond} A$ (que é equivalente a $\Diamond\!\!\!\!-\, \neg A \wedge \Diamond\!\!\!\!-\, A$ em PN) e $\bullet_{\pm} A \stackrel{\text{def}}{=} A \wedge_{\Diamond} \neg_{\Diamond} A$ (que é equivalente a $\Diamond\!\!\!\!-\, \neg A \wedge \Diamond A$ em PN). Esses conectivos serão usados para representar diferentes estados superpostos uniformes da

seguinte maneira:

$$\bullet_{+}\ A \text{ representa o estado } |\,\psi\,\rangle = |\,0\,\rangle + |\,1\,\rangle,$$
$$\bullet_{-}\ A \text{ representa o estado } |\,\psi\,\rangle = -|\,0\,\rangle - |\,1\,\rangle,$$
$$\bullet_{\mp}\ A \text{ representa o estado } |\,\psi\,\rangle = -|\,0\,\rangle + |\,1\,\rangle,$$
$$\bullet_{\pm}\ A \text{ representa o estado } |\,\psi\,\rangle = |\,0\,\rangle - |\,1\,\rangle.$$

Nesta representação as magnitudes das amplitudes de probabilidade não são consideradas, já que estamos supondo que todas são iguais e portanto não necessitam ser explicitamente apontadas.

Usando a negação $\neg_{\Diamond\!\!\!\!-}$ quatro conectivos de consistência podem também ser definidos (usando a negação $\neg_{\Diamond}$ em lugar de $\neg_{\Diamond\!\!\!\!-}$ se obtêm definições equivalentes): $\circ_{+}A \stackrel{\text{def}}{=} \neg_{\Diamond\!\!\!\!-} \bullet_{+} A$ (que é equivalente a $\boxplus\neg A \vee \boxplus A$ em PN), $\circ_{-}A \stackrel{\text{def}}{=} \neg_{\Diamond\!\!\!\!-} \bullet_{-} A$ (que é equivalente a $\boxdot\neg A \vee \boxdot A$ em PN), $\circ_{\pm}A \stackrel{\text{def}}{=} \neg_{\Diamond\!\!\!\!-} \bullet_{\mp} A$ (que é equivalente a $\boxplus\neg A \vee \boxdot A$ em PN) e $\circ_{\mp}A \stackrel{\text{def}}{=} \neg_{\Diamond\!\!\!\!-} \bullet_{\pm} A$ (que é equivalente a $\boxdot\neg A \vee \boxplus A$ em PN). Com estas definições as seguintes derivações podem ser demonstradas:

Teorema 5.2.

$$A, \neg_{\Diamond\!\!\!\!-}A, \circ_{+}A \vdash_{PN} B; \qquad \Diamond A, \neg_{\Diamond\!\!\!\!-}A, \circ_{\pm}\neg A \vdash_{PN} B; \qquad \Diamond A, \neg_{\Diamond\!\!\!\!-}A, \circ_{\mp}A \vdash_{PN} B;$$
$$\Diamond A, \neg_{\Diamond}A, \circ_{-}A \vdash_{PN} B; \qquad A, \neg_{\Diamond}A, \circ_{\pm}\neg A \vdash_{PN} B; \qquad A, \neg_{\Diamond}A, \circ_{\mp}\neg A \vdash_{PN} B.$$

Na representação de n-qubits, o estado base $|\,a_1, a_2, \ldots, a_n\,\rangle$ (onde $a_i \in \{0,1\}$) será representado por $\Diamond_i(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n)$, onde $\Diamond_i$ é $\Diamond\!\!\!\!-$ se o fator de fase relativo é positivo ou é $\Diamond$ em caso contrário, e onde $\alpha_j = p_j$ $(1 \leq j \leq n)$ se $a_j = 1$ ou $\alpha_j = \neg p_j$ em caso contrário.

Para expressar o produto tensorial vamos acrescentar a PN o conectivo $\otimes$, regido

pelos seguintes axiomas:

$$(\Diamond\!\!\!\!-A \otimes \Diamond\!\!\!\!-B) \rightarrow \Diamond\!\!\!\!-(A \wedge B), \tag{AX8}$$

$$(\Diamond\!\!\!\!-A \otimes \Diamond B) \rightarrow \Diamond(A \wedge B), \tag{AX9}$$

$$(\Diamond A \otimes \Diamond\!\!\!\!-B) \rightarrow \Diamond(A \wedge B), \tag{AX10}$$

$$(\Diamond A \otimes \Diamond B) \rightarrow \Diamond\!\!\!\!-(A \wedge B), \tag{AX11}$$

$$((A \wedge B) \otimes C) \rightarrow ((A \otimes C) \wedge (B \otimes C)), \tag{AX12}$$

$$(A \otimes (B \wedge C)) \rightarrow ((A \otimes B) \wedge (A \otimes C)). \tag{AX13}$$

Para ilustrar como estes axiomas podem representar produtos tensoriais, considere as fórmulas $\bullet_+ A$ e $\bullet_\pm B$, as quais representam os estados $|\,\psi\,\rangle = |\,0\,\rangle + |\,1\,\rangle$ e $|\,\varphi\,\rangle = |\,0\,\rangle - |\,1\,\rangle$, respectivamente. Pelas definições de $\bullet_+$ e $\bullet_\pm$, e os axiomas (AX8) a (AX13), tem-se que $\bullet_+ A \otimes \bullet_\pm B \vdash_{PN} \Diamond\!\!\!\!-(\neg A \wedge \neg B) \wedge \Diamond(\neg A \wedge B) \wedge \Diamond\!\!\!\!-(A \wedge \neg B) \wedge \Diamond(A \wedge B)$, que sob a interpretação que estamos propondo expressa o estado $|\,\psi, \varphi\,\rangle = |\,\psi\,\rangle \otimes |\,\varphi\,\rangle = |\,00\,\rangle - |\,01\,\rangle + |\,10\,\rangle - |\,11\,\rangle$.

Para expressar o fenômeno da interferência quântica, consideraremos que as possibilidades $\Diamond\!\!\!\!-$ e $\Diamond$ se anulam mutuamente. Portanto, é lícito estabelecer que a fórmula α *interfere destrutivamente* na teoria Γ se $\Gamma \vdash_{PN} \Diamond\!\!\!\!-\alpha$ e $\Gamma \vdash_{PN} \Diamond\alpha$. Conseqüentemente, os estados de n-qubits são representados por teorias em PN assim:

Definição 5.1. Seja Γ um conjunto de fórmulas de PN, com conjunto de variáveis proposicionais $Var(\Gamma) = \bigcup_{\gamma \in \Gamma} Var(\gamma) = \{p_1, p_2, \ldots, p_n\}$. A teoria Γ *representa o estado do n-qubit* $|\,\psi\,\rangle$ em PN se:

1. as fórmulas que representam os estados-base presentes em $|\,\psi\,\rangle$ (i.e. os estados-base com amplitudes diferentes de 0) são conseqüências de Γ em PN, e

2. as fórmulas que representam estados-base não presentes em $|\,\psi\,\rangle$ não são conseqüências de Γ em PN, ou interferem destrutivamente em Γ.

Os exemplos a seguir ilustram a representação de estados de n-qubits na lógica PN:

Exemplo 5.1. Seja $\Gamma = \{\Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2), \Diamond\!\!\!\!-(p_1 \wedge p_2)\}$. Pela definição 5.1, Γ representa o estado emaranhado $|\,\psi\,\rangle = |\,0,0\,\rangle + |\,1,1\,\rangle$ em PN, pois $\Gamma \vdash_{PN} \Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2)$

(que representa o estado base $|\,0,0\,\rangle$), $\Gamma \vdash_{PN} \Diamond\!\!\!\!-(p_1 \wedge p_2)$ (que representa o estado base $|\,1,1\,\rangle$), $\Gamma \nvdash_{PN} \Diamond(\neg p_1 \wedge \neg p_2)$ (portanto, a fórmula $\neg p_1 \wedge \neg p_2$ não interfere destrutivamente em Γ), $\Gamma \nvdash_{PN} \Diamond(p_1 \wedge p_2)$ (portanto, a fórmula $p_1 \wedge p_2$ não interfere destrutivamente em Γ), $\Gamma \nvdash_{PN} \Diamond\!\!\!\!-(\neg p_1 \wedge p_2)$ (que representa o estado base $|\,0,1\,\rangle$) e $\Gamma \nvdash_{PN} \Diamond\!\!\!\!-(p_1 \wedge \neg p_2)$ (que representa o estado base $|\,1,0\,\rangle$).

Exemplo 5.2. Seja $\Gamma = \{\bullet_{+}p_1, \bullet_{\pm}p_1\}$. Pela definição 5.1, Γ representa o estado $|\,\psi\,\rangle = |\,0\,\rangle$ em PN, pois $\Gamma \vdash_{PN} \Diamond\!\!\!\!-\neg p_1$ (que representa o estado base $|\,0\,\rangle$), $\Gamma \nvdash_{PN} \Diamond\neg p_1$ (portanto, a fórmula $\neg p_1$ não interfere destrutivamente em Γ), e $\Gamma \vdash_{PN} \Diamond\!\!\!\!-p_1$ (que representa o estado-base $|\,1\,\rangle$), mas $\Gamma \vdash_{PN} \Diamond p_1$ (portanto, a fórmula p_1 interfere destrutivamente em Γ).

As computações nos CQs consistem basicamente em transformações, nível por nível, do estado de registros de qubits através de operadores unitários. Para expressar computações de CQs por meio de noções lógicas, os operadores unitários de aridade n serão representados por funções F: $ForNQ \to ForNQ$, onde $ForNQ$ é o conjunto de fórmulas que representam algum estado de um n-qubit em PN. Para F ser bem definida, se α e β são fórmulas de PN que representam o mesmo estado de um n-qubit, então F deve satisfazer a condição $F(\alpha) = F(\beta)$. Por outro lado, pela linearidade dos operadores quânticos, as funções F devem satisfazer as condições:

$$F(\Diamond_i(\alpha_1 \wedge \ldots \wedge \alpha_n) \wedge \ldots \wedge \Diamond_j(\beta_1 \wedge \ldots \wedge \beta_n)) \\ = F(\Diamond_i(\alpha_1 \wedge \ldots \wedge \alpha_n)) \wedge \ldots \wedge F(\Diamond_j(\beta_1 \wedge \ldots \wedge \beta_n)), \quad (5.1)$$

onde $\Diamond_i, \Diamond_j \in \{\Diamond\!\!\!\!-, \Diamond\}$ e α_l, β_l são literais com variável proposicional p_l, e

$$F(\Diamond\alpha) = \Diamond F(\Diamond\!\!\!\!-\alpha). \quad (5.2)$$

Portanto as funções para representar operadores unitários podem ser descritas indicando somente as transformações que realizam sobre as representações de estados base. Por exemplo, as transformações realizadas pela porta de Hadamard (Equação 4.8) são representadas pela função $\mathcal{H}$ tal que:

$$\mathcal{H}(\Diamond\!\!\!\!-\neg p) = \bullet_{+}p, \\ \mathcal{H}(\Diamond\!\!\!\!-p) = \bullet_{\pm}p. \quad (5.3)$$

E as transformações realizadas pela porta U_f, do CQ definido na Seção 4.3.1, são representadas pela função $\mathcal{U}$ tal que:

$$\mathcal{U}(\Diamond(\neg p_1 \wedge \neg p_2)) = \begin{cases} \Diamond(\neg p_1 \wedge \neg p_2) \text{ se } f(0) = 0, \\ \Diamond(\neg p_1 \wedge p_2) \text{ se } f(0) = 1. \end{cases}$$

$$\mathcal{U}(\Diamond(\neg p_1 \wedge p_2)) = \begin{cases} \Diamond(\neg p_1 \wedge p_2) \text{ se } f(0) = 0, \\ \Diamond(\neg p_1 \wedge \neg p_2) \text{ se } f(0) = 1. \end{cases}$$

$$\mathcal{U}(\Diamond(p_1 \wedge \neg p_2)) = \begin{cases} \Diamond(p_1 \wedge \neg p_2) \text{ se } f(1) = 0, \\ \Diamond(p_1 \wedge p_2) \text{ se } f(1) = 1. \end{cases}$$

$$\mathcal{U}(\Diamond(p_1 \wedge p_2)) = \begin{cases} \Diamond(p_1 \wedge p_2) \text{ se } f(1) = 0, \\ \Diamond(p_1 \wedge \neg p_2) \text{ se } f(1) = 1. \end{cases} \tag{5.4}$$

Para expressar a aplicação de uma porta unitária, que opera sobre n-qubits, sobre um registro de m-qubits (com $m \geq n$), os qubits a ser considerados serão especificados por meio de subíndices, quando for necessário. Por exemplo, a expressão $\mathcal{H}_2(\Diamond(p_1 \wedge p_2))$ indicará que a transformação $\mathcal{H}$ vai ser realizada sobre o qubit representado por p_2. Assim, o resultado de aplicar uma transformação $F_{i_1,\ldots,i_n}$, de aridade n, sobre a representação de um estado base de um m-qubit será:

$$F_{i_1,\ldots,i_n}(\Diamond(\alpha_1 \wedge \ldots \wedge \alpha_m)) = \Diamond(\bigwedge_{j \notin \{i_1,\ldots,i_n\}} \alpha_j) \otimes F(\Diamond(\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n})). \tag{5.5}$$

Usando as definições anteriores (e denotando através de concatenação as transformações de um mesmo nível do circuito) podem ser expressos CQs cujas transformações preservem a uniformidade dos estados. Por exemplo, o CQ da Seção 4.3.1 é expresso pela fórmula:

$$\mathcal{H}_1(\mathcal{U}(\mathcal{H}_1\mathcal{H}_2(\Diamond\neg p_1 \otimes \Diamond p_2))).$$

As transformações realizadas pelo circuito podem ser seguidas eliminando o conectivo $\otimes$ e os operadores presentes na fórmula que representa o circuito, levando em consideração as propriedades de $\otimes$ e dos operadores F, e a definição 5.1. No exemplo

anterior, supondo que $f(0) = f(1) = 0$ em 5.4, as transformações realizadas são as seguintes:

$$
\begin{aligned}
&\mathcal{H}_1(\mathcal{U}(\mathcal{H}_1\mathcal{H}_2(\Diamond\!\!\!\!-\neg p_1 \otimes \Diamond\!\!\!\!-\, p_2))) \\
&\Rightarrow \mathcal{H}_1(\mathcal{U}(\mathcal{H}_1\mathcal{H}_2(\Diamond\!\!\!\!-(\neg p_1 \wedge p_2)))) \text{ por (AX8)} \\
&\Rightarrow \mathcal{H}_1(\mathcal{U}(\mathcal{H}_1(\Diamond\!\!\!\!-\neg p_1) \otimes \mathcal{H}_2(\Diamond\!\!\!\!-\, p_2))) \text{ por (5.5)} \\
&\Rightarrow \mathcal{H}_1(\mathcal{U}((\Diamond\!\!\!\!-\, p_1 \wedge \Diamond\!\!\!\!-\neg p_1) \otimes (\Diamond p_2 \wedge \Diamond\!\!\!\!-\neg p_2))) \text{ por (5.3) e def. de } \bullet_+ \text{ e } \bullet_\pm \\
&\Rightarrow \mathcal{H}_1(\mathcal{U}(\Diamond(p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(p_1 \wedge \neg p_2) \wedge \Diamond(\neg p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2))) \\
&\quad \text{por (AX8) a (AX13)} \\
&\Rightarrow \mathcal{H}_1(\Diamond(\mathcal{U}(\Diamond\!\!\!\!-(p_1 \wedge p_2))) \wedge \mathcal{U}(\Diamond\!\!\!\!-(p_1 \wedge \neg p_2)) \wedge \Diamond(\mathcal{U}(\Diamond\!\!\!\!-(\neg p_1 \wedge p_2))) \wedge \\
&\quad \mathcal{U}(\Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2))) \text{ por (5.1) e (5.2)} \\
&\Rightarrow \mathcal{H}_1(\Diamond\Diamond\!\!\!\!-(p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(p_1 \wedge \neg p_2) \wedge \Diamond\Diamond\!\!\!\!-(\neg p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2)) \\
&\quad \text{por (5.4)} \\
&\Rightarrow \mathcal{H}_1(\Diamond(p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(p_1 \wedge \neg p_2) \wedge \Diamond(\neg p_1 \wedge p_2) \wedge \Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2)) \\
&\quad \text{por Teorema 5.1} \\
&\Rightarrow \Diamond(\mathcal{H}_1(\Diamond\!\!\!\!-(p_1 \wedge p_2))) \wedge \mathcal{H}_1(\Diamond\!\!\!\!-(p_1 \wedge \neg p_2)) \wedge \Diamond(\mathcal{H}_1(\Diamond\!\!\!\!-(\neg p_1 \wedge p_2))) \wedge \\
&\quad \mathcal{H}_1(\Diamond\!\!\!\!-(\neg p_1 \wedge \neg p_2)) \text{ por (5.1) e (5.2)} \\
&\ldots \\
&\Rightarrow \Diamond\!\!\!\!-\neg p_1 \wedge \bullet_\pm p_2.
\end{aligned}
$$

A representação de computações quânticas que propomos através do formalismo modal da lógica PN pode ser considerada uma legítima linguagem de programação, que substitui operações algébricas por operações lógicas, aproximando-se bastante de uma linguagem de programação funcional clássica. Essa linguagem caracteriza um fragmento importante da computação quântica que corresponde às computações quânticas onde todas as superposições são uniformes: de fato, com base na Definicão 5.1, é possível demonstrar sem dificuldade tal caracterização. Uma linguagem de programação como a que propomos pode ser proveitosa na definição e validação de algoritmos quânticos.

Capítulo 6

Paraconsistência, fenômenos quânticos e computação: algumas reflexões

No capítulo anterior foram apresentadas e avaliadas algumas relações entre os modelos de computação paraconsistente, introduzidos nos capítulos anteriores, e os modelos de computação quântica. Defendemos que certas noções lógicas, especialmente baseadas em lógicas paraconsistente, são naturalmente apropriadas para interpretar e compreender o alcance e as potencialidades da computação quântica. O ponto alto de nossa análise concerne dois aspectos essenciais: primeiro, estados superpostos, indispensáveis na teoria quântica, podem ser legitimamente considerados como *estados inconsistentes*, e segundo, que estados emaranhados podem, igualmente de forma natural, serem vistos como *conjunções não-separáveis*. Além disso, indicamos como o paralelismo intrínseco aos modelos de computação quântica pode ser simulado (ao menos parcialmente) através das *variáveis ocultas nos L-circuitos* e sugerimos como o conceito de fase relativa pode ser interpretado através de certos *operadores modais de pró-possibilidade e anti-possibilidade*.

Embora essas relações sejam parciais, sugerem um novo caminho que pode resultar frutífero na interpretação da computação quântica. Tal caminho ainda não

foi totalmente explorado, já que existe um universo enorme de lógicas paraconsistêntes (e ainda podem ser definidas infinitas novas lógicas), cada uma delas com características próprias. Assim, não é uma tarefa fácil identificar se de fato todas as propriedades da computação quântica podem ser representadas adequadamente através de alguma lógica paraconsistente. Acreditamos, contudo, firmemente que uma tal interpretação seria útil ao esclarecimento das verdadeiras potencialidades de tais modelos de computação.

Apresentamos agora alguns argumentos que justificam nossa confiança na interpretação paraconsistente da computação quântica:

1. **Outras propostas com elementos similares**: em [Bat05], Giulia Battilotti chega, através de um caminho totalmente diferente ao seguido por nós, à mesma conclusão de que a lógica 'interna' à computação quântica deve ser paraconsistente e que as contradições são úteis para representar estados superpostos. Por outro lado, Maria Luisa Dalla Chiara, Roberto Giuntini e Roberto Leporini em [DGL03] propõem 'lógicas computacionais quânticas' definindo novos operadores lógicos através das portas dos CQs; as lógicas por eles definidas incluem operadores de negação paraconsistentes; os autores, porém, não fazem menção explicita a tal característica.

2. **Algumas propostas indicando que a lógica paraconsistente é adequada como lógica subjacente à mecânica quântica**: num escopo mais amplo, alguns trabalhos apontam que a lógica paraconsistente é apropriada para ser usada como lógica subjacente à mecânica quântica em geral (não somente para os modelos de computação quântica). Um desses trabalhos (já mencionado no final da Seção 4.1) é o de Newton da Costa e Décio Krause (cf. [dK04] e [dK06]) que propõem uma lógica paraconsistente para caracterizar o conceito de complementariedade introduzido por Niels Bohr. Outra proposta na mesma direção, mas com um outro enfoque totalmente diferente, é a de Dalla Chiara e Giuntini (cf. [DG02]), que consiste em definir 'lógicas quânticas paraconsistentes' através de certos tipos de estruturas algébricas. Embora estas idéias sejam diferentes das idéias propostas no presente trabalho e pare-

çam não ter qualquer aplicação na interpretação da computação quântica, tais propostas também apontam características paraconsistentes na descrição que a mecânica quântica faz do mundo físico.

3. **Semelhanças entre semânticas adequadas para lógicas paraconsistentes e interpretações da mecânica quântica**: é realmente surpreendente que novas propostas de semânticas, que resultaram ser aptas para caracterizar lógicas paraconsistentes, estejam baseadas em concepções muito próximas dos princípios norteadores de algumas interpretações da mecânica quântica. Em particular, a *semântica de traduções possíveis* (já mencionada no final do Capítulo 2), traduzindo fórmulas em múltiplos 'cenários lógicos', guarda bastante semelhança com a interpretação de múltiplos mundos: cada cenário lógico pode ser pensado como correspondendo à descrição de um dos múltiplos mundos; de fato, esta idéia já é apontada em [MS04]. Na mesma direção, as lógicas paraconsistentes definidas através de lógicas modais, devido a sua interpretação por meio da semântica de mundos possíveis, também permitem capturar a idéia de múltiplos mundos da mecânica quântica. Por outro lado, o cálculo de anéis de polinômios (descrito na Seção 3.2), ao introduzir 'variáveis ocultas' para expressar o indeterminismo dos conectivos presentes nas lógicas paraconsistentes (e outras lógicas não clássicas), segue a mesma idéia das teorias quânticas de variáveis ocultas, indicando que o indeterminismo implícito na semântica dessas lógicas é produto do desconhecimento dos valores de tais variáveis.

Devido à importância de se oferecer uma interpretação sólida da mecânica quântica que possa ser comumente aceita pela comunidade científica, continuar explorando conexões entre tal teoria física e lógicas paraconsistentes representa um desafio de grande relevância. Não deve ser por acaso que tão variadas propostas acabem apontando relações entre conceitos quânticos e paraconsistentes.

Começar por estudar uma lógica subjacente ao escopo da computação quântica, ao invés de estudar uma lógica da mecânica quântica em geral, representa vantagens metodológicas porque muitas das características essenciais da mecânica quântica já estão presentes nos modelos de computação quântica, onde podem ser estudados

de maneira simplificada para depois serem estendidas a sistemas quânticos em geral. Além disso, o desejo de entender as verdadeiras potencialidades da computação quântica tem estimulado fortemente a análise rigorosa de algumas propriedades fundamentais da mecânica quântica, como é o caso dos estados emaranhados, e parece estar renovando o debate a respeito da interpretação da mecânica quântica. Deutsch, por exemplo, afirma em [Deu85] que os computadores quânticos podem ser úteis na validação da interpretação de múltiplos mundos. Questões referentes à não-localidade e interação à distância também estão sendo amplamente discutidas, em relação a tais discussões vamos fazer a seguir algumas observações. Ainda mais, não podemos perder de vista que o que se configura como computação quântica poderia, em princípio, ser estudado de maneira abstrata, independente dos princípios contemporaneamente aceitos pela mecânica quântica.

Desde o trabalho fundacional de Turing (cf. [Tur36]), no qual definiu o modelo de máquinas abstratas, foi explicitada a condição de que o número de células 'observadas' pela máquina a cada instante de tempo deveria ser limitado, e de que as mudanças de símbolos nas células da fita deveriam também estar confinadas ao fragmento observado, reduzindo finalmente tal fragmento a uma única célula. Portanto, o modelo de máquina de Turing opera de maneira estritamente 'local', sem permitir qualquer tipo de 'ação a distância'. De maneira análoga, quando Deutsch definiu o modelo de MTQs foi colocada a restrição de que a máquina deveria operar por 'meios finitos', indicando com isso que : "(i) somente um subsistema finito (embora não sempre o mesmo) pode mudar em cada passo de computação, (ii) as mudanças dependem unicamente do estado de um subsistema finito e (iii) as regras que especificam as mudanças do sistema podem ser dadas finitamente no sentido matemático (por exemplo como um inteiro)." (cf. [Deu85, pág. 100], tradução nossa). Dessa forma, tendo em consideração que as condições impostas por Deutsch são bastante similares às impostas por Turing, poderíamos então concluir que as MTQs operam de maneira local?

À primeira vista parece que a resposta à pergunta formulada no parágrafo anterior deve ser claramente afirmativa, porém, algumas propriedades quânticas como os estados emaranhados de fato permitem certo tipo de ação a distância, de acordo

com algumas interpretações da mecânica quântica (cf. [Ber08]).

A questão da não-localidade na mecânica quântica tem sido extensamente discutida. No famoso argumento de Einstein, Podolsky e Rosen (EPR) com respeito à 'completude' da mecânica quântica (no marco da disputa entre Einstein e Niels Bohr sobre o status e a interpretação dessa teoria) a 'localidade' é considerada como um pressuposto (cf. [EPR35]). A noção de *localidade* consiste na impossibilidade de ação a distância: se dois sistemas físicos estão suficientemente distantes, nenhuma ação física num dos sistemas pode afetar o estado do outro sistema. Partindo desse pressuposto, e de alguns outros, EPR propõem um experimento mental por meio do qual chegam à conclusão de que a mecânica quântica só pode ser entendida como uma descrição 'incompleta' do mundo físico.

Em 1952, David Bohm, seguindo as idéias de Einstein, propôs completar a teoria quântica introduzindo 'variáveis ocultas' (cf. [Boh52]). A teoria proposta por Bohm tem o mesmo poder de predição da versão ortodoxa, e tem certas características não-locais.

Considerando algumas pressuposições plausíveis, John Bell demonstrou, em 1964, que qualquer teoria realista local para o experimento proposto em EPR deveria satisfazer certas desigualdades (hoje chamadas *desigualdades de Bell*), as quais são incompatíveis com as predições da mecânica quântica (cf. [Bel64]). A demonstração de Bell exclui, portanto, a possibilidade de 'completar' a teoria quântica através de uma teoria na qual seja válido o principio de localidade. Vários experimentos têm sido realizados para comprovar se desigualdades similares às propostas por Bell seriam violadas na realidade. Os resultados obtidos favorecem amplamente as predições da mecânica quântica. A demonstração de Bell pode também ser interpretada como postulando a necessidade de considerar teorias não-locais se se pretende obter teorias mais completas que preservem o poder preditivo da mecânica quântica.

Numa posição mais extrema, Sandú Popescu e Daniel Rohrlich em [PR94] propõem a não-localidade como um axioma da mecânica quântica e formulam a pergunta de se essa seria a única teoria que concilia não-localidade e causalidade.[1] Desde tal

[1]Agradeço ao professor Marcos de Oliveira pela indicação do texto de Popescu e Rohrlich, e por sugerir que os modelos paraconsistentes por nós propostos também contêm certas características de

perspectiva, e levando-se em conta a discussão anterior, os modelos de computação quântica podem então ser pensados como intrinsecamente não-locais, e talvez seja essa a propriedade crucial que pode permitir computações mais eficientes.

Por outro lado, o modelo de MTPNSs (definido na Seção 2.4) ao dar a possibilidade de estabelecer inter-relações entre diferentes elementos da máquina, elementos estes possivelmente distantes, permite também um certo tipo de ação a distância; semelhante ao obtido através dos estados emaranhados (de fato, no Capítulo 5, Seção 5.1, as inter-relações nesse modelo foram usadas para simular estados emaranhados). Nos L-circuitos (definidos na Seção 3.3), as 'restrições polinomiais' permitem também estabelecer inter-relações entre elementos distantes (como já foi mencionado na Seção 5.2), portanto pode também ser considerado um modelo de computação não local.

É interessante indicar que no primeiro modelo de MTPs (definido na Seção 2.3), onde não é possível criar inter-relações entre diferentes elementos da máquina, e no modelo de mbC-circuitos, o qual não contem restrições polinomiais que liguem os valores de diferentes variáveis, não existe nenhuma possibilidade de ação a distância. Esses modelos são precisamente os que resultam ter menor poder computacional com respeito à complexidade algorítmica (cf. capítulos 2 e 3).

Finalmente, consideramos que nosso trabalho propõe um caminho interessante para estudar questões de complexidade algorítmica desde uma nova perspectiva lógica, aplicável em particular à computação quântica. São vários os problemas e alternativas a explorar que ficam em aberto, e naturalmente esperamos continuar a desenvolver as idéias aqui propostas em trabalhos futuros.

não-localidade, o que será discutido adiante.

Apêndice A

Elementos de lógica paraconsistente e modal

No presente apêndice apresentamos uma breve introdução à lógica paraconsistente e à lógica modal, com a intenção de facilitar o entendimento de nossas propostas a leitores com menos conhecimento sobre este tipos de lógica. Para um estudo mais aprofundado desses temas recomendamos [CCM07] e [CP08], respectivamente.

Antes de definir o que é uma 'lógica' é preciso definir uma 'linguagem' através da qual serão expressas as fórmulas ou enunciados da lógica. A linguagem de uma lógica L é definida sobre uma *assinatura* $C = \{C_n\}_{n\in\mathbb{N}}$, onde C_n é um conjunto de símbolos que representam *conectivos* de aridade n. A *linguagem* ou conjunto de *fórmulas bem formadas (FBFs)* de L é definida escolhendo-se um conjunto de variáveis (usualmente um conjunto infinito enumerável) e definindo recursivamente as regras de formação das fórmulas.

No caso da *lógica proposicional clássica (CP)*, a linguagem pode ser definida considerando o conjunto de variáveis proposicionais $X = \{p_1, p_2, \ldots\}$ e a assinatura C com $C_1 = \{\neg\}$, $C_2 = \{\wedge, \vee, \rightarrow\}$ e $C_n = \emptyset$ para $n = 0$ e $n > 2$ (o conetivo unário $\neg$ é chamado *negação* e os conetivos binários $\wedge$, $\vee$ e $\rightarrow$ são chamados *conjunção*, *disjunção* e *implicação*, respectivamente). As FBF são definidas recursivamente, usando '(' e ')' como símbolos auxiliares, mediante as seguintes regras:

1. Se $\alpha \in X$ então α é uma FBF.
2. Se α e uma FBF então $\neg\alpha$ é uma FBF.
3. Se α e β são FBFs então $(\alpha \# \beta)$ é uma FBF, onde $\#$ é qualquer dos conectivos $\wedge$, $\vee$ ou $\rightarrow$.
4. FBFs somente podem ser geradas através das condições 1 a 3.

Vamos agora definir o que é uma lógica, de forma geral, com base na noção de 'relação de conseqüência'. Dado um conjunto For de fórmulas de uma linguagem, uma *relação de conseqüência* $\Vdash$ é um subconjunto de $\wp(For) \times For$, onde $\wp(For)$ representa o conjunto das partes de For. Uma relação de conseqüência é *tarskiana* se, para quaisquer formulas α e β, e conjuntos de fórmulas Γ e Δ de For, são satisfeitas as seguintes condições:[1]

$$\text{Reflexividade: } \alpha \in \Gamma \text{ implica } \Gamma \Vdash \alpha, \tag{Con1}$$

$$\text{Monotonicidade: } (\Delta \Vdash \alpha \text{ e } \Delta \subseteq \Gamma) \text{ implica } \Gamma \Vdash \alpha, \tag{Con2}$$

$$\text{Corte: } (\Delta \Vdash \alpha \text{ e } \Gamma, \alpha \Vdash \beta) \text{ implica } \Delta \cup \Gamma \Vdash \beta. \tag{Con3}$$

Neste apêndice vamos considerar somente relações de conseqüência tarskianas.

Um *lógica* (tarskiana) L é uma estrutura da forma $\langle For, \Vdash \rangle$, contendo um conjunto de fórmulas e uma relação de conseqüência (tarskiana) definida sobre essas fórmulas.

Uma lógica pode ser expressa de diversas maneiras; uma delas consiste em se definir um conjunto de fórmulas a serem consideradas como os *axiomas* do sistema, e definir *regras de inferência* que permitim deduzir algumas fórmulas a partir de outras. Esse método de definir uma lógica é chamado *método axiomático* (ou *hilbertiano*). Usualmente a regra de *substituição uniforme*, que consiste na substituição simultânea de todas as ocorrências de uma variável proposicional por uma fórmula, está implícita na definição dos sistemas axiomáticos (ou os axiomas realmente correspondem a *esquemas axiomáticos*, onde são usadas meta-variáveis para representar fórmulas arbitrárias).

[1] Do lado esquerdo da relação $\Vdash$, as fórmulas e conjuntos de fórmulas são usualmente separadas por ',' o que expressa a união dessas fórmulas.

A *lógica positiva clássica (CP^+)*, que é o maior fragmento de CP que não contém o conectivo de negação, pode ser definida através dos seguintes axiomas:

$$\alpha \to (\beta \to \alpha), \tag{Ax1}$$

$$(\alpha \to \beta) \to ((\alpha \to (\beta \to \gamma)) \to (\alpha \to \gamma)), \tag{Ax2}$$

$$\alpha \to (\beta \to (\alpha \wedge \beta)), \tag{Ax3}$$

$$(\alpha \wedge \beta) \to \alpha, \tag{Ax4}$$

$$(\alpha \wedge \beta) \to \beta, \tag{Ax5}$$

$$\alpha \to (\alpha \vee \beta), \tag{Ax6}$$

$$\beta \to (\alpha \vee \beta), \tag{Ax7}$$

$$(\alpha \to \gamma) \to ((\beta \to \gamma) \to ((\alpha \vee \beta) \to \gamma)), \tag{Ax8}$$

$$\alpha \vee (\alpha \to \beta). \tag{Ax9}$$

e a regra de inferência:

$$\frac{\alpha, \alpha \to \beta}{\beta}, \tag{MP}$$

chamada regra de *modus ponens*, que indica que a fórmula β pode ser deduzida das fórmulas α e $\alpha \to \beta$. CP pode ser definida estendendo a linguagem de CP^+ com um conectivo de negação $\neg$ e acrescentando o axioma:

$$\alpha \to (\neg\alpha \to \beta). \tag{Ax10}$$

Para definir o que é uma 'lógica paraconsistente' é necessário primeiro definir outros conceitos, o que faremos a seguir.

Uma *teoria* de uma lógica L é qualquer conjunto $\Gamma \subseteq For$. Uma teoria Γ, de uma lógica com operador de negação $\neg$, é *contraditória* se existir uma fórmula α tal que $\Gamma \Vdash \alpha$ e $\Gamma \Vdash \neg\alpha$. Por outro lado, uma teoria Γ é *trivial* se para toda fórmula α tem-se que $\Gamma \Vdash \alpha$.

Em CP é válido o *princípio de explosão*, que afirma que para toda teoria Γ de CP e toda fórmula α e β tem-se que $\Gamma, \alpha, \neg\alpha \Vdash \beta$. Portanto, em CP toda teoria contraditória é necessariamente trivial.

As *lógicas paraconsistentes* são definidas como lógicas que servem como base de teorias contraditórias mas não triviais, i.e. uma lógica L é paraconsistente se existir

uma teoria Γ e fórmulas α e β de L tais que $\Gamma \Vdash \alpha$, $\Gamma \Vdash \neg\alpha$ e $\Gamma \nVdash \beta$ (onde o símbolo $\nVdash$ denota a negação da relação $\Vdash$). Fica claro, então, que as lógicas paraconsistentes separam as noções de contraditoriedade e trivialidade com respeito às teorias lógicas.

As *lógicas da inconsistência formal (LFIs)* são uma hierarquia de lógicas paraconsistentes, definidas por Walter Carnielli e João Marcos (ver referências e detalhes em [CCM07]), que estendem CP^+ introduzindo um conectivo de negação (com características paraconsistentes) e conectivos unários $\circ$ e $\bullet$ que expressam *consistência* e *inconsistência*, respectivamente. Nas LFIs o princípio de explosão não é válido, mas é válido um *princípio de explosão controlada* que estabelece que para toda teoria Γ e toda fórmula α e β de uma lógica L tem-se que $\Gamma, \alpha, \neg\alpha, \circ\alpha \Vdash \beta$. O conetivo $\circ$ nas LFIs permite, então, representar uma noção de 'consistência' na linguagem objeto, e as propriedades do conectivo $\bullet$ são definidas (axiomaticamente) de maneira a se comportar de maneira dual ao conectivo de consistência. A idéia das LFIs é que permitem pensar na equação CONTRADIÇÃO + CONSISTÊNCIA = TRIVIALIDADE, ao contrário do que acontece em CP, onde CONTRADIÇÃO = TRIVIALIDADE.

Uma lógica fundamental na hierarquia das LFIs, chamada mbC, pode ser definida axiomaticamente estendendo CP^+ com os conectivos $\neg$, $\circ$ e $\bullet$, e acrescentando os seguintes axiomas:[2]

$$\alpha \vee \neg\alpha, \tag{Ax11}$$

$$\circ\alpha \rightarrow (\alpha \rightarrow (\neg\alpha \rightarrow \beta)). \tag{bc1}$$

Adicionalmente, em mbC tem-se as seguintes definições $\wr\alpha \stackrel{\text{def}}{=} (\neg\alpha \wedge \circ\alpha)$, $\bot \stackrel{\text{def}}{=} (\beta \wedge \wr\beta)$, $\sim \alpha \stackrel{\text{def}}{=} \alpha \rightarrow \bot$ e $\bullet\alpha \stackrel{\text{def}}{=} \sim \circ\alpha$.

Outra LFI que estamos usando neste trabalho, denominada $LFI1$, pode ser definida acrescentando a mbC os seguintes axiomas:

$$\neg \circ \alpha \rightarrow (\alpha \wedge \neg\alpha), \tag{ci}$$

$$\neg\neg\alpha \rightarrow \alpha, \tag{cf}$$

$$\alpha \rightarrow \neg\neg\alpha, \tag{ce}$$

[2]Os axiomas serão nomeados de acordo com as convenções em [CCM07].

$$\bullet\ (\alpha \wedge \beta) \leftrightarrow ((\bullet\alpha \wedge \beta) \vee (\bullet\beta \wedge \alpha)), \tag{cj1}$$

$$\bullet\ (\alpha \vee \beta) \leftrightarrow ((\bullet\alpha \wedge \neg\beta) \vee (\bullet\beta \wedge \neg\alpha)), \tag{cj2}$$

$$\bullet\ (\alpha \rightarrow \beta) \leftrightarrow (\alpha \wedge \bullet\beta), \tag{cj3}$$

onde $\alpha \leftrightarrow \beta \stackrel{\text{def}}{=} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

Esta breve introdução sobre as lógicas paraconsistentes, mais as referências sugeridas, é suficiente para um guia ao leitor não-especialista.

Vamos agora descrever brevemente em que consistem as lógicas modais.

As *lógicas modais* são extensões da lógica clássica obtidas através da introdução de conectivos que permitem expressar modos de qualificar à verdade, tais como *possibilidade* (usualmente representada pelo conectivo $\Diamond$) e *necessidade* (usualmente representada pelo conectivo $\Box$). Uma classe de lógicas modais, denominadas *lógicas modais normais*, é definida como a classe de lógicas nas quais é valido o axioma:

$$\Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta), \tag{K}$$

e a regra de inferência:

$$\vdash \alpha \text{ implica } \vdash \Box\alpha; \tag{Nec}$$

chamada *regra de necessitação*. Nessas lógicas o conectivo de possibilidade pode ser definido em termos dos conectivos de negação e necessitação: $\Diamond\alpha \stackrel{\text{def}}{=} \neg\Box\neg\alpha$.

A lógica modal normal mais fraca, que estende CP acrescentando o axioma (K) e a regra de necessitação é denominado K. Outras lógicas modais normais são obtidos mediante a introdução de novos axiomas. Alguns desses axiomas são os seguintes:

$$\Box\alpha \rightarrow \alpha, \tag{T}$$

$$\alpha \rightarrow \Box\neg\Box\neg\alpha, \tag{B}$$

$$\Box\alpha \rightarrow \Box\Box\alpha, \tag{4}$$

A lógica modal $S5$, usada em varias partes deste trabalho, pode ser definida acrescentando-se a K os axiomas (T), (B) e (4).

Até agora definimos os sistemas de lógica de uma maneira puramente sintática, sem considerar valores de verdade ou qualquer interpretação dos símbolos da linguagem. Uma lógica pode também ser expressa de maneira semântica, definindo a

relação de conseqüência mediante a interpretação dos símbolos da linguagem em alguma classe de estruturas matemáticas. Usualmente uma lógica é definida de ambas as formas: de maneira sintática e de maneira semântica. É preciso então demonstrar que a representação sintática e a representação semântica são equivalentes; isso se faz mostrando que: (1) a relação de conseqüência sintática é um subconjunto da relação de conseqüência semântica, nesse caso se diz que a representação sintática da lógica é *correta* com respeito à representação semântica; e (2) a relação de conseqüência semântica é um subconjunto da relação de conseqüência sintática, nesse caso se diz que a representação sintática da lógica é *completa* com respeito à representação semântica. Se uma representação semântica de uma lógica é equivalente a uma representação sintática, diz-se que essa semântica é *adequada* para essa lógica (ou que *caracteriza* essa lógica).

Um tipo particular de semântica, chamado *semântica de matrizes*, consiste em traduzir fórmulas da linguagem numa 'matriz'. Uma *matriz* é uma tripla $\mathcal{M} = \langle M, D, O\rangle$, onde M é um conjunto não vazio (usualmente os valores em M representam valores de verdade), D é um subconjunto de M denominado conjunto de *valores designados* (que representam os valores considerados como verdadeiros), e O é um conjunto de funções sobre M que são usadas para interpretar os conectivos lógicos (usualmente essas funções são representadas através de tabelas-verdade).

É fácil definir uma semântica de matrizes com dois valores de verdade para CP, e muitas outras lógicas podem ser caracterizadas por matrizes finitas (i.e. matrizes onde o conjunto M é finito). Contudo, não é possível caracterizar mbC e $S5$ (e outras lógicas paraconsistentes e modais) através de matrizes finitas tais que os valores de verdade sejam verofuncionais (isto é, tais que o valor de verdade de uma sentença complexa dependa funcionalmente dos valores das sentenças componentes (cf. [CCM07] e [CP08], respectivamente). Por exemplo, nas lógicas paraconsistentes, se 0 representa o valor-verdade 'falso' e 1 representa o valor-verdade 'verdadeiro', a negação $\neg$ não é verofuncional, pois o valor-verdade v é tal que $v(\neg p) = 1$ se $v(p) = 0$, mas funcionalmente indeterminado se $v(p) = 1$. Em outras palavras, $v(\neg p)$ não depende *sempre* funcionalmente de $v(p)$. De forma um tanto similar, nas lógicas modais temos que $v(\Box p)$ também não depende *sempre* funcionalmente de $v(p)$.

Apesar de que uma semântica não-verofuncional como acima é útil do ponto de vista meramente matemático, deixa a desejar da perspectiva semântica propriamente dita, já que não explica o funcionamento semântico dos conectivos. Outros tipos de semântica foram propostos para interpretar esse tipo de lógicas: no caso das lógicas paraconsistentes um tipo de semântica adequada é a chamada *semântica de traduções possíveis* (cf.[CCM07]), e no caso das lógicas modais uma das semânticas mais bem sucedidas é a chamada *semântica relacional* ou *semântica de mundos possíveis*, devida a Saul Kripke (cf. [CP08]).

Vamos fazer uma breve descrição da semântica relacional, necessária para se compreender a definição do modelo de MTPNSs (Seção 2.4.1).

Um *modelo relacional* é uma tripla $M = \langle W, R, v\rangle$, onde:

- W é um conjunto não vazio cujos elementos são chamados de *mundos possíveis*;
- R é uma relação binária em W chamada *relação de acessibilidade* (se $w_1, w_2 \in W$ e $(w_1, w_2) \in R$ se diz que w_2 é *acessível* a partir de w_1);
- e v: $Var \to \wp(W)$ (onde Var representa o conjunto de variáveis proposicionais da lógica) é uma função que associa variáveis proposicionais a mundos possíveis (a função v determina, para cada variável proposicional, em que mundos possíveis ela é verdadeira).

A noção de *verdade* (ou *satisfação*) de uma fórmula α (de uma lógica modal L) em um modelo relacional $M = \langle W, R, v\rangle$, no mundo $w \in W$ (denotado por $M, w \vDash \alpha$), é definida recursivamente por:

1. $M, w \vDash p$ se e somente se $w \in v(p)$.
2. $M, w \vDash \neg\alpha$ se e somente se $M, w \nvDash \alpha$.
3. $M, w \vDash \alpha \wedge \beta$ se e somente se $M, w \vDash \alpha$ e $M, w \vDash \beta$.
4. $M, w \vDash \alpha \vee \beta$ se e somente se $M, w \vDash \alpha$ ou $M, w \vDash \beta$.
5. $M, w \vDash \alpha \to \beta$ se e somente se $M, w \nvDash \alpha$ ou $M, w \vDash \beta$.
6. $M, w \vDash \Box\alpha$ se e somente se $M, w' \vDash \alpha$ para todo $w' \in W$ tal que $(w, w') \in R$.

7. $M, w \vDash \Diamond\alpha$ se e somente se existir $w' \in W$ tal que $(w, w') \in R$ e $M, w' \vDash \alpha$.

Uma fórmula α é *válida* em um modelo relacional M se for verdadeira em todos os mundos possíveis de M.

Algumas fórmulas da lógica modal estão associadas a propriedades da relação de acessibilidade. Por exemplo, numa classe de modelos relacionais apropriada para uma lógica modal onde a fórmula (T) seja um teorema, a relação de acessibilidade deve ser *reflexiva* (i.e. para todo mundo $w \in W$ deve ter-se que $(w, w) \in W$). Na classe de modelos relacionais adequados para a lógica $S5$ a relação de acessibilidade deve ser reflexiva, simétrica e transitiva.

Neste apêndice somente descrevemos sistemas de lógica em nível proposicional. Através da introdução de operadores de quantificação como $\forall$ ('para todo') e $\exists$ ('existe') podem ser definidas outras lógicas com maior poder expressivo, chamadas *lógicas de ordem superior*, a partir das *lógicas de primeira ordem* ou de *predicados*. A definição de lógicas paraconsistentes e modais de ordem superior e mesmo de primeira ordem está fora do escopo deste apêndice.

Apêndice B

Elementos de álgebra linear

No presente apêndice apresentamos alguns elementos básicos de álgebra linear, com a intenção de evitar ao leitor não iniciado a necessidade de consultar um livro de álgebra linear para o entendimento do Capitulo 4.

Definição B.1 (Corpo)**.** Um *corpo* é um conjunto C, que contem pelo menos dois elementos, munido de duas operações binárias: *adição* (denotada por $+$) e *multiplicação* (denotada por $\cdot$), tal que as operações satisfazem às propriedades:

1. Associatividade da adição: $(a + b) + c = a + (b + c)$, para todo $a, b, c \in C$.
2. Existência de identidade aditiva: existe um elemento em C, denotado por 0, tal que $a + 0 = 0 + a = a$, para todo $a \in C$.
3. Existência de inversos aditivos: para todo elemento $a \in C$ existe um elemento em C, denotado por $-a$, tal que $a + (-a) = (-a) + a = 0$.
4. Comutatividade da adição: $a + b = b + a$, para todo $a, b \in C$.
5. Associatividade da multiplicação: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, para todo $a, b, c \in C$.
6. Existência de identidade multiplicativa: existe um elemento em C, denotado por 1, tal que $a \cdot 1 = 1 \cdot a = a$, para todo $a \in C$.
7. Existência de inversos multiplicativos: para todo elemento $a \in C$, se $a \neq 0$ então existe um elemento em C, denotado por a^{-1}, tal que $a \cdot a^{-1} = a^{-1} \cdot a = 1$.

8. Comutatividade da multiplicação: $a \cdot b = b \cdot a$, para todo $a, b \in C$.

9. Distributividade da multiplicação em relação à adição: $c \cdot (a+b) = (c \cdot a) + (c \cdot b)$, para todo $a, b, c \in C$.

Definição B.2 (Espaço vetorial)**.** Seja C um corpo, cujos elementos serão chamados de *escalares*. Um *espaço vetorial* sobre C é um conjunto não vazio V, cujos elementos serão chamados de *vetores*, munido de duas operações: *adição vetorial* (denotada por $+$) e *multiplicação por um escalar* (denotada por justaposição). A adição vetorial atribui a cada par de vetores $(\mathbf{u}, \mathbf{v}) \in V \times V$ um vetor $\mathbf{u} + \mathbf{v}$ em V, enquanto a multiplicação por um escalar atribui a cada par $(a, \mathbf{u}) \in C \times V$ um vetor $a\mathbf{v}$ em V. Essas operações devem satisfazer às propriedades:

1. Associatividade da adição vetorial: $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$, para todo $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$.

2. Comutatividade da adição vetorial: $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$, para todo $\mathbf{u}, \mathbf{v} \in V$.

3. Existência de identidade para a adição vetorial: existe um elemento em V, denotado por $\mathbf{0}$, tal que $\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u}$, para todo $\mathbf{u} \in V$.

4. Existência de inversos para a adição vetorial: para todo elemento $\mathbf{u} \in V$ existe um elemento em V, denotado por $-\mathbf{u}$, tal que $\mathbf{u} + (-\mathbf{u}) = (-\mathbf{u}) + \mathbf{u} = \mathbf{0}$.

5. Distributividade da multiplicação por um escalar em relação à adição vetorial: $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$, para todo $a \in C$ e para todo vetor $\mathbf{u}, \mathbf{v} \in V$.

6. Distributividade da multiplicação por um escalar em relação à adição de escalares: $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$, para todo $a, b \in C$ e para todo $\mathbf{u} \in V$.

7. Compatibilidade da multiplicação por um escalar com a multiplicação de escalares: $(a \cdot b)\mathbf{u} = a(b\mathbf{u})$, para todo $a, b \in C$ e para todo $\mathbf{u} \in V$.

8. Existência de identidade para a multiplicação por um escalar: $1\mathbf{u} = \mathbf{u}$, para todo $\mathbf{u} \in V$, onde 1 denota a identidade multiplicativa em C.

Definição B.3 (Subespaço)**.** Um *subespaço* de um espaço vetorial V é um subconjunto S de V que satisfaz as condições de espaço vetorial com respeito às operações obtidas restringindo as operações de V a S.

Definição B.4 (Combinação linear)**.** Seja V um espaço vetorial sobre o corpo C; qualquer expressão da forma:

$$a_1\mathbf{v_1} + \ldots + a_n\mathbf{v_n},$$

onde $a_i \in C$ e $\mathbf{v_i} \in V$ para todo i, é chamada uma *combinação linear* dos vetores $\mathbf{v_1}, \ldots, \mathbf{v_n}$.

Definição B.5 (Independência linear)**.** Seja V um espaço vetorial sobre o corpo C. Um subconjunto não vazio de vetores S de V é dito *linearmente independente* se nenhum elemento de S pode ser expresso como combinação linear de outros vetores em S. Em outras palavras, se para todo $\mathbf{v_1}, \ldots, \mathbf{v_n}$ em S, tem-se que:

$$a_1\mathbf{v_1} + \ldots + a_n\mathbf{v_n} = \mathbf{0} \text{ implica que } a_1 = \ldots = a_n = 0,$$

com $a_1, \ldots, a_n \in C$. Se um subconjunto de vetores S de V não é linearmente independente então é *linearmente dependente*.

Definição B.6 (Subespaço gerado)**.** Seja V um espaço vetorial sobre o corpo C. O *subespaço gerado* por um subconjunto de vetores S de V, denotado por $span(S)$, é o conjunto de todas as combinações lineares de vetores de S. Isto é:

$$span(S) = \{a_1\mathbf{v_1} + \ldots + a_n\mathbf{v_n} \mid a_i \in C, v_i \in S\}.$$

Definição B.7 (Base)**.** Qualquer conjunto de vetores S de V que seja linearmente independente e que gere V (i.e., que $span(S) = V$) é chamado uma *base* de V.

É possível demonstrar que todas as bases de um mesmo espaço vetorial tem a mesma cardinalidade, isso permite definir a noção de dimensão de um espaço vetorial assim:

Definição B.8 (Dimensão)**.** A *dimensão* de um espaço vetorial V é a cardinalidade de qualquer base de V.

Pode ser provado que, se V é um espaço vetorial finito (i.e, de dimensão finita) n sobre um corpo C, então V é isomorfo a C^n (o espaço vetorial sobre C onde os vetores são n-uplas de elementos em C).

Definição B.9 (Operador linear)**.** Sejam V e W espaços vetoriais sobre o mesmo corpo C. Um *operador linear* é qualquer função $A\colon V \to W$ tal que:

$$A\left(\sum_{i=1}^{n} a_i \mathbf{u_i}\right) = \sum_{i=1}^{n} a_i A(\mathbf{u_i}),$$

para todo $n \geq 1$, todo $a_i \in C$ e todo $\mathbf{u_i} \in V$.

Definição B.10 (Representação matricial)**.** Sejam V e W espaços vetoriais finitos sobre o corpo C, e seja $A\colon V \to W$ um operador linear entre esses espaços. Suponha que $\mathbf{v_1}, \ldots, \mathbf{v_m}$ é uma base para V e que $\mathbf{w_1}, \ldots, \mathbf{w_n}$ é uma base para W. Então, para cada $j = 1, \ldots, m$, existem números $a_{1j} \in C$ até $a_{nj} \in C$ tais que:

$$A(\mathbf{v_j}) = \sum_{i=1}^{n} a_{ij} \mathbf{w_i}.$$

A matriz, de dimensão $n \times m$, cujas entradas são os valores a_{ij} é chamada de *representação matricial* do operador A.

Definição B.11 (Produto interno)**.** Seja V um espaço vetorial sobre o corpo C, onde $C = \mathbb{R}$ ou $C = \mathbb{C}$. Um *produto interno* em V é uma função $\langle \cdot, \cdot \rangle\colon V \times V \to C$ com as seguintes propriedades:

1. $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$, para todo $\mathbf{v} \in V$, e $\langle \mathbf{v}, \mathbf{v} \rangle = 0$ se e somente se $\mathbf{v} = \mathbf{0}$.
2. $\langle r\mathbf{u} + s\mathbf{v}, \mathbf{w} \rangle = r\langle \mathbf{u}, \mathbf{w} \rangle + s\langle \mathbf{v}, \mathbf{w} \rangle$, para todo $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ e para todo $a, b \in C$.
3. Se $C = \mathbb{R}$ então $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$ ou se $C = \mathbb{C}$ então $\langle \mathbf{u}, \mathbf{v} \rangle = \overline{\langle \mathbf{v}, \mathbf{u} \rangle}$, para todo $\mathbf{u}, \mathbf{v} \in V$.[1]

Definição B.12 (Norma e distância)**.** Seja V um espaço vetorial com produto interno e sejam $\mathbf{v}, \mathbf{w} \in V$. A *norma* do vetor $\mathbf{v}$ está definida por:

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}.$$

[1] Todo número complexo α pode ser escrito na forma $\alpha = a + bi$, e sua conjugada $\overline{\alpha}$ é dada por $\overline{\alpha} = a - bi$. Logo, $\overline{\langle \mathbf{v}, \mathbf{u} \rangle}$ representa a conjugada de $\langle \mathbf{v}, \mathbf{u} \rangle$.

E a *distância* entre os vetores $\mathbf{v}$ e $\mathbf{w}$ está definida por:

$$dist(\mathbf{v}, \mathbf{w}) = \|\mathbf{v} - \mathbf{w}\|.$$

Definição B.13 (Seqüência de Cauchy)**.** Seja V um espaço vetorial com produto interno, uma seqüência $\mathbf{v_1}, \mathbf{v_2}, \ldots$ de vetores em V é uma *seqüência de Cauchy* se:

$$lim_{i,i\to\infty}(dist(\mathbf{v_i}, \mathbf{v_j})) = 0.$$

Definição B.14 (Seqüência convergente)**.** Seja V um espaço vetorial com produto interno, uma seqüência $\mathbf{v_1}, \mathbf{v_2}, \ldots$ de vetores em V é *convergente* se existe um elemento $\mathbf{v} \in V$ tal que, para todo número real ϵ, existe um número natural N tal que $dist(\mathbf{v}, \mathbf{v_i}) < \epsilon$ para todo $i > N$.

Definição B.15 (Espaço de Hilbert)**.** Um *espaço de Hilbert* é um espaço vetorial com produto interno *completo*, i.e., onde toda seqüência de Cauchy é convergente.

Todo espaço vetorial com produto interno de dimensão finita é completo, portanto é um espaço de Hilbert.

Definição B.16 (Vetor dual)**.** Seja V um espaço com produto interno sobre o corpo C e seja $\mathbf{v} \in V$, o *vetor dual* de $\mathbf{v}$ é o operador linear $\hat{\mathbf{v}}$: $V \to C$ tal que $\hat{\mathbf{v}}(\mathbf{w}) = \langle \mathbf{v}, \mathbf{w} \rangle$, para todo $\mathbf{w} \in V$.

Definição B.17 (Transposta conjugada de uma matriz)**.** Seja A uma matriz de dimensão $n \times m$, cujos elementos a_{ij} são números complexos (o primeiro subíndice indica a fila, e o segundo a coluna, do elemento na matriz). A *transposta conjugada* de A, denotada por $A^\dagger$, é uma matriz de dimensão $m \times n$ com elementos $b_{ji} = \overline{a_{ij}}$, onde $\overline{a_{ij}}$ denota a transposta conjugada de a_{ij}.

É possível demonstrar que, se V é um espaço de Hilbert complexo (i.e., sobre o corpo $\mathbb{C}$) de dimensão n, a representação matricial de $\hat{\mathbf{v}}$ é o vetor-coluna de $\mathbb{C}^{n\times 1}$ que corresponde à transposta conjugada do vetor-linha $\mathbf{v}$, para todo $\mathbf{v} \in V$.

Definição B.18 (Notação de Dirac)**.** Na notação de Dirac:

1. Os vetores de um espaço de Hilbert V são denotados por $|\cdot\rangle$, e são chamados de *kets*.

2. O produto interno de $|\,\mathbf{v}\,\rangle$ com $|\,\mathbf{w}\,\rangle$ é denotado por $\langle\,\mathbf{v}\,|\,\mathbf{w}\,\rangle$.

3. Os vetores duais dos kets são denotados por $\langle\,\cdot\,|$, e são chamados de *bras*. Assim, o dual do vetor $|\,\mathbf{v}\,\rangle$ é denotado por $\langle\,\mathbf{v}\,|$, e $\langle\,\mathbf{v}\,|(|\,\mathbf{w}\,\rangle) = \langle\,\mathbf{v}\,|\,\mathbf{w}\,\rangle$ para todo $|\,\mathbf{w}\,\rangle \in V$.

4. Dados um ket $|\,\mathbf{v}\,\rangle$ e um bra $\langle\,\mathbf{w}\,|$, o *produto externo* de $|\,\mathbf{v}\,\rangle$ com $\langle\,\mathbf{w}\,|$ é o operador linear $|\,\mathbf{v}\,\rangle\langle\,\mathbf{w}\,|\colon V \to V$ tal que $|\,\mathbf{v}\,\rangle\langle\,\mathbf{w}\,|(|\,\mathbf{u}\,\rangle) = \langle\,\mathbf{w}\,|\,\mathbf{u}\,\rangle|\,\mathbf{v}\,\rangle$, para todo vetor $|\,\mathbf{u}\,\rangle$ de V. Observe que $\langle\,\mathbf{w}\,|\,\mathbf{u}\,\rangle|\,\mathbf{v}\,\rangle$ denota o produto do escalar $\langle\,\mathbf{w}\,|\,\mathbf{u}\,\rangle$ pelo ket $|\,\mathbf{v}\,\rangle$.

Definição B.19 (Ortogonalidade)**.** Seja V um espaço vetorial com produto interno. Dois vetores $\mathbf{u}, \mathbf{v} \in V$ são *ortogonais* se $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

Definição B.20 (Vetor unitário)**.** Seja V um espaço vetorial com produto interno e $\mathbf{v} \in V$ um vetor, se diz que $\mathbf{v}$ é um *vetor unitário* se $\|\mathbf{v}\| = 1$.

Definição B.21 (Base orto-normal)**.** Seja V um espaço vetorial com produto interno. Uma *base orto-normal* para V é uma base onde todos os elementos são unitários e onde todo par de elementos são ortogonais.

Definição B.22 (Auto-vetor, auto-valor e auto-espaço)**.** Seja A um operador linear sobre um espaço vetorial V ($A\colon V \to V$). Um *auto-vetor* de A é um vetor diferente de $\mathbf{0}$ tal que $A\mathbf{v} = v\mathbf{v}$, onde v é qualquer escalar. Neste caso v é chamado o *auto-valor* correspondente ao auto-vetor $\mathbf{v}$. O espaço gerado pelos auto-vetores de A é chamado de *auto-espaço* de A.

Definição B.23 (Representação diagonal)**.** Seja V um espaço vetorial com produto interno. A representação diagonal de um operador linear $A\colon V \to V$ é uma representação da forma:

$$A = \sum_i \lambda_i |\, i\,\rangle\langle\, i\,|,$$

onde os vetores $|\,i\,\rangle$ forman um conjunto orto-normal de auto-vetores de A, com auto-valores correspondentes λ_i. Observe que $|\,i\,\rangle\langle\,i\,|\colon V \to V$ é um operador linear (ver Definição B.18) para todo i.

Diz-se que um operador A é *diagonalizável* se tem representação diagonal.

Definição B.24 (Operador adjunto)**.** Seja A: $V \to V$ um operador linear sobre um espaço vetorial com produto interno V. Então existe um único operador linear A^*: $V \to V$ tal que:

$$\langle A(\mathbf{u}), \mathbf{v}\rangle = \langle \mathbf{u}, A^*(\mathbf{v})\rangle,$$

para todo $\mathbf{u}, \mathbf{v} \in V$. O operador A^* é chamado *operador adjunto* de A.

Se (a_{ij}) é a representação matricial de A (Definição B.10) então a representação matricial de A^* é dada pela matriz transposta conjugada $(a_{ij})^\dagger$ (Definição B.17).

Definição B.25 (Operador auto-adjunto ou Hermitiano)**.** Um operador linear A: $V \to V$ num espaço vetorial com produto interno V é chamado *auto-adjunto* ou *Hermitiano* se seu operador adjunto é ele mesmo, isto é, se $A^* = A$.

Definição B.26 (Operador unitário)**.** Um operador linear A: $V \to V$ num espaço vetorial com produto interno V é *unitário* se $A \circ A^* = I$, onde o símbolo $\circ$ representa a composição de operadores e I representa o operador identidade sobre o espaço de Hilbert V.

Definição B.27 (Produto tensorial de espaços vetoriais)**.** Sejam V e W espaços vetoriais de dimensão finita com bases $\{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$ e $\{\mathbf{w_1}, \ldots, \mathbf{w_m}\}$, respectivamente. O *produto tensorial* de V e W, denotado por $V \otimes W$, é um espaço vetorial de dimensão $n \cdot m$ gerado pelo conjunto $\{(\mathbf{v_i}, \mathbf{w_j}) : \mathbf{v_i} \in V, \mathbf{w_j} \in W\}$, e com produto interno dado por $\langle(\mathbf{v_i}, \mathbf{w_j}), (\mathbf{v_k}, \mathbf{w_l})\rangle = \langle \mathbf{v_i}, \mathbf{v_k}\rangle\langle \mathbf{w_j}, \mathbf{w_l}\rangle$, para todo $\mathbf{v_i}, \mathbf{v_k} \in V$ e $\mathbf{w_j}, \mathbf{w_l} \in W$.

Definição B.28 (Produto tensorial de vetores)**.** Sejam V e W espaços vetoriais de dimensão finita com bases $\{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$ e $\{\mathbf{w_1}, \ldots, \mathbf{w_m}\}$, respectivamente. O *produto tensorial* dos vetores $\mathbf{v} = \sum_{i=1}^{n} a_i \mathbf{v_i} \in V$ e $\mathbf{w} = \sum_{j=1}^{m} b_j \mathbf{w_j} \in W$, denotado por $\mathbf{v} \otimes \mathbf{w}$, é dado por:

$$\mathbf{v} \otimes \mathbf{w} = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j (\mathbf{v_i}, \mathbf{w_j}).$$

Definição B.29 (Produto tensorial de matrizes)**.** Sejam $\mathbf{u}$ uma matriz de dimensão $m \times n$ e $\mathbf{v}$ uma matriz de dimensão $k \times l$, o *produto tensorial* de $\mathbf{u}$ e $\mathbf{v}$, denotado por

$\mathbf{u} \otimes \mathbf{v}$, é uma matriz de dimensão $mk \times nl$ obtida assim:

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_{11}\mathbf{v} & u_{12}\mathbf{v} & \dots \\ u_{21}\mathbf{v} & u_{22}\mathbf{v} & \\ \vdots & & \ddots \end{bmatrix} = \begin{bmatrix} u_{11}v_{11} & u_{11}v_{12} & \dots & u_{12}v_{11} & u_{12}v_{12} & \dots \\ u_{11}v_{21} & u_{11}v_{22} & & u_{12}v_{21} & u_{12}v_{22} & \\ \vdots & & \ddots & & & \\ u_{21}v_{11} & u_{21}v_{12} & & & & \\ u_{21}v_{21} & u_{21}v_{22} & & & & \\ \vdots & & & & & \end{bmatrix}$$

Referências Bibliográficas

[AB06] Anders, Simon e Hans J. Briegel: *Fast Simulation of Stabilizer Circuits Using a Graph-state Representation*. Physical Review A, 73(022334):1–9, 2006. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/0504117`.

[AB09] Arora, Sanjeev e Boaz Barak: *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[AC07] Agudelo, Juan C. e Walter Carnielli: *Unconventional Models of Computation Through Non-standard Logic Circuits*. Lecture Notes in Computer Science, 4618:29–40, 2007.

[AC09] Agudelo, Juan C. e Walter Carnielli: *Paraconsistent Machines and their Relation to Quantum Computing*. Journal of Logic and Computation, 2009. Versão preliminar disponível em `http://arxiv.org/abs/0802.0150`.

[Agu06] Agudelo, Juan C.: *Da Computação Paraconsistente à Computação Quântica*. Tese de Mestrado, Universidade Estadual de Campinas- UNICAMP, 2006.

[Aha98] Aharonov, Dorit: *Quantum Computation*. In Stauffer, Dietrich (editor): *Annual Review of Computational Physics*, volume VI. 1998. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/9812037`.

[Aha03] Aharonov, Dorit: *A Simple Proof that Toffoli and Hadamard are Quantum Universal*. Versão preliminar disponível em `http://arxiv.org/abs/quant-ph/0301040`, 2003.

[AKS04] Agrawal, Manindra, Neeraj Kayal, e Nitin Saxena: *Primes is in P*. Annals of Mathematics, 160(2):781–783, 2004. Disponível em `http://annals.math.princeton.edu/issues/2004/Sept2004/Agrawal.pdf`.

[Bat05] Battilotti, Giulia: *Basic logic and quantum computing: logical judgements by an insider observer*. International Journal of Quantum Information, 3(1), 2005. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/0407057`.

[BBBV97] Bennett, Charles H., Ethan Bernstein, Gilles Brassard, e Umesh Vazirani: *Strengths and Weaknesses of Quantum Computing*. SIAM Journal on Computing, 26(5):1510–1523, 1997.

[BBHT98] Boyer, Michel, Gilles Brassard, Peter Høyer, e Alain Tapp: *Tight Bounds on Quantum Searching*. Fortschritte der Physik, 46(4–5):493–505, 1998. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/9605034v1`.

[Büc62] Büchi, J. Richard: *Turing-machines and the Entscheidungsproblem*. Mathematische Annalen, 148:201–213, 1962.

[Bel64] Bell, John S.: *On the Einstein-Podolsky-Rosen Paradox*. Physics, 1:195–200, 1964.

[Ber98] Bernstein, Daniel J.: *Detecting perfect powers in essentially linear time*. Mathematics of Computation, 67(223):1253–1283, 1998.

[Ber08] Berkovitz, Joseph: *Action at a Distance in Quantum Mechanics*. In Zalta, Edward N. (editor): *The Stanford Encyclopedia of Philosophy*. 2008. Disponível em `http://plato.stanford.edu/archives/win2008/entries/qm-action-distance/`.

[Béz02] Béziau, Jean Yves: *S5 is a paraconsistent logic and so is first-order classical logic.* Logical Studies, (9):301–309, 2002.

[Béz05] Béziau, Jean Yves: *Paraconsistent logic from a modal viewpoint.* Journal of Applied Logic, 3:7–14, 2005.

[BG81] Bennett, Charles H. e John T. Gill: *Relative to a Random Oracle A, $P^A \neq NP^A \neq co - NP^A$ with Probability 1.* SIAM Journal on Computing, 10(1):96–113, 1981.

[BHT98] Brassard, Gilles, Peter Høyer, e Alain Tapp: *Quantum Counting.* Lecture Notes in Computer Science, 1443:820–831, 1998. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/9805082`.

[BJ89] Boolos, George e Richard Jeffrey: *Computability and logic.* Cambridge University Press, 3rd. edição, 1989.

[Boh52] Bohm, David: *A suggested interpretation of the quantum theory in terms of hidden variables I and II.* Physics Review, 85:166–193, 1952.

[Bra98] Braunstein, Samuel L.: *Error Correction for Continuous Quantum Variables.* Physical Review Letters, 80(18):4084–4087, 1998.

[BV93] Bernstein, Ethan e Umesh Vazirani: *Quantum complexity theory.* Proc. of the 1993 ACM Symposium on Theory of Computing, páginas 11–20, 1993.

[BV97] Bersntein, Ethan e Umesh Vazirani: *Quantum complexity theory.* SIAM Journal on Computing, 26(5):1411–1473, 1997.

[Cal07] Calude, Christian S.: *De-quantizing the solution of Deutsch's problem.* International Journal of Quantum Information, 5:409–415, 2007. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/0610220`.

[Car90] Carnielli, Walter A.: *Many-valued logics and plausible reasoning.* In *Proceedings of the Twentieth International Symposium on Multiple-Valued*

Logic, páginas 328–335, Charlotte, NC, USA, 1990. IEEE Computer Society.

[Car05] Carnielli, Walter A.: *Polynomial Ring Calculus for Many-valued Logics*. In Werner, B. (editor): *Proceedings of the 35th International Symposium on Multiple-Valued Logic*, páginas 20–25. IEEE Computer Society, 2005. Versão preliminar disponível em *CLE e-Prints* vol 5, n. 3, 2005: `http://www.cle.unicamp.br/e-prints/vol\_5,n\_3,2005.html`.

[CCM07] Carnielli, Walter A., Marcelo E. Coniglio, e João Marcos: *Logics of Formal Inconsistency*. In Gabbay, D. e F. Guenthner (editores): *Handbook of Philosophical Logic*, volume 14, páginas 15–107. Springer, 2nd edição, 2007. Versão preliminar disponível em *CLE e-Prints* vol 5, n. 1, 2005. `http://www.cle.unicamp.br/e-prints/vol\_5,n\_1,2005.html`.

[CD08] Carnielli, Walter A. e Francisco A. Doria: *Are the Foundations of Computer Science Logic Dependent?* In Dégremont, C., L. Keiff, e H. Rückert (editores): *Dialogues, Logics and Other Strange Things - Essays in Honour of Shahid Rahman*, páginas 87–107. Londres: College Publications, 2008.

[Cdd00] Carnielli, Walter A., João M. de Almeida, e Sandra de Amo: *Formal inconsistency and evolutionary databases*. Logic and logical philosophy, páginas 115–152, 2000.

[CE05] Carnielli, Walter A. e Richard L. Epstein: *Computabilidade, funções computáveis, lógica e os fundamentos da Matemática*. São Paulo: Editora UNESP, 2005.

[CEMM98] Cleve, Richard, Artur Ekert, Chiara Macchiavello, e Michele Mosca: *Quantum algorithms revisited*. Proceedings of the Royal Society of London. Series A, 454:339–354, 1998.

[Chu36a] Church, Alonso: *A note on the Entscheidungsproblem*. Journal of Symbolic Logic, 1(1):40–41, 1936.

[Chu36b] Church, Alonso: *An unsolvable problem of elementary number theory.* American Journal of Mathematics, 58:345–363, 1936.

[CK02] Clote, Peter e Evangelos Kranakis: *Boolean Functions and Computation Models.* Springer-Verlag, 2002.

[Cop08] Copeland, Jack: *The Modern History of Computing.* In Zalta, Edward N. (editor): *The Stanford Encyclopedia of Philosophy.* 2008. Disponível em `http://plato.stanford.edu/archives/fall2008/entries/computing-history/`.

[Cot03] Cotogno, Paolo: *Hypercomputation and the Physical Church-Turing Thesis.* British Journal for the Philosophy of Science, 54:181–223, 2003.

[CP08] Carnielli, Walter e Claudio Pizzi: *Modalities and Multimodalities.* Springer, 2008.

[Dav82] Davis, Martin: *Computability and unsolvability.* New York: Dover Publications, Inc., 1982.

[dD95] da Costa, Newton C. A. e Francisco A. Doria: *On Jaśkowski's Discussive Logics.* Studia Logica, 54(1):33–60, 1995.

[Deu85] Deutsch, David: *Quantum theory, the Church-Turing principle and the universal quantum computer.* Proceedings of the Royal Society of London. Series A, 400:97–117, 1985. Disponível em `http://www.qubit.org/resource/deutsch85.ps`.

[Deu89] Deutsch, David: *Quantum computational networks.* Proceedings of the Royal Society of London. Series A, 425:73–90, 1989.

[DG02] Dalla Chiara, María Luisa e Roberto Giuntini: *Quantum Logics.* In *Handbook of Philosophical Logic*, volume 6, páginas 129–228. Netherlands: Kluwer Academics Publishers, 2002.

[DGL03] Dalla Chiara, Maria Luisa, Roberto Giuntini, e Roberto Leporini: *Computational Logics. A Survey.* In Hendricks, Vincent F. e Jacek Malinowski

(editores): *Trends in Logic: 50 Years of Studia Logica*, páginas 229–271. Kluwer Academic Publishers, 2003. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/0305029`.

[DJ92] Deutsch, David e Richard Jozsa: *Rapid solution of problems by quantum computation.* Proceedings of the Royal Society of London. Series A, 439:553–558, 1992.

[dK04] da Costa, Newton C. A. e Décio Krause: *Complementarity and paraconsistency.* In Rahman, S., J. Symons, D. M. Gabbay, e J. P. Bendegen (editores): *Logic, Epistemology, and the Unity of Science*, volume 1. Springer, 2004.

[dK06] da Costa, Newton C. A. e Décio Krause: *The Logic of Complementarity.* In Benthem, J. van, G. Heinzmann, M. Rebuschi, e H. Visser (editores): *The Age of Alternative Logics: Assessing Philosophy of Logic and Mathematics Today*, páginas 103–120. Springer, 2006. Versão preliminar disponível em: `http://philsci-archive.pitt.edu/archive/00001559`.

[EJ96] Ekert, Artur e Richard Jozsa: *Quantum Computation and Shor's Factoring Algorithm.* Reviews of Modern Physics, 68(3):733–753, 1996.

[EPR35] Einstein, Albert, Boris Podolsky, e Nathan Rosen: *Can quantum-mechanical description of physical reality be considered complete?* Physical Review, 47:777–780, 1935.

[Eve57] Everett, Hugh: *Relative state formulation of quantum mechanics.* Reviews of Modern Physics, 29, 1957. Tesis de doutorado, Princenton University: reimpreso em *The Many-Worlds Interpretation of Quantum Mechanics*, B. DeWitt e N. Graham, eds., Princenton University Press, 1973; The theory of the universal wave function, p. 1–140.

[FBM+06] Freeth, T., Y. Bitsakis, X. Moussas, J. H. Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey, M. Allen, A. Crawley, P. Hockley, T. Malzbender, D. Gelb, W.Ambrisco, e M. G.

Edmunds: *Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism.* Nature, 444(30):587–591, 2006.

[Fey82] Feynman, Richard P.: *Simulating Physics with Computers.* International Journal of Theoretical Physics, 21:467–488, 1982.

[For03] Fortnow, Lance: *One Complexity Theorist's View of Quantum Computing.* Theoretical Computer Science, 292:597–610, 2003.

[Gan80] Gandy, Robin: *Church's thesis and principles for mechanisms.* In *The Kleene symposium*, páginas 123–148. The Kleene symposium, North-Holland, 1980.

[Gan88] Gandy, Robin: *The confluence of ideas in 1936.* In Herken, R. (editor): *The universal Turing machine: a half-century survey*, páginas 51–102. Springer-Verlag, New York, 1988.

[Göd34] Gödel, Kurt: *On undecidable propositions of formal mathematical systems.* In *[Göd86]*, páginas 346–369. 1934.

[Göd36] Gödel, Kurt: *On the length of proofs.* In *[Göd86]*, páginas 397–399. 1936.

[Göd64] Gödel, Kurt: *'Postscriptum' to [Göd34].* In *[Göd86]*, páginas 369–371. 1964.

[Göd86] Gödel, Kurt: *Collected works*, volume I: Publications 1929–1936. Oxford University Press, 1986.

[Gen05] Genovese, Marco: *Research on hidden variable theories: A review of recent progresses.* Physics Reports, 413:319–396, 2005.

[Gro96] Grover, Lov K.: *A fast quantum mechanical algorithm for database search.* In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, páginas 212–219, New York, NY, USA, 1996. ACM Press, ISBN 0-89791-785-5.

[Gru99] Gruska, Jozef: *Quantum computing*. Cambridge: McGraw-Hill International (UK) Limited, 1999.

[Hil71] Hillgarth, Jocelyn N.: *Ramon Lull and Lullism in Fourteenth-Century France*. Clarendon Press (Oxford), 1971.

[Jam74] Jammer, Max: *The Philosophy of Quantum Mechanics*. John Wiley & Sons, inc., 1974.

[Jaś48] Jaśkowski, Stanisław: *Rachunek zdań dla systemów dedukcyjnych sprzecznych*. Studia Societatis Scientiarun Torunesis, Sectio A, I(5):57–77, 1948. Traduzido ao inglês como 'A propositional calculus for inconsistent deductive systems' in *Logic and Logic Philosophy*, 7:35-56, 1999.

[Jaś49] Jaśkowski, Stanisław: *O koniunkcji dedukcyjnej w rachunku zdań dla systemów dedukcyjnych sprzecznych*. Studia Societatis Scientiarun Torunesis, Sectio A, I(8):171–172, 1949. Translated into English as 'On discussive conjunction in the propositional calculus for inconsistent deductive systems' in *Logic and Logic Philosophy*, 7:57-59, 1999.

[Joz98] Jozsa, Richard: *Entanglement and Quantum Computation*. Oxford University Press, January 1998. Versão preliminar disponível em `http://arxiv.org/abs/quant-ph/9707034`.

[LB99] Lloyd, Seth e Samuel L. Braunstein: *Quantum Computation over Continuous Variables*. Physical Review Letters, 82(8):1784–1787, 1999.

[Man80] Manin, Yuri: *Computable and Uncomputable (in Russian)*. Sovetskoye Radio, Moscow, 1980.

[Mar99] Marcos, João: *Semânticas de Traduções Possíveis*. Tese de Mestrado, Universidade Estadual de Campinas, Instituto de Filosofia e Ciências Humanas, 1999.

[Mar05a] Marcos, João: *Modality and paraconsistency.* In Bilkova, M. e L. Behounek (editores): *The Logica Yearbook*, páginas 213–222. Prague: Filosofia, 2005.

[Mar05b] Marcos, João: *Nearly every normal modal logic is paranormal.* Logique et Analyse, 48(189/182):279–300, 2005.

[Moo67] Moody, Ernest: *History of Logic.* In Edwards, Paul (editor): *Encyclopedia of Philosophy*, volume 8, página 530. New York: Macmillan, 1967.

[MS95] Mundici, Daniele e Wilfried Sieg: *Paper machines.* Philosophia Mathematica, 3(3):5–30, 1995.

[MS04] Mateus, Paulo e Amílcar Sernadas: *Exogenous Quantum Logic.* In *Proc. of CombLog'2004, Workshop on Combination of Logics: Theory and Applications*, páginas 142–149. CLC, Department of Mathematics, IST, Lisbon, Portugal, 2004. Disponível em: `http://wslc.math.ist.utl.pt/ftp/pub/SernadasA/04-MS-fiblog24s.pdf`.

[NC00] Nielsen, Michael A. e Isaac L. Chuang: *Quantum Computation and Quantum Information.* Cambridge: Cambridge University Press, 2000.

[Odi89] Odifreddi, Piergiorgio: *Classical recursion theory. The theory of functions and sets of natural numbers.* Studies in logic and the foundations of mathematics, volume 125. Amsterdam: North-Holland, 1989.

[ON00] Ozawa, Masanao e Haramichi Nishimura: *Local transition function of quantum Turing machines.* Theoretical Informatics and Applications, 34:379–402, 2000. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/9811069`.

[Oza98] Ozawa, Masanao: *Quantum Turing machines: local transitions, preparation, measurement and halting.* Disponível em: `http://arxiv.org/abs/quant-ph/9811069`, 1998.

[Pap94] Papadimitriou, Christos: *Computational Complexity*. Adisson-Wesley, 1994.

[Pit90] Pitowsky, Itamar: *The physical Church thesis and physical computational complexity*. Iyyun, A Jerusalem Philosophical Quarterly, 39:81–99, 1990.

[Pos36] Post, Emil: *Finite Combinatory Process-Formulation 1*. The Journal of Simbolic Logic, 1(3):103–105, 1936.

[PR94] Popescu, Sandu e Daniel Rohrlich: *Quantum nonlocality as an axiom*. Foundations of Physics, 24(3):379–385, 1994.

[Sav98] Savage, John E.: *Models of Computation: Exploring the Power of Computing*. Addison-Wesley, 1998.

[SB99] Sieg, W. e J. Byrnes: *An abstract model for parallel computations*. The Monist, 82:150–164, 1999.

[SC00] Sylvan, Richard e Jack Copeland: *Computability is Logic-Relative*. In Priest, Graham e Dominic Hyde (editores): *Sociative logics and their applications: essays by the late Richard Sylvan*, páginas 189–199. London: Ashgate Publishing Company, 2000.

[Sho94] Shor, Peter W.: *Algorithms for Quantum Computation: Discrete Log and Factoring*. In *Proc. 35th Symposium on Foundations of Computer Science*, páginas 124–134. IEEE Computer Society Press, 1994.

[Sho97] Shor, Peter W.: *Polinomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM Journal on Computing, 26(5):1484–1509, 1997.

[Sie94] Sieg, Wilfried: *Mechanical procedures and mathematical experience*. In George, A. (editor): *Mathematics and Mind*, Logic and Computation in Philosophy, páginas 71–117. Oxford University Press, 1994. Versão preliminar disponível em: `http://www.hss.cmu.edu/philosophy/techreports/26_Sieg.pdf`.

[Sie97] Sieg, Wilfried: *Step by recursive step: Church's analysis of effective calculability.* Bulletin of Symbolic Logic, 3(2):154–180, 1997.

[Sie02] Sieg, Wilfried: *Calculations by man & machine: Conceptual analysis.* In Sieg, W., R. Sommer, e C. Talcott (editores): *Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman*, páginas 396–415. A K Peters, 2002. Versão preliminar disponível em: `http://www.hss.cmu.edu/philosophy/techreports/104_Sieg.pdf`.

[Sie06] Sieg, Wilfried: *Gödel on computability.* Philosophia Mathematica, 14:189–207, 2006.

[Sie08] Sieg, Wilfried: *Church without Dogma: Axioms for computability.* In Lowe, B., A. Sorbi, e B. Cooper (editores): *New Computational Paradigms*, páginas 139–152. Springer Verlag, 2008. Versão preliminar disponível em: `http://www.hss.cmu.edu/philosophy/sieg/Church\%20without\%20dogma.pdf`.

[Sim97] Simon, Daniel R.: *On the Power of Quantum Computation.* SIAM Journal on Computing, 26(5):1474–1483, 1997.

[SM02] Stockmeyer, Larry e Albert R. Meyer: *Cosmological Lower Bound on the Circuit Complexity of a Small Problem in Logic.* Journal of the ACM, 49(6):753–784, 2002.

[Sma98] Smale, Steve: *Mathematical Problems for the Next Century.* Mathematical Intelligencer, 20:7–15, 1998.

[Soa96] Soare, Robert I.: *Computability and recursion.* The Bulletin of Symbolic Logic, 2(3):284–321, 1996. Disponível em: `http://www.math.ucla.edu/~asl/bsl/0203/0203-002.ps`.

[Sta04] Stannett, Mike: *Hypercomputational Models.* In *[Teu04]*, páginas 135–157. 2004.

[TD02] Terhal, Barbara M. e David P. DiVincenzo: *Classical Simulation of Noninteracting-fermion Quantum Circuits.* Physical Review A, 65(032325):1–10, 2002. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/0108010`.

[Teg98] Tegmark, Max: *The Interpretation of Quantum Mechanics: Many Worlds or Many Words?* Fortschritte der Physik, 46:855–862, 1998. Versão preliminar disponível em: `http://arxiv.org/abs/quant-ph/9709032`.

[Teu04] Teuscher, Christof (editor): *Alan Turing: Life and Legacy of a Great Thinker.* Springer, 2004.

[TMR53] Tarski, Alfred, Andrzej Mostowski, e Raphael M. Robinson: *Undecidable Theories.* In Brouwer, L. E. J., E. W. Beth, e A. Heyting (editores): *Studies in Logic and the Foundations of Mathematics.* North-Holland, 1953.

[Tur36] Turing, Alan M.: *On computable numbers, with an application to the Entscheidungsproblem.* Proceedings of the London Mathematical Society, 42(2):230–265, 1936. A correction, ibid., 43:544–546, 1937.

[Vai02] Vaidman, Lev: *Many-Worlds Interpretation of Quantum Mechanics.* In Zalta, Edward N. (editor): *The Stanford Encyclopedia of Philosophy.* 2002. Disponível em `http://plato.stanford.edu/archives/sum2002/entries/qm-manyworlds/`.

[van90] van Emde Boas, Peter: *Machine Models and Simulations.* In Leeuwen, J. van (editor): *Handbook of Theoretical Computer Science.* Elsevier Science Publishers B.V., 1990.

[VS00] Vélez, Mario e Andrés Sicard: *Sobre un modelo de computación cuántica sobre variables continuas.* Disponível em: `http://sigma.eafit.edu.co:90/~asicard/archivos/ccc.ps.gz`, 2000.

[Wol85] Wolfram, Stephen: *Undecidability and Intractability in Theoretical Physics.* Physical Review Letters, 54(8):735–738, 1985.

[Yao93] Yao, Andrew C.: *Quantum circuit complexity.* In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, páginas 352–360. IEEE Computer Society Press, Los Alamitos, CA, 1993.