UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MÔNICA XAVIER PY

Análise da Máquina de Turing Persistente com Múltiplas Fitas de Trabalho

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Tiarajú Asmuz Diverio
Orientador

Prof. Dr. Antônio Carlos da Rocha Costa
Co-orientador

Porto Alegre, maio de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Py, Mônica Xavier

Análise da Máquina de Turing Persistente com Múltiplas Fitas de Trabalho / Mônica Xavier Py. – Porto Alegre: PPGC da UFRGS, 2003.

74 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Tiarajú Asmuz Diverio; Co-orientador: Antônio Carlos da Rocha Costa.

1. Máquina de Turing Persistente Paralela. 2. Máquina de Turing. 3. Teoria da Computação. 4. Teoria da Computação Interativa. 5. Máquina de Turing Persistente. I. Diverio, Tiarajú Asmuz. II. Costa, Antônio Carlos da Rocha. III. Título.

“Este trabalho é dedicado à mãe Imara,
ao meu irmão Diogo e a
minha Vó Heride.”

AGRADECIMENTOS

Gostaria de agradecer a todos aqueles que, em contextos diferentes, contribuíram para que chegasse até aqui. Na impossibilidade de citar todas as pessoas, gostaria de destacar alguns nomes, que de alguma forma direta ou indiretamente contribuíram para o desenvolvimento desse trabalho. Agradeço especialmente aqueles que estiveram diariamente envolvidos nesse trabalho.

- Aos meus familiares, especialmente a minha mãe, pelo apoio, carinho e estímulo dado ao longo desses anos
- Ao meu orientador, o Prof. Dr. "general" Tiarajú Asmuz Diverio, por sua orientação, a minha admiração e agradecimento pelo apoio constante, pela confiança, amizade e, principalmente pelo incentivo dedicado durante todo o decorrer do curso
- Ao meu co-orientador, o Prof. Dr. Antônio Carlos da Rocha Costa, pelas contribuições que enriqueceram esse estudo, seu apoio foi indispensável à realização deste trabalho
- Aos meus co-co-orientadores Carlos Holbig e Marilton Sanchotene de Aguiar, por aguentar as seguidas incomodações no ICQ :-)
- Ao grupo GMCPAD, especialmente ao Diego, Sanger, Clarissa e Rúbia
- Aos amigos: Cadinho, SAC, Jú (bixxu), Márcia, Wives, Patty, Mozart, Chatilla, Delcino, Martinotto e Júlio, pois as conversas e cafés foram fundamentais para o progresso do trabalho
- Aos amigos e aos colegas aqui do II, pela amizade e incentivo nos momentos difíceis, os meus sinceros agradecimentos
- Ao pessoal do Instituto de Informática, funcionários e professores, agradeço especialmente a Eli (Portaria), Astrogildo (Segurança), Bea e Ida (Biblioteca)
- Ao CNPq pela bolsa de estudos concedida, tornando possível a realização desse trabalho

Por fim, gostaria de agradecer ao Rafael, "mon mari", pelo estímulo, carinho, amor, e por me aguentar 24 horas por dia!!!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 8

LISTA DE SíMBOLOS 9

LISTA DE FIGURAS 10

LISTA DE TABELAS 11

RESUMO 12

ABSTRACT 13

1 INTRODUÇÃO 15
1.1 A Teoria da Computação 15
1.2 A Teoria da Computação Interativa 17
1.3 Motivação do trabalho 17
1.4 Objetivos e Contribuições 18
1.5 Estrutura do Texto 19

2 NOÇÕES DE COMPUTABILIDADE 21
2.1 Teoria da Computação Clássica 21
2.2 Tese de Church 22
2.3 Algoritmos e Procedimentos efetivos 23
2.4 Modelos de computação 23
2.4.1 Máquina de Turing 24
2.4.2 Formalização do modelo 25
2.4.3 Máquinas de Turing como Máquina Universal 26
2.5 Outras abordagens à computabilidade 27
2.6 Máquinas Equivalentes à Máquina de Turing 27
2.7 Modificações sobre a Máquina de Turing 28
2.8 Máquina de Turing Paralela 29
2.8.1 Analogia da Classificação de Flynn nas Máquinas de Turing 30

3 TEORIA DA COMPUTAÇÃO INTERATIVA 33
3.1 Evolução da Computação 33
3.1.1 Mudança de Paradigma 34
3.1.2 Características da Computação Interativa 35
3.2 Noções para Computação Interativa 35
3.2.1 Embasamento conceitual 36

3.3 Máquinas para Computação Interativa . . . 37
3.4 Máquina Interativa Seqüencial — SIM . . . 37
3.4.1 Expressividade e Comportamento Interativo da SIM . . . 38
3.5 Máquina Interativa Multi seqüencial — MIM . . . 40
3.6 Considerações sobre as Máquinas Interativas . . . 41

4 MODELOS DE INTERAÇÃO . . . 43
4.1 A Computação Interativa de Wegner . . . 43
4.2 Máquina de Turing Persistente — PeTM . . . 44
4.2.1 Computação na PeTM . . . 44
4.2.2 Seqüência de interações na PeTM . . . 45
4.3 Modelos de Comportamento da PeTM . . . 46
4.4 Modelos baseados em linguagem da computação da PeTM . . . 47
4.4.1 Linguagem da PeTM . . . 47
4.4.2 PeTM com amnésia e TM . . . 47
4.4.3 PeTM com memória finita e estado finito . . . 48
4.5 Modelo de comportamento da PeTM baseado em autômato . . . 49
4.5.1 Autômato com estado finito e seu índice . . . 49
4.5.2 PeTM e FST . . . 50
4.5.3 PeTM e LTS . . . 51
4.6 Modelando comportamento da PeTM por Função definida recursivamente . . . 51
4.7 Abordagem baseada em ambiente para Comportamento da PeTM . . . 52
4.7.1 Equivalência e Expressividade relativas . . . 53
4.7.2 Ambientes finitos de PeTM . . . 54
4.7.3 Hierarquia de expressividade infinita . . . 55
4.8 Considerações sobre a PeTM . . . 56

5 FORMALIZAÇÃO DA PETM-PAR . . . 57
5.1 Máquina de Turing Persistente Paralela — PeTM-Par . . . 57
5.1.1 Computação na PeTM-Par . . . 57
5.1.2 Seqüência de interações na PeTM-Par . . . 58
5.1.3 Operações pertinentes para a PeTM-Par . . . 58
5.2 Modelagens de Sistemas na PeTM-Par . . . 60
5.2.1 Sistema de Computação Par-a-Par . . . 60
5.2.2 Conta bancária . . . 62
5.3 Modelos de Comportamento da PeTM-Par . . . 62
5.4 Modelo baseado em linguagem da computação da PeTM-Par . . . 63
5.4.1 Linguagem na TM . . . 63
5.4.2 Equivalência e Expressividade na PeTM-Par . . . 63
5.5 Modelo da PeTM-Par baseado em funções . . . 63
5.5.1 Mapeamento Ξ . . . 63
5.6 Equivalência da PeTM-Par e PeTM baseadas em função . . . 64
5.7 Modelo da PeTM-Par baseado em Ambiente . . . 67
5.8 Considerações para a PeTM-Par . . . 67

6 CONCLUSÃO . . . 69
6.1 Resultados Obtidos e Contribuições . . . 70
6.2 Trabalhos futuros . . . 70

REFERÊNCIAS 71

LISTA DE ABREVIATURAS E SIGLAS

TCI	Teoria da Computação Interativa
TCC	Teoria da Computação Clássica
TM	Turing Machine (Máquina de Turing)
PTM	Parallel Turing Machine (Máquina de Turing Paralela)
MTFL	Máquina de Turing com fita limitada
FSA	Finite State Automata (Autômato com Estado Finito)
IM	Interactive Machine (Máquina Interativa)
SIM	Sequential Interaction Machine (Máquina Interativa Sequencial)
MIM	Multi-Stream Interaction Machine (Máquina Interativa Multi-Sequencial)
PeTM	Persistent Turing Machine (Máquina de Turing Persistente)
PeTM-Par	Máquina de Turing Persistente Paralela
LTS	Labelled Transition System (Sistema de Transição Etiquetada)
UC	Unidade de Controle

LISTA DE SÍMBOLOS

ι	Seqüência de entrada e saída
Σ	Alfabeto da máquina
Π	Função programa da TM
ε	Palavra vazia
$\mathcal{L}(M)$	Linguagem da máquina M
$\langle w_1, w_2, \ldots, w_n \rangle$	Lista de fitas de trabalho
O	Ambiente
$\oplus$	Ou-exclusivo
Θ	Seqüência infinita
ϕ	Mapeamento da máquina PeTM
Ξ	Mapeamento da PeTM-Par

LISTA DE FIGURAS

Figura 2.1: Representação da computação algorítmica 24
Figura 2.2: Máquina de Turing 25
Figura 2.3: fita $\times$ UC $\times$ cabeça 32
Figura 2.4: fita $\times$ UC $\times$ n cabeças 32
Figura 2.5: fita $\times$ n UCs $\times$ cabeça 32
Figura 2.6: fita $\times$ n UCs $\times$ n cabeças 32
Figura 2.7: n fitas $\times$ UC $\times$ cabeça 32
Figura 2.8: n fitas $\times$ UC $\times$ n cabeças 32
Figura 2.9: n fitas $\times$ n UCs $\times$ cabeça 32
Figura 2.10: n fitas $\times$ n UCs $\times$ n cabeças 32

Figura 3.1: Máquina Interativa Seqüencial 39

Figura 4.1: Máquina de Turing Persistente 45

Figura 5.1: Máquina de Turing Persistente Paralela 58
Figura 5.2: Seqüência de Interação da PeTM-Par 59
Figura 5.3: Construção da PeTM a partir da PeTM-Par 66

LISTA DE TABELAS

Tabela 2.1: Classificação 30
Tabela 3.1: Paralelo entre Algoritmos e Interação 34
Tabela 5.1: Relação entre a evolução de estados da PeTM-Par e da PeTM . 67

RESUMO

Nos últimos 70 anos têm sido apresentadas várias propostas para caracterização da noção intuitiva de computabilidade. O modelo de Computação mais conhecido para expressar a noção intuitiva de algoritmo é a Máquina de Turing. Esse trabalho apresenta máquinas abstratas que representam diferentes formas de comportamento computacional, sendo possível abordar a diversidade entre a Teoria da Computação Clássica (Máquina de Turing) e a Teoria da Computação Interativa (Máquina de Turing Persistente). Com a evolução dos sistemas de computação, surgiu a necessidade de estender a definição de Máquina de Turing para tratar uma diversidade de novas situações, esses problemas conduziram a uma mudança de paradigma. Neste contexto foi desenvolvido a Máquina de Turing Persistente, que é capaz de fundamentar a Teoria da Computação Interativa. Máquinas de Turing Persistentes (PeTM) são modelos que expressam comportamento interativo, esse modelo é uma extensão da Máquina de Turing. O presente trabalho tem como objetivo explorar paralelismo na Máquina de Turing Persistente, através da formalização de uma extensão paralela da PeTM e o estudo dos efeitos sobre essa extensão, variando o número de fitas de trabalho. Contribuições desse trabalho incluem a definição de uma máquina de Turing Persistente Paralela para modelar computação interativa e uma exposição de conceitos fundamentais e necessários para o entendimento desse novo paradigma. Os métodos e conceitos apresentados para formalização da computação na Máquina de Turing Persistente Paralela desenvolvidos nessa dissertação, podem servir como base para uma melhor compreensão da Teoria da Computação Interativa e da forma como o paralelismo pode ser especificado em modelos teóricos.

Palavras-chave: Máquina de Turing Persistente Paralela, Máquina de Turing, Teoria da Computação, Teoria da Computação Interativa, Máquina de Turing Persistente.

Analysis of the Persistent Turing Machine with Multiple Work Tapes

ABSTRACT

In the last 70 years, several proposals have been presented to characterise the intuitive notion of computability. The computational model best known for expressing the intuitive notion of algorithm is the Turing Machine. This work presents abstract machines which represent different ways of computational behaviour, permitting to explore the diversity between the Theories of Classic Computation (Turing Machine) and Interactive Computation (Persistent Turing Machine). The evolution of computing systems motivated the extension of the Turing Machine definition in order to deal with a set of new situations. In this context the Persistent Turing Machine has been developed, being able to fundament the Theory of Interactive Computation. Persistent Turing Machines (PeTM) are extensions of Turing Machine which express interactive behaviour. This work has the purpose of exploring parallelism on the Persistent Turing Machine, by formalising a parallel extension of the PeTM and studying the effects of varying the number of tapes on this extension. The contributions of this work include the definition of a Parallel Persistent Turing Machine for modelling interactive computing and an exposition of fundamental and necessary concepts for the understanding of this new paradigm. The methods and concepts presented in this thesis for the formalisation of computation on the Parallel Persistent Turing Machine may serve as a basis for a better comprehension of the Theory of Interactive Computation and of how parallelism may be specified in theoretical models.

Keywords: Parallel Persistent Turing Machine, Turing Machine, Theory of Computation, Theory of Interactive Computing, Persistent Turing Machine.

1 INTRODUÇÃO

Nos últimos 70 anos, têm sido apresentadas diferentes propostas para caracterização da noção de computabilidade. O modelo de Computação mais conhecido para expressar a noção intuitiva de algoritmo é a Máquina de Turing. Esse trabalho apresenta máquinas abstratas, Máquina de Turing e Máquina de Turing Persistente, que representam diferentes formas de comportamento computacional, sendo possível abordar a diversidade entre a Teoria da Computação Clássica e a Teoria da Computação Interativa, respectivamente. Nesta dissertação, é possível verificar como o paralelismo/concorrência pode ser especificado nesses modelos teóricos, podendo servir como base para a compreensão da Teoria da Computação Interativa.

1.1 A Teoria da Computação

A Ciência da Computação vem se expandindo em várias direções, tornando-se necessário desenvolver novos formalismos, que são bastante importantes para ter um sistema bem definido. Com essa expansão, é possível distinguir duas áreas na Ciência da Computação: uma aborda comportamento algorítmico, e a outra aborda comportamento interativo. A Teoria da Computação Clássica enquadra-se no primeiro caso, sendo vista como base da Ciência da Computação (LEWIS; PAPADIMITRIOU, 1998). Com a evolução da Ciência da Computação, a Teoria da Computação teve seu desenvolvimento motivado pelas necessidades originadas pelos avanços tecnológicos, requerendo, assim, uma maior elaboração de novos formalismos e/ou um maior detalhamento dos existentes.

Sabe-se que a Teoria da Computação teve origem na lógica formal, muito antes da existência de computadores. Alguns matemáticos dedicavam-se ao estudo e à formalização da noção de algoritmos, mas a revolução da Computação começou efetivamente a realizar-se no ano de 1935, quando Alan Turing definiu um dispositivo computacional com o propósito de modelar a noção intuitiva de algoritmos.

Essa noção de algoritmos foi discutida na publicação (TURING, 1936), que resultou na definição da Máquina de Turing (TM). Turing propôs esse modelo com o objetivo de explorar os limites da capacidade de expressar soluções de problemas (DIVERIO; MENEZES, 2000). Durante os primeiros 60 anos esse formalismo se estabeleceu como princípio básico, determinando uma fundamentação para Ciência da Computação. Nesse contexto, surgiu a Tese de Church-Turing, onde o que é computável é tudo que pode ser representado em uma Máquina de Turing. Dentro desse paradigma, utiliza-se o modelo de Máquina de Turing para

estudar: computabilidade, reconhecimento de linguagens e solucionabilidade de problemas.

Outro aspecto importante e atual na Ciência da Computação é a Teoria da Concorrência e/ou Processamento Paralelo. O Processamento Paralelo surgiu da necessidade de maior poder de processamento, logo no início da história do computador. Uma das primeiras máquinas paralelas, lançada em 1955, foi o IBM 704 cujo responsável foi Gene Amdahl. Amdahl foi, pode-se dizer, um dos primeiros teóricos na área de Processamento Paralelo, tendo formulado a conhecida Lei de Amdahl (ANDREWS, 1991). Poucos anos depois, em 1962, Carl Adam Petri propõe um modelo de comunicação em sistemas concorrentes que vem a ser conhecido como Redes de Petri (REISIG, 1985), um dos principais modelos para formalização de concorrência. Mais recentemente, em 1980, Robin Milner (1989) define o *Calculus of Communicating Systems*, ou CCS, que é outro formalismo que se aproxima da noção de processos concorrentes existente nos sistemas operacionais da atualidade.

Nesse contexto, o trabalho desenvolve a idéia de paralelismo/concorrência, usando o formalismo de máquinas abstratas já existentes. Na literatura, essa modificação na máquina tradicional é conhecida como Máquina de Turing Paralela. Diversos pesquisadores definiram modelos paralelos, entre os quais, destaca-se o trabalho de Wiedermann (1984) e de Worsch (1997). As modificações ocorreram nas estruturas do modelos, tendo sido acrescentados um ou mais componentes no modelo original (DIVERIO; MENEZES, 2000). Contudo, foi provado que esses formalismos têm o mesmo poder computacional que a Máquina de Turing original. Busca-se ainda, dentro dessa ótica, uma formalização do conceito de paralelismo/concorrência em modelos teóricos, visto que é de grande importância nas pesquisas desenvolvidas atualmente.

Assim, este trabalho foi elaborado preocupando-se com as definições e conceitos estabelecidos desde o início da Computação, chamando a atenção para a importância da Máquina de Turing. Essas noções são necessárias para abordar computabilidade na Teoria da Computação. Porem, Church e Kleene (KLEENE, 1967) definiram Funções Recursivas também com o objetivo de formalizar a noção intuitiva de função computável. Conforme os resultados estabelecidos por Turing, Church e Kleene, pode-se dizer que não existem algoritmos para a solução de determinadas funções, sendo então chamadas de funções não computáveis. A partir desde momento, descobriu-se o limite entre funções computáveis e não computáveis, verificando assim, os limites alcançados por um computador comum. Foi provado que os dois formalismos são equivalentes, isto é, geram o mesmo resultado em termos de capacidade de solucionar problemas, o que veio a reforçar a Hipótese de Church que será detalhada na Seção 2.2.

Uma vez apontadas a importância e a necessidade do estudo de algoritmos e da Máquina de Turing, pode-se apresentar e discutir a Teoria da Computação Interativa, que trata uma variedade de novos problemas (WEGNER; GOLDIN, 1999a). Esses problemas vão além dos limites teóricos da computação (SIPSER, 1997). Esse limite tem sido investigado e questionado quanto à forma de definir programa e o comportamento da máquina.

1.2 A Teoria da Computação Interativa

Com a evolução dos sistemas de computação, surgiu a necessidade de estender a definição de Máquina de Turing para tratar uma diversidade de novos problemas, como por exemplo, sistemas interativos (WEGNER; GOLDIN, 1999a). Essa evolução da computação é apresentada por uma mudança de paradigma, de algorítmico para interativo.

Os Sistemas Reativos são exemplos de sistemas interativos, que requerem a especificação do relacionamento de entradas e saídas no decorrer do tempo. Tais descrições envolvem seqüências de eventos, ações, condições e fluxo de informações, freqüentemente com restrições de tempo explícitas que se combinam para formar o comportamento do sistema. Máquinas de Estados e Redes de Petri são freqüentemente utilizadas para especificar o comportamento desses sistemas (FERBER, 1999).

A compreensão de que Máquinas de Turing não podem expressar computação interativa conduz a essa mudança de paradigma. Isso requer uma reformulação nos conceitos e princípios fundamentais sobre a natureza da Ciência da Computação. Por exemplo, a noção intuitiva de computação deve ser ampliada para além da Tese de Church (WEGNER, 2001). Essa extensão, sob o ponto de vista da Ciência da Computação, requer também uma extensão dos modelos de Computação.

A partir dessa necessidade, introduziu-se o conceito de Máquina Interativa (WEGNER; GOLDIN, 1999b) e mais adiante, desenvolveu-se o modelo de Máquina de Turing Persistente (GOLDIN, 2000). Essa extensão do paradigma de algoritmos para o paradigma de interação fornece uma maior/melhor fundamentação para modelos de interação seqüencial, conforme será demonstrado no trabalho.

Pode-se, brevemente, descrever uma Máquina de Turing como sendo um dispositivo computacional que transforma símbolos de entrada pré-definidos em símbolos de saída, mas que não permite comunicação com o ambiente durante o processo da computação ("*shutting out the world*" (WEGNER, 1998)). Tal formalismo descreve a Teoria da Computação Clássica da Ciência da Computação.

Segundo a definição de máquina apresentada por Wegner e Goldin (1998; 2000), pode-se abordar a diversidade entre o modelo de Máquina de Turing da Teoria da Computação Clássica e o modelo de Máquina de Turing Persistente (PeTM) da Teoria da Computação Interativa, podendo esta última ser vista como um dispositivo de computação interativa que admite ações de entrada e saída durante o processo da Computação (WEGNER; GOLDIN, 2001).

Uma Máquina de Turing Persistente é uma Máquina de Turing com múltiplas fitas e com uma fita de trabalho persistente preservada entre as interações, na qual a seqüência de entrada gera dinamicamente a seqüência de saída de símbolos (palavras). Esse modelo é uma extensão das Máquinas de Turing que expressam comportamento interativo (GOLDIN; WEGNER, 2000).

1.3 Motivação do trabalho

Este trabalho busca uma abordagem teórica dentro da Ciência da Computação, mostrando aspectos para a formalização do conceito de computabilidade e

os modelos de maquinas abstratas, nos quais a Computação é baseada.

O presente trabalho está dividido em duas partes: a primeira parte abrange a Teoria da Computação Clássica, e a outra, a Teoria da Computação Interativa, sendo que cada parte será exemplificada por um dispositivo teórico. É possível ainda, verificar propriedades e características comuns às duas teorias citadas, que estão baseadas no paradigma de algoritmos e no paradigma de interação, respectivamente.

A primeira parte tem como objetivo contextualizar conceitos e definições dentro da Teoria da Computação, para então explorar a Teoria da Computação Interativa. A outra parte, baseada em (WEGNER, 2001), introduzirá alguns conceitos em que o trabalho está fundamentado, necessários para abordar o paradigma de interação. Nesse contexto, Wegner definiu a PeTM, demonstrando que a máquina tem comportamento interativo seqüencial.

Uma motivação para o trabalho deve-se à possibilidade de um formalismo baseado na Máquina de Turing Persistente ser capaz de modelar sistemas que não podem ser expressos por funções computáveis (GOLDIN, 2000; GOLDIN; WEGNER, 2000, 1998). A idéia principal do trabalho é definir uma máquina de Turing Persistente *Paralela*, isto é, acrescentar fitas de trabalho na Máquina de Turing Persistente já existente. Com isso, será possível analisá-la e discutir como ficam relacionados os formalismos e o comportamento interativo dentro do contexto da Teoria da Computação Interativa.

Chega-se ao tema desta dissertação, que é o paralelismo em modelos teóricos no contexto da Teoria da Computação Interativa. De acordo com a idéia principal, que partiu da equivalência de Máquinas de Turing e Máquinas de Turing Paralelas, deseja-se mostrar que o acréscimo de fitas persistentes na Máquina de Turing Persistente não aumenta a expressividade do modelo original.

1.4 Objetivos e Contribuições

Os objetivos deste trabalho são:

- A definição de um modelo de Máquina de Turing Persistente Paralela (PeTM-Par) e o estudo de seus efeitos, sobre essa definição, da variação do número de fitas de trabalho;

- Analisar o modelo de Máquina de Turing Persistente e a sua expressividade, relacionando-o com os modelos da Teoria da Computação Clássica;

- Identificar características das teorias Clássica e Interativa para descrever limites da computabilidade (computação).

Nessa dissertação, discute-se Computabilidade em dois contextos diferentes, Teoria da Computação Clássica e Teoria da Computação Interativa, sendo que as máquinas de Turing são os formalismos adotados para demonstrar as características e relações entre esses contextos. Pode-se dizer que este trabalho relaciona a Teoria da Computação Clássica e a Teoria da Computação Interativa, mas sem confrontá-las, pois estão inseridas em contextos diferentes, baseadas em algoritmo e em interação, respectivamente.

Assim, as contribuições desta dissertação incluem a extensão de um modelo de máquina paralela, que tem um comportamento interativo seqüencial, a demonstração que a PetM é equivalente a Petm-Par, e uma exposição de conceitos fundamentais para melhor entender a Computação Interativa empregando paralelismo.

Os modelos de computação — Máquina de Turing Persistente e Máquina de Turing Persistente Paralela — que serão desenvolvidos neste trabalho, poderão servir como base para uma maior compreensão da Computação Interativa e de como o paralelismo pode ser especificado em modelos teóricos.

Na seção seguinte, é descrita a forma como o texto está estruturado.

1.5 Estrutura do Texto

O texto encontra-se dividido em seis Capítulos.

No Capítulo 2, será mostrado o desenvolvimento dos conceitos teóricos que formaram a base para o surgimento da Teoria da Computação. Esse estudo é essencial para o prosseguimento da dissertação. Inicialmente, serão abordados: algoritmos, modelos de máquinas e a Tese de Church. Em seguida, será apresentada a construção da Máquina de Turing, que foi um dispositivo computacional relevante à Computação. Na seqüência, é descrita a Máquina de Turing Paralela.

No Capítulo 3, será feita uma introdução à mudança de paradigma de algoritmos para interação. Uma vez formada a estrutura conceitual, será apresentada um modelo de máquina interativa. Seguindo o estudo, será detalhado, no Capítulo 4, o modelo de máquina de Turing Persistente, no qual o modelo definido na dissertação está baseado. Apresenta ainda, os modelos de comportamento da Máquina de Turing Persistente.

O Capítulo 5 apresenta a definição formal da Máquina de Turing Persistente Paralela, reunindo todas as definições e conceitos exibidos ao longo do texto.

Por fim, o Capítulo 6 inclui as considerações finais, resultados obtidos, bem como uma análise sobre eles e trabalhos futuros, onde são apresentados alguns tópicos que poderão dar continuidade ao trabalho.

2 NOÇÕES DE COMPUTABILIDADE

Neste capítulo busca-se uma abordagem de abrangência teórica dentro da Ciência da Computação, mostrando os aspectos de formalização dos principais conceitos e modelos nos quais a Teoria da Computação é baseada. Ao longo do capítulo é apresentada uma definição de Máquina de Turing, seguida pela enumeração de suas diversas modificações, até formar a base necessária para descrever as Máquinas de Turing Paralelas. Os conceitos apresentados neste capítulo estão baseados nos livros-texto de Minsky (1967), Bird (1976), Lewis (1998), Diverio (2000), Menezes (2001), Toscani (2001) e Hopcroft (2001).

2.1 Teoria da Computação Clássica

Nesta seção é introduzido o surgimento da Teoria da Computação que, conforme será visto, partiu dos pensadores e matemáticos no início do século XX. Ao longo do texto, quando refere-se a Teoria da Computação, utilizar-se-á a denominação Teoria da Computação Clássica (TCC).

Sabe-se que a TCC teve origem na lógica formal, muito antes da existência de computadores. Alguns matemáticos dedicavam-se ao estudo e formalização da noção de algoritmos. A revolução da Computação começou efetivamente a realizar-se no ano de 1936 quando Alan Turing tomou conhecimento do *Entscheidungsproblem*, de Hilbert.

Um parênteses para falar de David Hilbert. Esse cientista, em 1900, propôs uma lista de 23 problemas[1] cuja solução estimularia vários outros pesquisadores. Vários desses problemas foram desde então resolvidos, enquanto que alguns resistem até hoje e permanecem ainda em aberto. O problema de número dez é especialmente interessante para o Computação: é o famoso problema da decidibilidade, o *Entscheidungsproblem*, já citado acima.

Historicamente, pode-se considerar o trabalho de Hilbert como marco da Teoria da Computação Clássica. A partir dele, Turing teve a idéia de um dispositivo teórico que se tornou um conceito essencial dentro da Teoria da Computação.

Turing mostrou que para qualquer algoritmo existe uma Máquina de Turing para representá-lo. Essa teoria foi estabelecida pela primeira vez em seu famoso artigo "*On Computable Numbers, with an application on the Entscheidungsproblem*" (TURING, 1936).

[1] Alguns problemas podem ser encontrados na obra de Schöning e Pruim (1998); uma tabela com uma descrição de cada problema pode ser encontrada em http://aleph0.clarku.edu/∼djoyce/hilbert/toc.html (JOYCE, 2001)

A Máquina de Turing era a resposta de Alan Turing ao problema de Hilbert. Turing estabeleceu um modelo formal de algoritmo; pouco tempo depois, Church manifestaria que qualquer procedimento efetivo poderia ser realizado por uma Máquina de Turing, o que vem a ser a Tese de Church (Seção 2.2). Assim, a noção de computabilidade começa a ser descrita sucintamente.

Também Church estava interessado no problema de Hilbert. O resultado a que Turing tinha chegado em 1936, sobre o problema da decisão de Hilbert, foi também alcançado por Church, alguns meses antes, empregando o conceito de lambda-definibilidade [2]. Kleene, em 1936, mostrou que todas as funções recursivas (KLEENE, 1967) podem ser representadas por lambda-definibilidade. Após tomar conhecimento da proposta de Church, Turing estabeleceu, em um apêndice de seu artigo, que a abordagem da lambda-definibilidade e a sua própria eram equivalentes (ZALTA, 2001). Nesse tempo Church formularia sua tese que estabelecia que a recursividade é a formalização do efetivamente computável.

A partir dos resultados encontrados por Turing e Church, pode-se dizer que existem funções para as quais não existe uma seqüência de passos que determinem o seu valor, com base nos seus argumentos. Interpretando de outra maneira, não existem algoritmos para a solução de determinadas funções. São as chamadas funções não computáveis, esclarecidas na Seção 2.2.

Embora estes formalismos são equivalentes, com enormes diferenças de notação e enfoque, cada um tem uma utilidade na Teoria da Computação Clássica. Por exemplo, o cálculo lambda é a base para o Lisp, a primeira linguagem de programação voltada para a manipulação simbólica. A equivalência dos formalismos relacionados por Turing e Church acima deu origem à chamada Hipótese de Church, detalhada a seguir.

2.2 Tese de Church

Classicamente, computabilidade é tida como tudo que pode ser resolvido como um programa em uma Máquina de Turing. Isso descreve exatamente a Hipótese de Church, que não pode ser demonstrada devido a noção intuitiva de algoritmos (DIVERIO; MENEZES, 2000). A Hipótese de Church é tida como verdadeira por todos os estudiosos relacionados à Teoria da Computação. É uma hipótese, e não um teorema, porque não foi demonstrado, apenas diz que um algoritmo (conceito informal) expresso na forma de uma Máquina de Turing é capaz de processar qualquer função.

As evidências externas e internas (Seção 2.4.3) apenas reforçam a Hipótese de Church, ou seja, todos os modelos apresentados, assim como variações da Máquina de Turing, mostraram ter, no máximo, a mesma capacidade computacional do modelo desenvolvido por Turing. Portanto, pode-se considerá-la como um princípio básico para Ciência da Computação.

Pode-se dizer ainda, que existem funções para as quais não existe uma TM

[2] O conceito de uma função lambda-definibilidade é devido a Church e Kleene e ao conceito de função recursiva de Gödel e Herbrand. A classe das funções lambda-definibilidade e a classe das funções recursivas são idênticas. Isso foi determinado por Church e Kleene para o caso de funções de inteiros positivos. Assim, na proposta de Church, os termos "função recursiva de inteiros positivos" podem ser substituídos pelos termos "função de inteiros positivos computável pela máquina de Turing" (ZALTA, 2001)

que as solucione. Expressando-se de outra forma, não existem algoritmos para a solução de determinadas funções. Essas funções são conhecidas como funções não computáveis. Logo, descobrir os limites entre funções computáveis e não computáveis é equivalente a verificar os limites do computador em geral.

Esses estudos ficaram conhecidos na Teoria da Computação Clássica como computabilidade. Computabilidade é a descrição e caracterização de problemas que podem ser resolvidos algoritmicamente.

2.3 Algoritmos e Procedimentos efetivos

Para estudar computação de um ponto de vista teórico, com a finalidade de caracterizar o que é ou não é computável, é necessário introduzir um modelo teórico que represente o que se compreenda como computação. Entretanto, cada pesquisador do assunto trabalha com um formalismo favorito, e por isso há diversas definições distintas. Até o momento, o termo *algoritmo* foi intuitivamente usado como solução de um problema, ou seja, uma forma de descrever se determinada propriedade é verificada ou não para uma dada classe de entrada.

Esta seção tem a intenção de desenvolver a noção intuitiva do que quer dizer computável. Para isso, será introduzido, informalmente, o conceito de algoritmo. Relativamente à noção intuitiva de algoritmo, pode-se afirmar:

- sua descrição deve ser finita e não-ambígua;
- deve consistir de um número finito de passos discretos, executáveis mecanicamente em um tempo finito.

Segundo Toscani e Veloso (2001), "algoritmo é um procedimento, consistindo de um conjunto de regras não ambíguas, as quais especificam, para cada entrada, uma seqüência finita de operações, terminando com uma saída correspondente".

Um algoritmo pode ser visto como um sistema fechado, que recebe entradas e as transforma em saídas. Levando em consideração o ponto de vista do observador (ou ambiente), a máquina pode usualmente ser considerada como uma caixa fechada com entradas e saídas (MINSKY, 1967), cuja representação gráfica pode ser verificada na Figura 2.1. Uma Máquina de Turing pode também ser vista como um sistema de transição de estados, detalhada na Seção 2.4.1. A TM tornou-se a mais conhecida definição de algoritmo, conduzindo assim à definição de "computável": um problema é computável se é solucionável em uma Máquina de Turing. Esta definição ficou conhecida como Função Turing-Computável, que também pode ser encontrada na literatura somente como Função Computável. Uma definição formal da Função Turing-Computável pode ser encontrada em Diverio e Menezes (2000).

2.4 Modelos de computação

Atualmente existem vários modelos para representar a noção de algoritmos. Dentre elas, pode-se citar as Máquinas com pilhas, a Máquina de Post, a Máquina Norma, a Máquina de Turing, as quais têm o mesmo poder computacional (DIVERIO; MENEZES, 2000). Nesta seção é feito um apanhado geral da Máquina de Turing e suas diversas alterações.

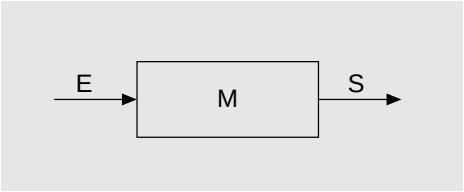

Figura 2.1: Representação da computação algorítmica

2.4.1 Máquina de Turing

Conforme já exposto, a Máquina de Turing é aceita como a formalização da noção de algoritmo (DIVERIO; MENEZES, 2000). Segundo a Tese de Church (LEWIS; PAPADIMITRIOU, 1998), esse modelo é considerado uma máquina universal (Seção 2.4.3), ou seja, é o limite máximo do que é computável. Esse formalismo representa um mecanismo simples que formaliza idéia de uma pessoa que realiza cálculos. Apesar de sua simplicidade, esse dispositivo possui, no mínimo, o mesmo poder computacional de qualquer computador de propósito geral (DIVERIO; MENEZES, 2000).

Nesse contexto, conforme já apresentado anteriormente, surgiu o paradigma de algoritmos, onde o que é computável é tudo que pode ser representado em uma Máquina de Turing. Dentro desse paradigma, emprega-se o modelo de Máquina de Turing para:

- *Computabilidade*, desenvolver programas que executam funções. Nesse contexto, a fita da Máquina de Turing, é usada como variável de entrada, memória de trabalho e variável de saída. O resultado da função (da computação) quando termina, é apresentado na fita. Assim, tem-se a classe das funções computáveis, que determinam o limite de computabilidade (MINSKY, 1967);

- *Reconhecimento de linguagens*, outro estudo desenvolvido nesse paradigma, onde programas são desenvolvidos nas máquinas de Turing de forma a reconhecer uma determinada linguagem. Na computação desses programas o conteúdo final da fita não é importante, mas sim o estado onde a máquina pára, pois determina o reconhecimento ou não da entrada e, ainda, o tipo de linguagem, recursiva e/ou enumerável recursivamente (MENEZES, 2001);

- *Solucionabilidade de problemas*; esse estudo determina se algum problema é ou não solucionável e/ou computável. Utiliza-se o princípio da redução de Turing, onde se reduz um problema a outro (DIVERIO; MENEZES, 2000).

A Máquina de Turing, por sua vez, é um modelo cuja computação é essencialmente seqüencial. Máquinas de Turing são dispositivos de computação que transformam palavras de entrada em palavras de saída por seqüências de transições de estado (HOPCROFT; MOTWANI; ULLMAN, 2001).

A interação de uma máquina de Turing consiste em receber uma entrada, depois processá-la, e devolver uma saída (resultado).

Essa máquina é constituída de três partes (conforme Figura 2.2):

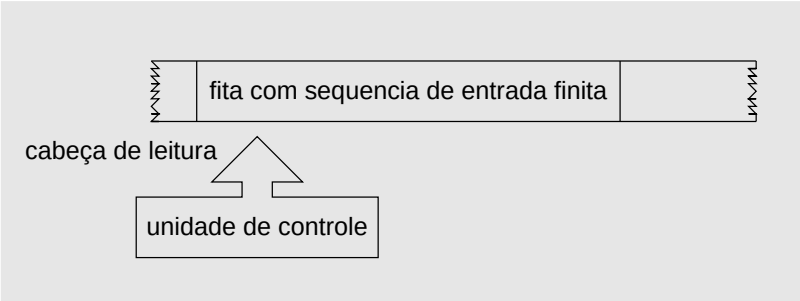

Figura 2.2: Máquina de Turing

- Fita — usada como dispositivo de entrada, de saída e de memória de trabalho;
- unidade de controle — reflete o estado corrente da máquina. Possui uma cabeça de leitura e gravação, que acessa uma célula da fita de cada vez e movimenta-se pra esquerda ou para direita;
- programa — função que define o novo estado da máquina e comanda as leituras, as gravações e o sentido do movimento da cabeça.

A seguir é exposta uma formalização da TM. Maiores detalhes sobre o funcionamento da Máquina de Turing são mostrados por Diverio e Menezes (2000).

2.4.2 Formalização do modelo

O modelo computacional descrito neste capítulo utiliza a definição formal apresentada em (DIVERIO; MENEZES, 2000).

Definição 2.1 (Máquina de Turing) *Uma Máquina de Turing M é uma 8-upla $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, *)$ onde:*

Σ *alfabeto de símbolos de entrada;*
Q *conjunto de estados possíveis da máquina, o qual é finito;*
Π *programa ou função de transição:*
$\Pi = Q \times (\Sigma \cup V \cup \{\beta, *\}) \rightarrow Q \times (\Sigma \cup V \cup \{\beta, *\}) \times \{E, D\}$;
q_0 *estado inicial da máquina, tal que $q_0 \in Q$;*
F *conjunto de estados finais, tal que $F \subset Q$*
V *alfabeto auxiliar;*
β *símbolo especial branco;*
$*$ *símbolo especial marcador de início de fita*

A configuração inicial da máquina é o símbolo de início de fita, que ocorre uma vez e sempre na célula mais à esquerda da fita, auxiliando na identificação de que a cabeça da fita se encontra na célula mais à esquerda. Depois vem a entrada, ocupando as células mais a esquerda, após o marcador, e as demais células são ocupadas pelo símbolo branco.

A unidade de controle possui um número finito e predefinido de estados. A cabeça de fita lê o símbolo de uma célula de cada vez e grava um novo símbolo. Após a leitura/gravação, a cabeça move uma célula para a direita ou esquerda.

O símbolo gravado e o sentido do movimento são definidos pelo programa. O programa é uma função que, dependendo do estado corrente da máquina e do símbolo lido, determina o símbolo a ser gravado, o sentido do movimento da cabeça e o novo estado.

Na função programa da Máquina de Turing, os símbolos EeD representam o sentido do movimento da cabeça, para esquerda e direita, respectivamente. A função programa $\Pi(p, a_u) = (q, a_v, m)$:

Considera	Determina
estado corrente (p)	novo estado (q)
símbolo lido (a_u)	símbolo gravado (a_v)
	sentido do movimento (m)

O programa pode ser representado como um grafo finito. Adicionalmente, é possível também representar a função programa na forma de tabela:

Π	a_u	a_v	...
p	(q, a_v, m)		
q			
...			

Essa definição é aqui apresentada de forma simplificada, porém suficiente para o entendimento do restante do trabalho. Definições mais elaboradas podem ser encontradas na literatura referenciada.

2.4.3 Máquinas de Turing como Máquina Universal

Uma máquina universal é a denominação utilizada para caracterizar que uma determinada máquina é capaz de representar qualquer algoritmo sob forma de um programa (MANNA, 1974). Para descrever que uma máquina é universal, deve-se observar as evidências externa e interna, conforme apresentada por Diverio e Menezes (2000):

Evidência Externa Consiste na demonstração de que qualquer extensão das capacidades da máquina proposta computa, no máximo, a mesma classe de funções, ou seja, não aumenta o seu poder computacional;

Evidência Interna Consiste na análise de modelos que também querem definir a noção de algoritmo, assim como a prova de que são, no máximo, computacionalmente equivalentes.

Vários modelos formais de computação foram criados e investigados. Na seção 2.5 são abordados alguns modelos computacionais, todos equivalentes em poder computacional ao modelo de Máquina de Turing. Dizer que dois dispositivos computacionais são equivalentes significa que ambos resolvem a mesma classe de problemas.

2.5 Outras abordagens à computabilidade

O mais importante questionamento sobre um problema é a sua computabilidade, quer dizer, se ele pode ser resolvido mecanicamente por um algoritmo. Para identificar e definir questões como essa, surgiram vários formalismos.

Destas buscas, alguns formalismos surgiram ao longo dos anos:

Funções Recursivas Parciais (1936) Gödel-Kleene contribuíram para a Teoria da Computação desenvolvendo as funções recursivas parciais, que são funções construídas sobre funções básicas usando três tipos de construções: composição, recursão e minimização; Formalismo denotacional (KLEENE, 1967);

Cálculo Lambda (1936) Church criou o Cálculo Lambda com o intuito de captar os aspectos da maneira pelo qual funções podem ser combinadas para formar outras funções. A principal característica do Cálculo Lambda é representar as funções como entidades que podem ser, como um dado qualquer, utilizadas como argumentos retornadas como valores de outras funções. Esse formalismo é a base para as linguagens funcionais (componente fundamental da linguagem de programação LISP);

Algoritmo de Markov (1951) Markov foi o responsável pela especificação de uma estrutura de controle para sistemas de produção. O algoritmo de Markov (HEIN, 2002, p. 781–783) é uma seqüência ordenada finita de produções, as quais são aplicadas na ordem de prioridade das palavras de entrada, transformando palavras de entrada em palavras de saída.

Teoria dos Domínios Dana Scott apresentou a teoria dos domínios como uma base para a formalização da semântica de linguagens de programação (SCOTT, 1972).

As propostas apresentadas anteriormente caracterizam a noção de computabilidade por diferentes formalismos. Considerando que todas as evidências externas e internas possíveis foram sempre verificadas, reforçando a Hipótese de Church, os demais modelos de máquinas propostos, assim como as diversas modificações da máquina de Turing, possuem, no máximo, o mesmo poder computacional.

2.6 Máquinas Equivalentes à Máquina de Turing

Para falar em máquinas equivalentes à TM, antes é necessário introduzir o conceito de simulação e equivalência de máquinas. Informalmente, a simulação de máquinas depende da existência de programas de uma e de outra máquina onde as correspondentes funções computadas coincidem. Duas máquinas são equivalentes se uma pode ser simulada pela outra, e vice-ersa. Uma definição formal é dada por Diverio e Menezes (2000).

A partir desses conceitos, algumas máquinas equivalentes à Máquina de Turing são descritas a seguir:

Máquina Norma Proposta por Bird (1976), é definida de forma a lembrar a arquitetura básica de computadores atuais. Esse modelo distingue noções de

programa e máquina. A máquina Norma possui como memória um conjunto infinito de registradores naturais e três instruções sobre cada registrador (adição do valor um; subtração do valor um; teste se o valor armazenado é zero). A definição formal da máquina pode ser encontrada em Diverio (2000);

Sistema de Post (1936) Esse modelo tem como principal característica o utilização de uma estrutura de dados do tipo fila para entrada, saída e memória de trabalho. Estruturalmente, a característica de uma fila é que o primeiro valor gravado também é o primeiro valor a ser lido. A definição formal desse formalismo é apresentado por Minsky (1967);

Máquina de Acesso Randômico Conhecida por RAM (*Random Acess Machine*), vem do fato que os modelos mais antigos não eram de acesso randômico; estavam baseados em fitas seqüenciais ou eram funcionais em sua natureza (AHO; HOPCROFT; ULLMAN, 1974);

Autômato com Pilhas Esse modelo possui a memória de entrada separada das memórias de trabalho e de saída. É composto por fita, pilhas, unidade de controle e função de transição. Têm-se que a existência de duas pilhas proporcionam o modelo com maior poder computacional para esse formalismo. É equivalente a máquina de Turing. Detalhes do funcionamento estão em Diverio e Menezes (2000).

2.7 Modificações sobre a Máquina de Turing

Na Seção 2.4.2, definiu-se um modelo de Máquina de Turing. Nesta seção serão apresentadas possíveis variações desse modelo, as quais são alternativas válidas para formalizar a noção intuitiva de algoritmos. Minsky sustenta que essas alternativas não aumentam o poder computacional das máquinas, independentemente da estrutura específica escolhida para tal (MINSKY, 1967). Entretanto, dependendo do contexto, pode ser mais conveniente utilizar um ou outro modelo.

Cada variação possui um acréscimo em relação ao modelo original. Deve-se ressaltar ainda que as variações listadas a seguir não esgotam as possibilidades, mas são suficientes para comentar o poder computacional da Máquina de Turing. Tais variações são descritas a seguir:

- Máquina de Turing com fita infinita à esquerda e à direita - esse modelo permite que a fita seja infinita dos dois lados.
- Máquina de Turing com múltiplas fitas - esse modelo possui k fitas infinitas à esquerda e à direita e k correspondentes cabeças de fita;
- Máquina de Turing Multidimensional - nesse modelo, a fita tradicional é substituída por uma estrutura do tipo arranjo k-dimensional, infinita em todas as $2k$ direções; Uma definição formal desse modelo pode ser encontrada em (ACIÓLY; BEDREGAL; LYRA, 2000);
- Máquina de Turing com múltiplas cabeças - o modelo com esta modificação possui k cabeças de leitura e gravação sobre uma única fita. Cada cabeça

possui um movimento independente. Com isso, o processamento depende do estado corrente e do símbolo lido em cada uma das cabeças;

- Máquina de Turing com múltiplas fitas e múltiplas unidades de controle - em alguns trabalhos, esse modelo é formalmente definida como uma TM não-determinística com uma única fita (VAN EMDE BOAS, 1990);

- Máquina de Turing Não-Determinística - a generalização deste modelo é verificada no programa, pois para o mesmo estado e símbolo lido, diversos caminhos são possíveis. Genericamente, uma máquina, processa uma entrada e tem como resultado um conjunto de diversos caminhos, como se houvesse uma multiplicação da unidade de controle, uma para cada caminho, processando independentemente, sem compartilhar recursos com as demais.

Poder-se-ia falar de Máquina de Turing com fita limitada (MTFL) (MENEZES, 2001), que também é uma modificação sob a máquina original; porém, esse modelo restringe o tamanho da fita, saindo do escopo do trabalho aqui apresentado. Ressalta-se ainda que a MTFL é menos poderosa que a TM, de acordo com a Hierarquia de Chomsky, reconhece a classe de linguagens sensíveis ao contexto. Outra observação válida, a qual é ainda uma questão em aberto na Teoria da Computação Clássica, é se o não-determinismo aumenta ou não o poder computacional das MTFL.

2.8 Máquina de Turing Paralela

As diversas modificações propostas na estrutura original da Máquina de Turing, como por exemplo as citadas na Seção 2.7, foram provadas não aumentar o poder computacional desse formalismo. Algumas dessas alterações no modelo tradicional podem ser encontradas na literatura como Máquina de Turing Paralela (PTM).

Wiedermann coloca que existem diferentes modelos para computação paralela com a Máquina de Turing (WIEDERMANN, 1992). O mesmo autor estabelece ainda que uma PTM é informalmente vista como um conjunto de TM seqüenciais trabalhando sobre uma única fita comum (1984).

Em alguns trabalhos, a Máquina de Turing Paralela é formalmente definida como uma Máquina de Turing não-determinística com uma única fita (VAN EMDE BOAS, 1990).

Novas definições para Máquina de Turing Paralela têm sido estudadas; esse modelo teórico é bastante analisado para o estudo do paralelismo (WIEDERMANN, 1995). Em um outro trabalho (PY et al., 2001), desenvolveu-se uma variação da Máquina de Turing que também se enquadra na classe das PTMs. A idéia do trabalho foi desenvolver a formalização de uma PTM juntamente com um protótipo para ensino do paralelismo e conceitos relacionados.

Worsch (1997) desenvolve também a idéia de paralelismo na Máquina de Turing. Esse pesquisador definiu uma Máquina de Turing Paralela com múltiplas unidades de controle e cabeças e uma ou mais fitas, o que, segundo o autor, pode ser considerado como uma generalização de automâtos celulares (WORSCH, 1999).

Tabela 2.1: Classificação

Fita	UC	Cabeça	Descrição
S	S	S	Máquina de Turing original
S	S	M	TM com UC com várias cabeças
S	M	S	TM com memória compartilhada
S	M	M	TM única fita
M	S	S	TM várias fitas
M	S	M	TM várias cabeças
M	M	S	TM várias fitas e UCs
M	M	M	TM (*cluster*)

Dada a diversidade de variações da PTM, propõe-se a seguir um esquema de classificação análogo ao definido por Flynn para as arquiteturas paralelas (HWANG; BRIGGS, 1984).

2.8.1 Analogia da Classificação de Flynn nas Máquinas de Turing

Existe, atualmente, uma grande variedade de Máquinas de Turing paralelas e vários critérios para classificação desses tipos de máquina. Conforme a classificação de Flynn, pode-se fazer uma analogia em relação às modificações da Máquina de Turing. Máquinas Paralelas são vistas como um computador de alto desempenho, ou podem ser um conjunto de processadores capazes de trabalhar cooperativamente na solução de um problema computacional. Entender e apreciar as vantagens e benefícios dos modelos de máquinas paralelas pode ser apropriado para comparar o poder de computação e a eficiência desses dispositivos com os modelos seqüenciais.

Segundo Flynn, as arquiteturas de máquinas paralelas podem ser classificadas em quatro sub-categorias conforme a multiplicidade do fluxo de dados e de instruções (HWANG; BRIGGS, 1984). Segundo essa classificação, o ponto principal do processo de um computador é a execução de um conjunto de instruções sobre um conjunto de dados. A palavra fluxo é empregada para descrever uma seqüência de instruções, executados em um único processador. Portanto, um fluxo de instruções é uma seqüência de instruções executadas por uma máquina, enquanto que um fluxo de dados é uma seqüência de dados de entrada, de resultados parciais ou intermediários, utilizados pelo fluxo de instruções.

A partir da classificação de Flynn, foi possível propor uma classificação análoga para as diversas modificações da Máquina de Turing. Baseou-se nessa classificação por ser simples e reconhecida pela comunidade científica. Sendo assim, adotou-se esse método para definir as classes de Máquinas de Turing Paralelas. A Tabela 2.1 mostra resumidamente como está relacionada a classificação com as classes das PTM e possíveis modelos.

Esses modelos podem ser descritos como:

Classe SSS Nesta classe encontra-se a Máquina de Turing original; tem fluxo de instruções e de dados únicos (Figura 2.3);

Classe SSM Este modelo apresenta uma fita, uma unidade de controle e várias

cabeças. Baseado nesse modelo de máquina, foi desenvolvido um protótipo para execução de programas, como por exemplo a função x^2 (PY et al., 2001) — Figura 2.4;

Classe SMS Nesta classe (Figura 2.5) localizam-se as TMs com uma única fita e múltiplas unidades de controle, podendo ser vista como uma máquina paralela de memória compartilhada;

Classe SMM Semelhante à anterior, porém com múltiplas cabeças por UC; pode ser comparada a uma máquina com processadores vetoriais; Figura 2.6;

Classe MSS Apresenta várias fitas, com apenas uma UC e uma cabeça — Figura 2.7;

Classe MSM Máquinas desta classe apresentam paralelismo de dados, ou seja, todas as cabeças realizam sempre a mesma operação, sendo controladas por uma única UC. É ilustrada na Figura 2.8;

Classe MMS Esta classe de máquinas, representada na Figura 2.9, pode ser vista como um *cluster* (STERLING, 2002); os processadores dessa PTM são controlados por programas independentes, e cada um decide seu próximo movimento independentemente dos demais;

Classe MMM Semelhante à anterior, porém com várias cabeças por UC; ilustrada na Figura 2.10; a máquina definida por Worsch (1997) enquadra-se nesta classe.

Cabe observar que essa lista de modelos de máquinas não tem a intenção de ser exaustiva, apenas de ilustrar diversas possibilidades de modificações na TM. Mais informações sobre a taxonomia para modelos de máquinas paralelas e suas relações podem ser encontradas nos trabalhos de Van Emde Boas (1990) e Wiedermann (1984; 1995).

No desenvolvimento deste capítulo, muitos outros aspectos da Teoria não foram abordados, não por serem menos importantes, mas por não serem essenciais à compreensão do trabalho desenvolvido. Por exemplo, a equivalência de programas não foi explorada, somente a equivalência de modelos teóricos.

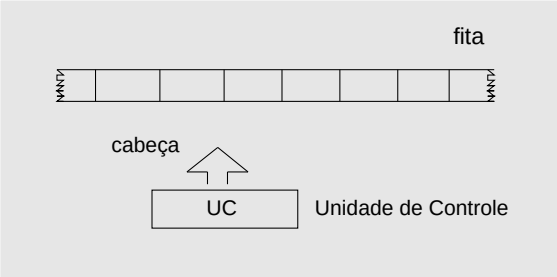

Figura 2.3: fita $\times$ UC $\times$ cabeça

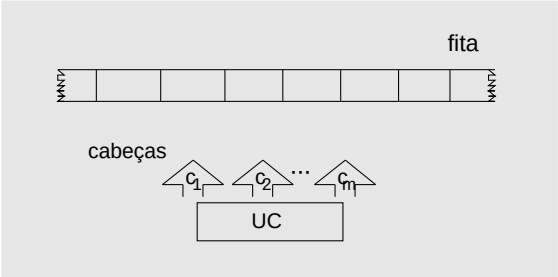

Figura 2.4: fita $\times$ UC $\times$ n cabeças

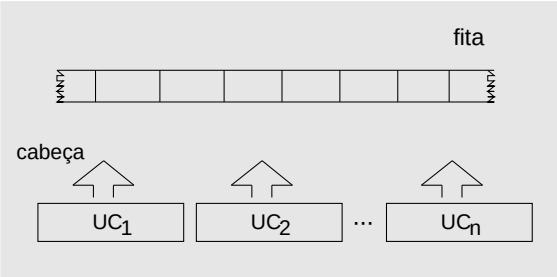

Figura 2.5: fita $\times$ n UCs $\times$ cabeça

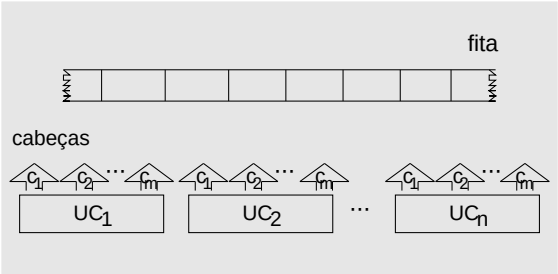

Figura 2.6: fita $\times$ n UCs $\times$ n cabeças

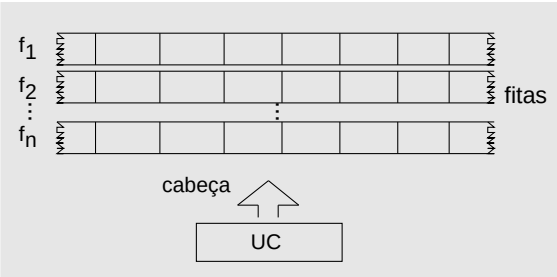

Figura 2.7: n fitas $\times$ UC $\times$ cabeça

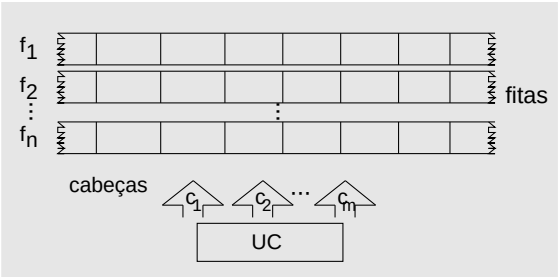

Figura 2.8: n fitas $\times$ UC $\times$ n cabeças

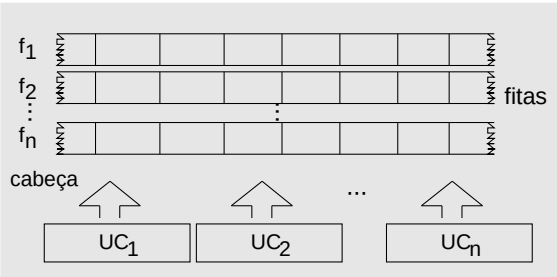

Figura 2.9: n fitas $\times$ n UCs $\times$ cabeça

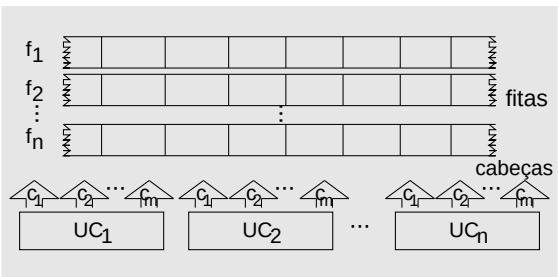

Figura 2.10: n fitas $\times$ n UCs $\times$ n cabeças

3 TEORIA DA COMPUTAÇÃO INTERATIVA

Este capítulo apresenta conceitos fundamentais da Computação Interativa, estabelecendo assim definições que sustentam que o comportamento interativo não pode ser expresso por Máquinas de Turing. Esses conceitos e definições serão usados ao longo do trabalho. O paradigma de Computação Interativa tem sido estudado e desenvolvido por Wegner em cooperação com Goldin (WEGNER, 1997a; WEGNER; GOLDIN, 1999a). Tais autores introduziram o conceito de Máquinas Interativas.

3.1 Evolução da Computação

A Computação está em constante evolução, impulsionada pelos avanços tecnológicos, e assim novas necessidades vão surgindo ao longo do tempo. Dentre essas necessidades, ressalta-se a reestruturação das abordagens teóricas. Van Leeuwen e Wiedermann afirmam que "o paradigma clássico de Turing pode não ser mais perfeitamente adequado para expressar todas as características da Computação atual" (2001).

Wegner e Goldin, em diversos artigos, apontam a necessidade de novos formalismos, que englobem a Computação Interativa. Um dos formalismos já aceitos pela comunidade científica para tratar interação são as *Máquinas Interativas*, ou IMs (descritas na Seção 3.3). A Computação Interativa tem tido um tratamento teórico através de Wegner em cooperação com Goldin. Wegner (2001) introduziu o conceito de Máquina Interativa, a qual é vista como alternativa para e aperfeiçoamento da computação algorítmica, sendo que esta incorpora o conceito de Máquina de Turing como noção principal.

Um amplo panorama da área de Computação Interativa é exposto por Goldin, Keil e Wegner em "*A Historical Perspective of Interactive Computing*" (2002). Nesse trabalho, os autores apontam diversos temas de pesquisa na Computação que envolvem conceitos de interatividade.

Uma das principais metas da Computação Interativa é a estabilização da base teórica para modelar atividades interativas. Considerando que a expressividade algorítmica é definida por poder computacional dos sistemas, a expressividade interativa é definida pela capacidade do observador (ambiente) de fazer distinções observacionais (WEGNER, 1998).

Ao longo do trabalho, as denominações agentes de computação finita, dispositivos computacionais, máquinas abstratas, entre outros, representam o mesmo conceito.

Tabela 3.1: Paralelo entre Algoritmos e Interação

Conceitos Algorítmicos	Conceitos Interativos
transformam entradas em saídas	serviços ao longo do tempo
programação estruturada	programação estruturada OO
sistemas fechados	sistemas abertos
Ciência da Computação algorítmica	Ciência da Computação empírica

3.1.1 Mudança de Paradigma

A mudança de paradigma de algoritmos para interação captura a mudança de tecnologia de "mainframes" para "workstations" e redes. Interação mostra-se mais poderosa que algoritmos baseados em regras, para solucionar problemas computacionais (WEGNER, 2001), assim busca-se desenvolver modelos capazes de formalizar esses novos conceitos.

O paradigma de interação provê uma nova maneira de abordar aplicações tanto na Ciência da Computação algorítmica quanto na Ciência da Computação Empírica, demonstrando a importância dos modelos interativos, como uma base para a análise dos problemas práticos na Computação (WEGNER, 1997a). Houve a necessidade de estender problemas computacionais de algoritmos para sistemas de tempo real e sistemas embutidos. Essa visão estendida da computação requer a extensão de modelos de computação de maneira que eles modelem processos interativos e de expressividade de modo que eles modelem serviços sobre o tempo. A noção intuitiva de computação deve ser expandida além da Tese de Church (WEGNER, 1999).

Explorar interação do ponto de vista computacional é descrever um modelo de computação genérico que possa interagir com o ambiente (LEEUWEN; WIEDERMANN, 2000a). O objetivo de um sistema interativo é usualmente não computar algum resultado final, mas reagir ou interagir com o ambiente onde o sistema está inserido e manter um comportamento ação-reação bem definido. Sistema Interativos estão sempre operando e podem ser vistos como máquinas que computam seqüências de palavras infinitas, mas diferem no sentido que sua entrada não é especificada e pode depender de saídas prévias e recursos externos (LEEUWEN; WIEDERMANN, 2000b).

A Tabela 3.1 mostra um paralelo entre Algoritmos e Interação (WEGNER; GOLDIN, 2001). Cada conceito algorítmico na coluna do lado esquerdo corresponde a um conceito interativo do lado direito.

Historicamente, a noção de solução de problemas tem sido representada com funções computáveis, que dão uma saída "resposta" para uma entrada "problema". Algoritmos representam o espaço de funções computáveis. Agentes de computação finita interativa são mais expressivos que Máquina de Turing, que é uma área de pesquisa que tem sido considerada fechada, e requer um novo estudo das hipóteses fundamentais sobre os modelos de computação (LEEUWEN; WIEDERMANN, 2000a).

Estudos estão surgindo sobre a questão de a Tese de Church ainda ser adequada para capturar o poder e limitações de sistemas de computação atuais (LEEUWEN; WIEDERMANN, 2000b). Segundo Church, a noção intuitiva de computabilidade efetiva para funções e algoritmos é formalmente expressada por máquina de Turing. Pode-se dizer que a Tese de Church tem a forma *a noção intuitiva*

X corresponde a noção formal Y, onde *X = computabilidade por algoritmos e Y = Máquina de Turing, Cálculo Lamba* (WEGNER; GOLDIN, 2001).

Aceita-se a Tese de Church, mas difere-se largamente dela, onde afirma-se que máquinas de Turing expressam completamente todas formas de computabilidade (WEGNER, 1997a). Wegner (1998) afirma que "A intuição de que computação corresponde a computabilidade formal com TM [...] falha quando a noção do que é computável é ampliada para incluir interação. Embora a Tese de Church seja válida no sentido restrito de que as TMs expressam comportamento algorítmico, a afirmação de que os algoritmos abrangem precisamente o que pode ser computável é inválida".

Tais controvérsias no tema estão fundadas levando em consideração as características da Computação Interativa. Algumas são abordadas na seção seguinte.

3.1.2 Características da Computação Interativa

Computação interativa pode ser caracterizada por três aspectos ausentes na computação algorítmica: entrelaçamento de entradas e saídas durante a computação; sequências de entradas e saídas; comportamento histórico-dependente de agentes de computação.

A computação interativa envolve a troca de mensagens entre máquinas computacionais e o ambiente (GOLDIN; KEIL, 2001). Conforme dito acima, uma das características da computação interativa envolve o consumo de símbolos de entrada e a produção de símbolos de saída durante a computação, ao invés de antes ou depois. Isto é conhecido como uma propriedade da interação.

Pode-se destacar outras propriedades da interação, que também são ausentes na computação algorítmica, onde a palavra de entrada é consumida antes do início da computação, e a palavra de saída é produzida depois que termina a computação (GOLDIN; WEGNER, 1998). As seguintes propriedades adicionais distinguem computação interativa e computação algorítmica:

- Ligação dinâmica: a próxima entrada fica indisponível até que a saída prévia tenha sido produzida;
- Estado de Persistência: valores internos podem ser preservados produzindo uma saída e consumindo a próxima entrada;
- Histórico-dependente: a saída corrente pode depender da entrada anterior;
- Informação secreta: noções de equivalência e expressividade são definidas sem o conhecimento do estado do sistema;
- Sem término: Computação é um processo, mais propriamente uma única transformação de entrada/saída;

3.2 Noções para Computação Interativa

Um dos primeiros passos para abordar Computação Interativa foi a extensão dos conceitos: problemas computacionais de algoritmos para sistemas embutidos e sistemas de tempo-real (WEGNER, 2001), extensão no aspecto de modelos de computação (WEGNER, 1998) de TM para IM e por fim noções da computação, que implica em reavaliação da Tese de Church (WEGNER; GOLDIN, 1999b).

Problema computacional Estender de algoritmos para sistemas reativos, embutidos e de tempo real. Problemas *algorítmicos* como ordenação, alcançabilidade de grafos, fluxo máximo e caixeiro viajante expressam decisão parametrizada e otimização de problemas que podem ser especificados por meio de funções computáveis. Problemas *interativos* como reservas aéreas, controle de robôs e reconhecimento de voz implicam serviços interativos ao longo do tempo. O comportamento de algoritmos é especificado por funções computáveis enquanto que sistemas interativos não possuem uma especificação formal de comportamento completa.

Modelo de computação Estender de Máquinas de Turing para máquinas interativas. Algoritmos são modelados por Máquinas de Turing, que transformam, de forma não interativa, palavras finitas de entrada em palavras de saída. A computação interativa é modelada por máquinas interativas, IMs, que estendem as TMs com ações de entrada e saída as quais interagem dinamicamente com o ambiente externo. As IMs não podem ser modeladas por TMs com uma fita de entrada inicial finita porque qualquer seqüência finita de entrada pode sempre ser estendida por um ambiente específico. Esta "prova" implica que as IMs são tão poderosas quanto TMs com fitas de entrada iniciais infinitas, e modelam sistemas reativos e TM com oráculo, as quais são mais expressivas que as TMs. As IMs são aqui definidas intencionalmente como máquinas com uma "propriedade interativa" que pode ser efetivamente verificada. Elas podem ter seqüência de entrada única ou múltipla, ações de entrada e saída síncronas ou assíncronas, e podem diferir em vários outros aspectos, mas todas as formas de interação expressam o comportamento de sistemas abertos. As IMs modelam objetos, agentes e computadores reais de forma mais direta que as TMs.

Noção de computação A Tese de Church, largamente interpretada, equipara a noção intuitiva de computação por algoritmos com a noção formal de computabilidade com TMs. Entretanto, a expressividade interativa de sistemas embutidos, reativos, programação orientada a agentes e reconhecimento de voz não pode ser exibida por algoritmos ou TMs. Embora os algoritmos expressem adequadamente a manipulação de números e computações isoladas de saídas a partir de entradas, o que requer expandir o modelo intuitivo de computação para incluir a interatividade.

3.2.1 Embasamento conceitual

Para compreender o texto subseqüente, alguns conceitos são aqui apresentados, pois são considerados fundamentais para o entendimento do trabalho. A noção de equivalência e expressividade de dispositivos de computação interativos estão intimamente relacionados com a noção de seus comportamentos, que é mostrado por interações com o ambiente.

O comportamento é definido por interações com o ambiente. Conforme Goldin (GOLDIN; KEIL, 2001), é possível classificar comportamento computacional por expressividade, algorítmico ou interativo e ainda o comportamento interativo pode ser classificado por seqüencial ou multi-seqüencial.

EXEMPLO 3.1: Comportamento de TM consiste de reconhecer palavras de entrada e retornar palavras de saída;

Qualquer modelo de comportamento induz a noção de equivalência para pares de dispositivos: dois dispositivos de computação são equivalentes se seus comportamentos são iguais.

Além disso, a noção de equivalência precede a noção de expressividade para classes de dispositivos: duas classes de dispositivos tem a mesma expressividade se os dois conjuntos correspondentes de comportamento são iguais. A classe de dispositivos de computação C_1 é mencionada ser menos expressiva que a classe C_2 se, para cada dispositivo em C_2, existe um equivalente em C_1. Se o inverso não for verdade, C_1 é dito ser pelo menos mais expressivo que C_2.

EXEMPLO 3.2: Para cada Autômato Finito, existe um Autômato com Pilha, mas não o inverso. Conseqüentemente Autômatos com Pilha são mais expressivos que Autômatos Finitos (HOPCROFT; MOTWANI; ULLMAN, 2001).

É importante explicitar a noção de paralelismo dada por Wegner e atribuída neste trabalho, onde paralelismo especifica a estrutura interna, sendo uma propriedade da computação de um sistema onde as ações se sobrepõem no tempo.

3.3 Máquinas para Computação Interativa

A Máquina Interativa é uma extensão da Máquina de Turing. A IM é uma TM que interage com uma ou mais seqüências durante a operação. A interação na IM é caracterizada por uma seqüência de vários pares de entrada e saída, onde a entrada na IM com tempo t pode ser influenciada pela saída anterior com tempo $t-1$. O funcionamento de uma IM entre duas interações é idêntico ao da TM (WEGNER; GOLDIN, 2001). O argumento informal que Máquina Interativa Seqüencial não é expressa por algoritmos é surpreendentemente simples, dependendo somente da noção das seqüências que são dinamicamente fornecidas.

As Máquinas Interativas são classificadas em dois tipos: Máquinas Interativas Seqüenciais (SIMs) e Máquinas Interativas Multi-seqüenciais (MIMs). A SIM é obtida pela restrição de uma única seqüência de entrada na IM, enquanto que a MIM contém várias. Existe uma forte evidência de que MIMs são mais expressivas que SIMs. Isso contrasta com o fato de que Máquinas de Turing com uma ou múltiplas fitas possuem a mesma expressividade. A prova de que MIMs são mais expressivas que SIMs requer um modelo preciso de computação com múltiplas seqüências e uma definição cuidadosa de expressividade. Na seção seguinte é apresentada a definição de uma SIM.

3.4 Máquina Interativa Seqüencial — SIM

A Máquina de Turing é um dispositivo de computação finita que, de forma não interativa, transforma símbolos de entrada em símbolos de saída por seqüência de transições de estado (DIVERIO; MENEZES, 2000).

Definição 3.1 (Máquina de Turing) *Uma Máquina de Turing M é uma 8-upla $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, *)$ onde:*

Σ *alfabeto de símbolos de entrada;*
Q *conjunto de estados possíveis da máquina, o qual é finito;*
Π *programa ou função de transição:*
$\Pi = Q \times (\Sigma \cup V \cup \{\beta, *\}) \rightarrow Q \times (\Sigma \cup V \cup \{\beta, *\}) \times \{E, D\}$;
q_0 *estado inicial da máquina, tal que* $q_0 \in Q$;
F *conjunto de estados finais, tal que* $F \subset Q$
V *alfabeto auxiliar;*
β *símbolo especial branco;*
$*$ *símbolo especial marcador de início de fita*

A configuração inicial da máquina é o símbolo de início de fita, que ocorre uma vez e sempre na célula mais à esquerda da fita, auxiliando na identificação de que a cabeça da fita se encontra na célula mais à esquerda.

A partir dessa verificação, pode-se definir a Máquina Interativa Seqüencial, que absorve características da TM (WEGNER; GOLDIN, 1999b). Uma Máquina Interativa Seqüencial(SIM) é uma máquina de transição de estados $M = (S, I, m)$ onde:

S conjunto enumerável de estados
I,O conjunto enumerável de entradas/saídas dinamicamente limitadas
m mapeamento

$$m : S \times I \rightarrow S \times O$$

Cada passo da computação $(s, i) \rightarrow (s', o)$ de uma Máquina Interativa pode ser visto como uma computação completa da TM, onde i é um símbolo (palavra) de entrada fornecida dinamicamente, a saída o pode resultar de entradas subseqüentes, e s' é o próximo estado da SIM. O comportamento da SIM é expresso por seqüências de entrada e saída.

Definição 3.2 (Seqüência de entrada/saída) *Seqüências de entradas e saídas têm a forma* $(i_1, o_1), (i_2, o_2), \ldots$, *onde* o_k *é computado conforme* i_k *mas precede e pode influenciar* i_{k+1}.

A dependência de i_{k+1} sobre o_k é chamada *ligação de entrada/saída*. A transição m proveniente de s_k para s_{k+1} pares de entrada/saída é associada com pares entrada/saída (i_k, o_k). Na Figura 3.1 pode-se visualizar como uma SIM comporta-se de acordo com a seqüência de entrada.

3.4.1 Expressividade e Comportamento Interativo da SIM

A expressividade de dispositivos de computação finitos é definida em termos da noção de comportamento observável que generaliza a noção de comportamento algorítmico.

Definição 3.3 (Comportamento do sistema) *O comportamento do sistema de um agente de computação finita* M *é especificado pelo conjunto de todas interações possíveis com todos ambientes possíveis, e é denotado por* $B(M)$

Comportamento observável de máquinas em relação a uma classe de ambientes é geralmente mais relevante para modelar aplicações interativas.

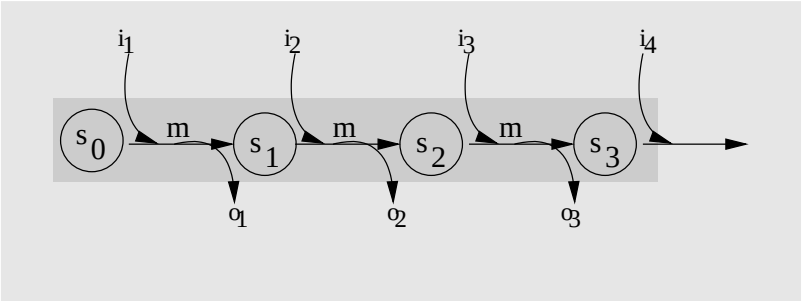

Figura 3.1: Máquina Interativa Seqüencial

Definição 3.4 (Comportamento observável) *O comportamento observável de um agente (máquina) M para um ambiente (observador) E é o conjunto de todos os comportamentos de M em E e é denotado por $B_E(M)$.*

Comportamento observável é uma projeção do comportamento de um sistema em um contexto específico. O comportamento do sistema M é definido como uma união do comportamento sobre todos os ambientes de observação possíveis

$$B(M) = \cup E \cdot B_E(M)$$

Definição 3.5 (Expressividade) *Uma máquina M_1 é mais expressiva que a máquina M_2 no ambiente E se $B_E(M_1)$ é um conjunto estritamente/rigorosamente maior de comportamento que $B_E(M_2)$*

A expressividade de agentes de computação finita pode ser formalmente definida por relações de equivalência de observação que medem a capacidade dos agentes para fazer distinções observacionais sobre seu ambiente.

Definição 3.6 (Equivalência de Observação) *Uma classe S de um sistema e uma classe E de ambientes determinam uma relação de equivalência de observação $EQ(S, E)$ tal que os sistemas $M_1, M_2 \in S$ são equivalentes se e somente se seus comportamentos são indistinguíveis para todo $e \in E$. Ou seja, $B_e(M_1) = B_e(M_2)$ para todo $e \in E$. Ambientes $e, e' \in E$ são equivalentes para sistemas S se induzem comportamento indistinguível $B_e(M) = B_{e'}(M)$ para todo $M \in S$.*

Máquinas induzem uma relação de equivalência em um ambiente E tal que a máquina M_1 é mais expressiva que a máquina M_2 se a relação de equivalência $EQ(M_1, E)$ induzida por M_1 em E é mais refinada que $EQ(M_2, E)$. Maior expressividade das máquinas implica maior distinguibilidade entre ambientes, que por sua vez significa que máquinas mais expressivas podem solucionar uma maior classe de problemas computacionais.

Expressividade observada depende do comportamento do sistema observado e do poder de observação dos observadores. Comportamento não pode ser observado a não ser que os observadores tenham capacidade de percebe-lo. Quando os observadores são limitados a interações simples então o comportamento observável é limitado ao comportamento da TM já que o comportamento interativo é inobservável.

O comportamento observado de agentes finitos pode ser significativamente especificado somente em relação a um observador, tanto para sistemas físicos quanto para sistemas computacionais. Observadores da TM podem observar somente pares I/O únicos, enquanto que observadores da SIM podem observar o comportamento de seqüências de entradas fornecidas interativamente que dependem de eventos que ocorrem dinamicamente no ambiente.

Para um estudo mais detalhado das SIMs recomenda-se os artigos de Wegner e Goldin (WEGNER; GOLDIN, 1999b; WEGNER, 1998; WEGNER; GOLDIN, 2001).

3.5 Máquina Interativa Multi seqüencial — MIM

Apresenta-se aqui a Máquina interativa multi-seqüencial (MIM) que expressa os modelos de interação de agentes autônomos, existindo evidências de serem mais expressivas que SIMs (WEGNER; GOLDIN, 1999b).

O comportamento da MIM não pode ser expresso por SIM (GOLDIN, 2000) no sentido que MIMs podem produzir uma classe maior de distinções observáveis que as SIMs. Sistemas distribuídos com agentes autônomos são modelados por MIMs. Um sistema distribuído pode ser definido como um sistema com interfaces geograficamente ou logicamente separadas. Um dos primeiros papéis dos sistemas distribuídos é de facilitar fluxo distribuído de informações entre agentes externos que observam ou acessam uma interface do sistema. Sistemas transmitem dados observados por fluxo de informação inobservável entre componentes do sistema.

Como exemplo de sistemas para MIMs, podem ser citados: sistema de reserva de passagens aéreas, a Internet, negociação multi-agente, base de dados distribuídos ou sistema ATM que fornecem serviços de múltiplos clientes autônomos.

Embora exista uma forte evidência que o comportamento das MIM não é expressado por SIMs, o comportamento é difícil de formalizar e maior expressividade é mais difícil de ser provada. MIM suporta comportamento de transações não serializáveis e concorrência verdadeira, enquanto que SIM suportam somente transações serializáveis e concorrência por interleaving/intercalação.

Definição 3.7 (Máquina Interativa Multi-Seqüenciais) *Uma MIM é um sistema de transição de estados que interage com seqüências múltiplas autônomas, ou então pode ser visto como agentes finitos que interagem com múltiplas seqüências independentes*

TMs expressam comportamento de um agente simples, SIMs expressam a interação de 2 agentes, enquanto que MIMs expressam a interação de n agentes sendo $n > 2$. MIMs produzem uma definição mais clara de sistemas distribuídos: Um sistema (agente computacional) é distribuído se e somente se suas interações podem ser descritas por uma MIM mas não por uma SIM.

Uma maior expressividade da máquina M_1 em relação à máquina M_2 para uma seqüência múltipla em um ambiente E pode a princípio ser definida observacionalmente pela propriedade que induz a relação de equivalência de observação $EQ(M_1, E)$ é preciso (precisão) que $EQ(M_2, E)$, assim como para SIMs.

Dois argumentos que sugerem uma maior expressividade das MIMs: MIMs podem ter comportamento não-determinístico, não expressível por SIMs; MIMs podem ter comportamento não serializável, também não expressível por SIMs.

Um detalhamento desses argumentos pode ser encontrado no trabalho de Wegner (1999b).

Demais particularidades sobre as MIMs podem ser encontrados na literatura referenciada. A MIM foi abordada neste capítulo para complementar o estudo de Máquinas Interativas, porém seu detalhamento não foi estendido por não se tratar diretamente do objeto de estudo do trabalho. Entretanto, indica-se o trabalho de Leeuwen e Wiedermann (2001), que tem um importante papel para as MIMs, pois a partir das definições básicas os pesquisadores desenvolveram as Máquinas Site.

3.6 Considerações sobre as Máquinas Interativas

O comportamento de uma máquina interativa pode ser observado por diferentes observadores externos (usuários, aplicações externas, ambientes, etc.). Embora as Máquinas Interativas possam coordenar a seqüência de vários ambientes, tais observadores podem não ter conhecimento das atividades uns dos outros, de modo que a saída da IM pode parecer subjetivamente não determinística para eles.

Por exemplo, um observador da TM pode somente relacionar pares de pergunta-resposta, enquanto um observador da SIM pode observar o comportamento seqüencial das seqüências de interações únicas. A interação que pode ser especificada não é necessariamente observável. Por exemplo, o comportamento de MIM é especificável, mas dificilmente observável.

Dando continuidade ao assunto, o capítulo seguinte descreve as Máquinas de Turing Persistentes, que são definidas com base nas características e conceitos das Máquinas Interativas Seqüenciais.

4 MODELOS DE INTERAÇÃO

No capítulo anterior foram expostos os principais conceitos e fundamentos em que a Teoria da Computação Interativa está baseada. Este capítulo apresenta o modelo da Máquina de Turing Persistente, que é um dispositivo de computação abstrato, sendo visto como uma extensão da Máquina de Turing para expressar comportamento interativo. Esse modelo foi definido por Wegner em cooperação com Goldin (WEGNER; GOLDIN, 1999b), e é especialmente relevante neste trabalho. O texto é baseado principalmente nos artigos *Mathematical Models of Interactive Computing*, de Peter Wegner (1999b), e *Persistent Turing Machines as a Model of Interactive Computation*, de Dina Goldin (2000), tendo como objetivo apresentar a descrição detalhada da Máquina de Turing Persistente (PeTM[1]).

4.1 A Computação Interativa de Wegner

Como mostrado no Capítulo 3, os fundamentos da Computação Interativa vêm sendo desenvolvidos em paralelo com a evolução da tecnologia na computação. Porém um grande passo no desenvolvimento da Computação Interativa teórica foi dado a partir do trabalho de Peter Wegner, que definiu as Máquinas Interativas (detalhadas na seção 3.3). A pesquisa nessa área vem despertando interesse, gerando subáreas para o desenvolvimento de outros modelos comprovando a visão alcançada por Wegner (1999).

Wegner introduziu a idéia de sistemas que interagem com o ambiente, a partir do artigo *Why interaction is more poweful than algorithms*; nele aparecem as primeiras considerações sobre computabilidade e interação, e a mudança de paradigma de algoritmos para interação. O trabalho *Mathematical models of interactive computing*, seguiu inserindo essa projeção, mas indo além, definindo máquinas interativas seqüenciais (única seqüência) e máquinas interativas multi-seqüenciais (múltiplas seqüências), conhecidas por SIM e MIM, respectivamente.

Baseado no seu estudo da Teoria da Computação Interativa, Wegner vislumbrou que adicionando uma fita de trabalho persistente na SIM seria possível manter o histórico do funcionamento da máquina. Assim, em *Behavior and Expressiveness of Persistent Tuirng Machines*, Wegner e Goldin (2000) elaboraram o que vem a ser a Máquina de Turing Persistente apresentada neste capítulo. Persistência para os autores pode ser vista como "uma habilidade do sistema de recordar enquanto

[1]Embora Wegner e Goldin utilizem a abreviatura PTM para designar a Máquina de Turing Persistente, neste trabalho é empregada a abreviatura PeTM para diferenciar da Máquina de Turing Paralela, já abordada no Capítulo 2

não realiza computações" (GOLDIN; WEGNER, 1998).

4.2 Máquina de Turing Persistente — PeTM

Nesta Seção será estendida a definição de Máquina de Turing para introduzir o conceito de persistência. Esse modelo foi desenvolvido por Wegner e Goldin (1999b).

A Máquina de Turing Persistente (PeTM) é uma Máquina de Turing com múltiplas fitas e com uma fita de trabalho persistente preservada entre as interações, na qual a seqüência de entrada gera, dinamicamente, seqüências de saída de símbolos (palavras). Esse modelo é uma extensão das Máquinas de Turing (TMs) que expressa comportamento interativo (GOLDIN; WEGNER, 2000). Assim, PeTM são dispositivos de computação interativa que admitem ações de entrada e saída durante o processo da computação (WEGNER; GOLDIN, 2001).

Definição 4.1 (PeTM) *Uma PeTM é uma máquina de Turing com múltiplas fitas e com uma fita de trabalho persistente, na qual o conteúdo é preservado entre as sucessivas computações da TM. A fita de trabalho persistente é conhecida como memória; o conteúdo da memória antes e depois da computação é conhecido como os estados da PeTM.*

Os estados da Máquina de Turing Persistente não devem ser confundidos com os estados da Máquina de Turing. Ao contrário da TM, o conjunto de estados da PeTM é infinito, representado por palavras de tamanho ilimitado.

O comportamento de uma PeTM é caracterizado por seqüências de entradas e saídas de símbolos. Os passos da computação nesse modelo não são sempre os mesmos; a saída da PeTM M e o término da computação dependem ambos da entrada e da fita de trabalho. Como resultado, M define um função recursiva parcial f_M de pares $(entrada, fita\ de\ trabalho)$ em pares $(saída, fita\ de\ trabalho)$.

$$f_M : E \times W \rightarrow S \times W$$

EXEMPLO 4.1: Uma secretária eletrônica A é uma PeTM na qual o conteúdo da fita de trabalho é uma seqüência de mensagens gravadas e na qual as operações são `record message`, `playback`, e `erase`. A função Turing-computável para A é:

$$\begin{aligned} f_A(\texttt{record}Y, X) &= (ok, XY); \\ f_A(\texttt{playback}, X) &= (X, X); \\ f_A(\texttt{erase}, X) &= (done, \varepsilon) \end{aligned}$$

4.2.1 Computação na PeTM

Os passos computacionais individuais de uma PeTM correspondem a computações da TM e são baseados em palavras. Entretanto, a semântica das computações da PeTM é baseada em seqüências, onde os elementos da seqüência são palavras sobre um alfabeto Σ. A seqüência de entrada é gerada pelo ambiente e a

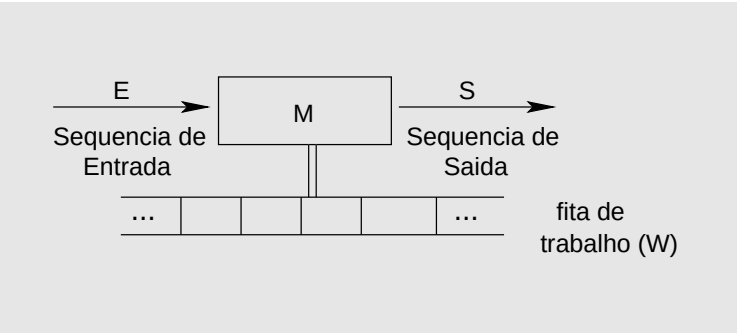

Figura 4.1: Máquina de Turing Persistente

seqüência de saída é gerada pela PeTM. As seqüências têm semântica dinâmica (tardia, "preguiçosa") de avaliação, onde o próximo valor não é gerado até que o anterior seja consumido.

Uma computação da PeTM é uma seqüência (infinita) de passos Turing-computáveis, consumindo símbolos da seqüência de entrada gerando símbolos da seqüência de saída:

Definição 4.2 (computação) *Dada uma PeTM M, definindo a função f_M e a seqüência de entradas $(i_1, i_2, ...)$, uma computação de M consiste na seqüência de passos definidos por f_M sobre a seqüência de entrada produzindo uma seqüência de saída $(o_1, o_2, ...)$. O estado da PeTM evolui durante a computação, começando no estado inicial w_0;*

$$f_M : (i_1, w_0) = (o_1, w_1); f_M : (i_2, w_1) = (o_2, w_2); f_M : (i_3, w_2) = (o_3, w_3); \ldots$$

A Figura 4.1 pode ser vista como uma PeTM.

EXEMPLO 4.2: Na máquina A do exemplo anterior, a seqüência de entrada é (`record` A, `erase`, `record` BC, `record` D, `playback`, ...) gera a seqüência de saída (*ok*, *done*, *ok*, *ok*, BCD, ...); os estados desenvolvem-se como apresentado: (ε, A, ε, BC, BCD, BCD, ...).

Esse exemplo representa uma única computação de A; o fato de que ações de entrada e saída nas seqüências ocorrerem no meio da computação caracteriza as PeTMs como dispositivos de computação interativa. O fato de que uma única seqüência de entrada ser envolvida caracteriza as PeTMs como dispositivos interativos seqüenciais, ou SIMs (WEGNER; GOLDIN, 1999b).

4.2.2 Seqüência de interações na PeTM

A interação da PeTM com o ambiente é descrita pelas seqüências de interação (entrada e saída), que alternam entradas e saídas.

Definição 4.3 (Seqüência de interação) *Têm a forma $(i_1, o_1), (i_2, o_2), \ldots$, onde is são os elementos de entrada gerados pelo ambiente (ou observador) e os são os elementos de saída gerados pela PeTM.*

Devido à semântica dinâmica de avaliação das seqüências de entrada da PeTM, assume-se que cada símbolo de entrada i_k é gerado depois da saída anterior o_{k-1}. Como resultado, os símbolos de entrada podem depender de saídas

anteriores e eventos externos no ambiente. Além disso, as saídas podem depender não somente da entrada correspondente mas de todas as anteriores, resultando em um comportamento dependente de histórico.

EXEMPLO 4.3: A computação do exemplo dado acima conduz à seguinte seqüência de interação:

$(\texttt{record}A, ok), (\texttt{erase}, done), (\texttt{record}BC, ok), (\texttt{record}D, ok), (\texttt{playback}, BCD), \ldots$

A quinta saída BCD depende da terceira e da quarta entrada, ilustrando a dependência de histórico.

O comportamento dependente de histórico de PeTMs não pode ser expressado por TMs, cuja saída é uma função da entrada. Embora a dependência de histórico possa ser simulada em TMs, tornando o histórico parte explícita da entrada, isso violaria a semântica baseada em observação da PeTM. Na modelagem de processos, sabe-se que a equivalência de observação é mais expressiva (forte) que equivalência de "rastros"; similarmente, não se pode assumir que o ato de forçar o estado da PeTM a ser observável preserva a expressividade do modelo (GOLDIN, 2000).

4.3 Modelos de Comportamento da PeTM

Assim como para qualquer TM, o comportamento da PeTM pode ser modelado de diferentes formas. Enquanto os modelos de computação para Máquinas de Turing são baseados em palavras, modelos para PeTM são baseados em seqüências de interação. Para qualquer PeTM M, diferentes modelos de comportamento podem ser observados: baseado em linguagem, baseado em autômatos, baseado em funções e baseado em ambiente.

Para qualquer PeTM M, quatro diferentes modelos de comportamento são apresentados:

- Baseado em linguagem: M é modelada por sua linguagem, que é o conjunto de todas seqüências de interações para M;
- Baseado em autômatos: M é modelada por um autômato de estados infinitos cujas transições são rotuladas por pares de palavras de entrada e saída; os caminhos através desse autômato são as seqüências de interações de M;
- Baseado em funções: M é modelada por uma função definida recursivamente de seqüências de entrada para seqüências de saída; essa função tem semântica de avaliação coindutiva, utilizando-se de recursão infinitamente sem atingir um caso base;
- Baseado em ambiente: M é modelada por um conjunto de seqüências de interações finitas ou infinitas que são produzidas pelo ambiente de M; isto traz noções de comportamento e equivalência que são relativos a ambientes.

Segundo Goldin (2000), assume-se que a PeTM é determinística, pois é baseada na Máquina de Turing determinística.

Em geral, um passo computacional da PeTM M é modelado por uma função $f_M : E \times W \rightarrow S \times W$, onde E, S e W são definidos em Σ^*. Várias sub-classes da Máquina de Turing Persistente são definidas nesta Seção, sendo sub-casos do modelo geral. As sub-classes são:

- PeTM com amnésia: esse modelo ignora o conteúdo da memória, comportando-se como se cada passo computacional inicia sempre no mesmo estado;
- PeTM com fita finita: tem um limite de memória disponível;
- PeTM com estados finitos: tem um número finito de estados;
- PeTM única: tem um conjunto finito de entrada/saída de símbolos;
- PeTM consistente: produz sempre o mesmo símbolo de saída para um dado símbolo de entrada no contexto de uma única seqüência de interação; diferentes seqüências de interação podem ter diferentes saídas para uma dada entrada.

4.4 Modelos baseados em linguagem da computação da PeTM

Nesta Seção, considera-se o modelo baseado em linguagem da computação da PeTM.

4.4.1 Linguagem da PeTM

O comportamento das Máquinas de Turing e outros modelos de computação baseados em símbolos podem ser descritos por uma linguagem, que é o conjunto de símbolos sobre um alfabeto finito.

Definição 4.4 (Linguagem) *O conjunto de todas as seqüências de interação (Definição 4.3) para uma PeTM M constitui sua linguagem, $\mathcal{L}(M)$.*

Apesar da mudança na semântica de palavras para seqüências, as definições de equivalência e expressividade aplicam-se às PeTMs assim como para outros dispositivos computacionais onde uma linguagem é definida:

Definição 4.5 (Equivalência e expressividade de PeTMs baseadas em linguagem) *Sejam M_1 e M_2 máquinas cujo comportamento é modelado por conjuntos de seqüências de interação; M_1 e M_2 são equivalentes se e somente se $\mathcal{L}(M_1) = \mathcal{L}(M_2)$.*

Sejam C_1 e C_2 classes de dispositivos de computação cujo comportamento é modelado por conjuntos de seqüências de interações; C_2 é mais expressivo que C_1 se o conjunto de todas as linguagens para elementos de C_1 é um subconjunto próprio de C_2.

4.4.2 PeTM com amnésia e TM

Uma PeTM com amnésia ignora o conteúdo da memória, comportando-se como se cada passo computacional iniciasse no mesmo estado. Conseqüentemente, as computações da PeTM com amnésia não têm dependência de histórico.

Definição 4.6 *Uma PeTM M é uma máquina com amnésia se e somente se $f_M(i, w) = f_M(i, w_0)$ para todas entradas i e estados w, onde w_0 é estado inicial da máquina M.*

Para se comparar a linguagem da PeTM com amnésia com a Máquina de Turing com múltiplas fitas, é necessário considerar a computação da TM sobre seqüências de palavras de entrada:

Definição 4.7 (Computação da TM baseada em seqüência) *Dada uma TM M definida pela função f_M com palavras de entradas e saídas, e uma seqüência de entrada $(i_1, i_2, \ldots)$, uma computação de M produz uma seqüência de saída $(o_1, o_2, \ldots)$, onde $f_M(i_k) = o_k$ para cada k.*

Uma PeTM com amnésia não é mais expressiva que uma Máquina de Turing baseada em seqüência:

Proposição 4.1 *O comportamento de qualquer PeTM com amnésia é equivalente a uma seqüência de computações da TM para a TM correspondente.*

Prova: Dada uma seqüência de entrada $(i_1, i_2, \ldots)$, a computação de uma PeTM com amnésia produz a seqüência de saída $(o_1, o_2, \ldots)$, onde:

$$f_M(i_1, w_0) = (o_1, w_1); f_M(i_2, w_1) = f_M(i_2, w_0) = (o_2, w_2);$$
$$f_M(i_3, w_2) = f_M(i_3, w_0) = (o_3, w_3); \ldots$$

Desde que w_0 é assumido como vazio, cada passo computacional de M é o mesmo que uma computação na TM que corresponde a M, chamada M_0: para todos k, $f_M(i_k) = f_M(i_k, w_0) = o_k$.

O contrário da Proposição 4.1 não é verdadeiro; uma Máquina de Turing com múltiplas fitas pode escrever algo na fita de trabalho e falhar ao apagar esse mesmo símbolo ao final da computação, então essa PeTM não é necessariamente amnésica. Mesmo assim, pode-se demonstrar que PeTMs são no mínimo tão expressivas quanto as TMs.

Proposição 4.2 *Para cada TM M, existe uma TM M' de modo que o comportamento da máquina M baseado em seqüência é equivalente ao da PeTM com amnésia baseado na M'.*

Prova: Para qualquer TM, uma máquina equivalente pode ser construída, sempre apaga a fita de trabalho no final da computação, garantindo que o comportamento correspondente da PeTM é amnésico.

Teorema 4.1 *A PeTM com amnésia tem a mesma expressividade que seqüências de computações da TM*

Prova: A prova deriva das Proposições 4.1 e 4.2

4.4.3 PeTM com memória finita e estado finito

A PeTM com memória finita tem um limite de memória disponível; isto refere-se ao número de bits que pode ser armazenado de forma persistente, sem implicar em limite no tamanho total das fitas de trabalho internas.

Em contraste, a PeTM com estado finito tem um número finito de estados; ou seja, o conjunto de todos os estados alcançáveis a partir do estado inicial via alguma seqüência de símbolos de entrada é finito.

Teorema 4.2 *PeTM com memória finita tem a mesma expressividade da PeTM com estado finito.*

Prova: Seja M uma PeTM com memória finita. Se n é o limite da memória de M, e c é o tamanho do alfabeto Σ de M, então M pode ter no máximo cn estados. Analogamente, seja M uma PeTM com estado finito. Se W é um conjunto de estados de M e k é o tamanho de W, então no máximo $\log k$ símbolos são necessários para representar algum estado de W. Portanto, pode-se construir uma PeTM equivalente a M a qual tem um limite na quantidade de memória disponível.

4.5 Modelo de comportamento da PeTM baseado em autômato

Nesta Seção, considera-se o modelo de autômato para PeTM, vista como um transdutor sobre um espaço infinito de estados.

4.5.1 Autômato com estado finito e seu índice

Todos os autômatos são baseados em estados, com transições rotuladas entre os estados, e possivelmente com valores associados a esses estados. Os autômatos de estados finitos (FSA) são os mais conhecidos, onde o conjunto de estados é finito, as transições de estados são símbolos de um alfabeto de entrada finito, e os estados são associados com um de dois valores, aceita e rejeita (HOPCROFT; MOTWANI; ULLMAN, 2001).

Um índice de um FSA é uma métrica para classificação dos FSAs; refere-se ao número de classes de equivalência induzidas pela linguagem do FSA.

Definição 4.8 (Autômato com índice) *Dado qualquer FSA A sobre um alfabeto Σ, onde $L(A)$ é o conjunto de todas as palavras aceitas por A, seja $\mathcal{P}(A)$ uma partição do alfabeto Σ^* em classes equivalentes, definidas conforme:*

Para qualquer $x, y \in \Sigma^*$, x e y pertencem à mesma classe de equivalência em $\mathcal{P}(A)$ se e somente se para todas $w \in \Sigma^*$, $xw \in L(A)$ se e somente se $yw \in \Sigma^*$.

Então, o tamanho de $\mathcal{P}(A)$ é conhecido como índice de A. Sabe-se que o índice do FSA A é finito (GOLDIN, 2000);

FST estendem os FSAs, substituindo o valor binário em cada estado por um símbolo de saída de um alfabeto de saída finito Δ. Esse símbolo de saída pode ser associado com os estados, como a Máquina de Moore, ou com transições, como a Máquina de Mealy. Tanto para Máquina de Moore quanto para Máquina de Mealy, a saída é uma seqüência de símbolos do mesmo tamanho da entrada. Sabe-se que essas máquinas têm a mesma expressividade (MENEZES, 2001). A entrada do FST também pode ser uma seqüência infinita de símbolos; a saída vai ser outra seqüência de símbolos.

A noção de índice pode ser estendida para transdutores [2], como segue:

Definição 4.9 (Índice dos transdutores) *Dado qualquer transdutor A sobre um conjunto de símbolos de entrada/saída em Σ e Δ, que mapeia seqüencias em Σ para seqüências*

[2] Conforme Goldin, uma transdutor é uma máquina de estado finito

em Δ por um mapeamento ϕ_A, seja $\mathcal{P}(A)$ uma partição de Σ^* em classes de equivalência, definida conforme:

Para qualquer $x, y \in \Sigma^*$, x e y pertencem à mesma classe de equivalência em $\mathcal{P}(A)$ se e somente se para todas $w \in \Sigma^*$, o último símbolo de $\phi_A(x, w)$ e $\phi_A(y, w)$ é o mesmo.

Então, a cardinalidade de $\mathcal{P}(A)$ é conhecido como índice de A.

A definição acima aplica-se tanto se o conjunto de símbolos é finito (um alfabeto) ou infinito (seqüências de palavras), e se o conjunto de estados é finito ou infinito. No caso dos FSTs, quando o conjunto de símbolos e o conjunto de estados são finitos, o índice também é finito.

4.5.2 PeTM e FST

Para qualquer PeTM M com estados finitos com uma função computável f_M, pode-se construir uma Máquina de Mealy A equivalente baseada em seqüências e vice-versa. A intuição por trás de tal construção é a seguinte:

Os estados de M correspondem aos estados de A, onde w_0 é estado inicial de A;

Para quaisquer dois estados w_1 e w_2, e quaisquer pares de entrada/saída de símbolos i, o, $f_M(i, w_1) = (o, w_2)$ se e somente se existe uma transição entre w_1 e w_2 em A etiquetado $''i/o''$ se e somente se $f_M(i, w_1) = (o, w_2)$.

Como resultado, essas duas classes de dispositivos têm a mesma expressividade:

Proposição 4.3 *Uma PeTM com estado finito tem a mesma expressividade que uma Máquina de Mealy baseada em seqüências.*

Prova: Conforme a construção acima. O Corolário da proposição 4.3, PeTM de estado finito tem um índice finito (conforme definição 4.9). O mesmo não é verdadeiro para a classe de todas PeTMs:

Proposição 4.4 *Existem PeTMs simples cujos índices são infinito;*

Prova: A prova que segue é por construção. Seja M uma PeTM binária cujas palavras são como segue:

Enquanto as seqüências de entrada de M é igual ao modelo $\sigma = 0101^2\ 01^30\ 1^40\ldots$, os bits de saída de M são todos $1's$; assim que exista um bit de entrada incorreto, a saída em M fica todo $0's$.

Seja $\mu(x)$ a saída produzida por M para qualquer seqüência de entrada binária x. Seja L o conjunto de todos os prefixos de σ que terminam por 0; L é enumerável e infinito. É simples ver que dado quaisquer pares x, y de elementos distintos de L, há uma seqüência binária w com a forma 1^n para $n \geq 1$, na qual o último símbolo de $\mu(x.w)$ é diferente do último símbolo de $\mu(y.w)$.

Teorema 4.3 *A classe de PeTMs simples (binárias) é mais expressiva que a classe de FST (binários).*

Prova: A prova deriva as proposições 4.3 e 4.4

Observa-se que para qualquer estado de qualquer PeTM, a relação de transição é finita (seu tamanho é limitado pelo número de pares de símbolos de entrada e saída), assim como para um FST. A expressividade extra de PeTM é assim obtida não somente da computabilidade de Turing das transições individuais, mas do estado interno ser de tamanho ilimitado.

4.5.3 PeTM e LTS

Existem estudos sobre uma classe de sistemas de transição com conjunto infinito de estados conhecida como Sistemas de Transição Rotulados (LTS — *Labeled Transition System*). A função de transição de um LTS associa a cada estado um conjunto de possíveis *ações*, e especifica um novo estado alcançado com cada ação. Assume-se que as ações são observáveis, ao passo em que os estados não são. Os LTSs são uma ferramenta padrão para modelagem de processos (MILNER, 1989). Dependendo da natureza do processo modelado as ações podem representar entradas, saídas ou uma combinação de ambas.

Define-se uma subclasse de LTSs chamada, LTS *efetivo*.

Definição 4.10 *Um LTS é efetivo se cada ação é um par de símbolos entrada/saída, e a função de transição Turing-computável; ou seja, existe uma função Turing-computável δ que, para qualquer par $(estado, entrada)$, retorna o par $(novo\ estado, saida)$.*

Percebe-se que a classe de PeTMs é equivalente à classe de LTS efetivos pela semântica de equivalência de rastros, onde os símbolos nas seqüências de entrada e saída da PeTM são os mesmos que as etiquetas ao longo dos caminhos do LTS efetivo correspondente:

Proposição 4.5 *Para qualquer PeTM M, existe um LTS efetivo equivalente T, e vice-versa. M e T são considerados equivalentes se $L(M)$ é o mesmo que o conjunto de rastros de T.*

Prova: Essa prova é similar à construção do FST apresentada na Seção 4.5.2 e está em (GOLDIN, 2000).

4.6 Modelando comportamento da PeTM por Função definida recursivamente

Como mencionado anteriormente, o comportamento da PeTM pode ser modelado como um mapeamento definido recursivamente de seqüências de entrada e de saída. Nesta Seção define-se um mapeamento ϕ_M, para qualquer PeTM.

Primeiro, definem-se duas funções sobre pares, *1st* e *2nd*, retornado o primeiro e segundo componente de cada par, respectivamente. Definem-se ainda três funções sobre seqüências, *head*, *tail* e *append*; A função *head*(σ) retorna o primeiro símbolo de uma seqüência σ, a função *tail*(σ) retorna o resto da seqüência, e a função *append*(s, σ) concatena um símbolo s no início da seqüência σ, retornando uma nova seqüência.

Assume-se que as seqüências são infinitas; sucessivas chamadas da função *tail* continuam retornando novas seqüências. Também assume-se que essas seqüências são ligadas dinamicamente: *head* e *tail* têm semântica de avaliação "preguiçosa". A ligação dinâmica de entradas e saídas de símbolos é que permite à PeTM

ser verdadeiramente interativa: o próximo símbolo é gerado somente depois que a saída prévia é produzida, permitindo que os símbolos de entrada e saída sejam interdependentes.

Seja M uma PeTM onde f_M é a correspondente função computável:

$$f_M : E \times W \rightarrow S \times W$$

Se ι é uma seqüência de entrada e w é o estado corrente de M, o primeiro passo da computação de M produzirá $f_M(head(\iota), w)$. Seja o o primeiro símbolo a sair e w' o novo estado de M, os mesmos são definidos conforme:

$$\begin{aligned} o &= 1st(f_M(head(\iota), w)) \\ w' &= 2nd(f_M(head(\iota), w)) \end{aligned}$$

Então pode-se definir a função $frec_M$, que recebe uma seqüência de entrada ι e um estado w e retorna uma seqüência de saída; $frec_M$ é definida recursivamente conforme:

$$frec_M(\iota, w) = append(o, frec_M(tail(\iota), w'))$$

Embora $frec_M$ retorne com sucesso a seqüência de saída, ela recebe o estado como uma das entradas, o que não é o desejado. O mapeamento desejado, ϕ_M, é obtido aplicando-se o estado inicial w_0 de M em $frec_M$:

$$\forall x, \phi_M(x) = frec_M(x, w_0)$$

Essa definição de ϕ_M é baseada em princípios coindutivos, que são mais expressivos que os princípios indutivos implícitos nas funções Turing-computável (WEGNER; GOLDIN, 2000). Como resultado, ϕ_M não pode ser expressado por uma função Turing-computável.

Teorema 4.4 *Para quaisquer PeTMs M_1 e M_2, $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ se e somente se $\phi_{M_1} = \phi_{M_2}$*

A prova, segundo Goldin, usa conceitos de coindução e métodos coalgébricos (JACOBS; RUTTEN, 1997).

4.7 Abordagem baseada em ambiente para Comportamento da PeTM

O comportamento da PeTM tem sido definido como uma noção absoluta (completa). Nesta Seção considera-se uma noção relativa do comportamento da PeTM, definida com relação aos ambientes das PeTMs. Isso gera uma abordagem diferente para a noção de expressividade, que contempla as classes de equivalência comportamental em diferentes ambientes.

Acredita-se que essa abordagem mostra-se mais efetiva para a formalização de sistemas interativos do que a abordagem absoluta (WEGNER; GOLDIN, 1999b).

A noção de ambiente apresentada é similar à noção de observador dada por Wegner e Goldin (2001). Considerando que a mesma PeTM pode ser observada por múltiplos observadores, assume-se que um ambiente permanece o mesmo ao longo da "vida" da PeTM (isso não quer dizer que o ambiente seja constante; ao contrário, ele obedece o mesmo conjunto de restrições).

4.7.1 Equivalência e Expressividade relativas

A seqüência de entrada de qualquer PeTM é gerada por um ambiente. Até aqui, foi assumido que todas as seqüências de entrada são possíveis. Entretanto, não se pode assumir que, no contexto onde os sistemas interativos normalmente executam, isso seja correto. Um ambiente típico terá restrições na seqüência de entrada ou nas seqüências que o mesmo é capaz de gerar; pode-se assumir que uma PeTM operando em um ambiente somente vai receber seqüências que satisfaçam as restrições apropriadas para aquele ambiente.

Identifica-se a noção de um ambiente O com as entradas que são possíveis em O; dentro de um dado ambiente, as PeTMs podem ser distinguíveis ou parecerem equivalentes:

Definição 4.11 (Ambiente) *Dada uma classe C de dispositivos computacionais com comportamento B, um ambiente O para C é um mapeamento de elementos de C para algum domínio β_O, $O : C \rightarrow \beta_O$, que é consistente/compatível, isto é:*

$$\forall\ M_1, M_2 \text{ em } C, \text{ se } B(M_1) = B(M_2), \text{ então } O(M_1) = O(M_2)$$

Os elementos de β_O são conhecidos como comportamentos possíveis (dentro do ambiente O). Quando $O(M_1) \neq O(M_2)$, diz-se que M_1 e M_2 são distinguíveis em O; caso contrário, diz-se que M_1 e M_2 parecem equivalentes em O.

A consistência de um ambiente significa que sistemas equivalentes parecerão equivalentes em todos ambientes; entretanto, sistemas que pareçam equivalentes em alguns sistemas podem ser distinguíveis em outros.

EXEMPLO 4.4: Assume-se que todos os símbolos de entrada da Secretária Eletrônica A inicia com um dos três comandos `record`, `playback`, `erase`. Isso é uma consideração correta em um ambiente onde os comandos são selecionados pelo acionamento de um de três botões possíveis. Se isso não é mais válido, a definição da secretária eletrônica pode ser estendida de várias maneiras. Por exemplo, se um símbolo de entrada não inicia com um dos três comandos, uma secretária eletrônica robusta iria ignorá-lo; uma não-robusta poderia apagar o conteúdo da fita. Embora essas máquinas não sejam equivalentes, elas parecerão equivalentes em qualquer ambiente onde somente esses comandos são possíveis.

Equivalência e expressividade de classes de modelos também podem ser definidas em relação a um ambiente, conforme apresentado a seguir:

Definição 4.12 *Dado algum ambiente O, uma classe C_2 de dispositivos computacionais parece no mínimo tão expressiva em O como a classe C_1, se, para cada máquina $M_1 \in C_1$, existe $M_2 \in C_2$ tal que $O(M_1) = O(M_2)$. C_1 e C_2 parecem equivalentes em O se C_1 também parece no mínimo tão expressivo quanto C_2 em O; de outra forma, C_2 parece mais expressiva em O.*

É possível que duas classes possam parecer equivalentes em alguns ambientes, mas uma será mais expressiva em um ambiente diferente. Será apresentado nessa seção que esse caso para Máquina de Turing e PeTM, que parecem equivalentes em ambientes algorítmicos.

4.7.2 Ambientes finitos de PeTM

Ao contrário das noções absolutas de comportamento da PeTM, onde todas eram baseadas em seqüências infinitas, os ambientes não têm que ser infinitos. Alguns ambientes podem ter um limite natural no seu tempo de vida (e o tempo de vida de qualquer PeTM que existe lá), e assim um limite no tamanho da seqüência de entrada que ele pode gerar.

Definição 4.13 (Observações e ambientes de PeTM finitos) *Dada uma PeTM M arbitrária, qualquer prefixo finito de uma seqüência de interação de M(4.2.2) é uma observação de M; seu comprimento é o numero de pares que a constitui. Um mapeamento da PeTM para conjuntos de suas observações constitui um ambiente finito da PeTM. Quando um ambiente finito admite observações de tamanho 1 apenas, então é conhecido como algorítmico.*

Ambientes algorítmicos são portanto ambientes finitos com o período de vida o mais curto possível, onde a instanciação do dispositivo de computação é para uma única interação. Essa definição de ambiente algorítmico é consistente com a de Wegner (1998); ele reflete a visão algorítmica tradicional da natureza dos dispositivos computacionais, como os apresentados em Teoria da Computação (SIPSER, 1997).

Se M_1 e M_2 são distinguíveis em um ambiente finito O, então $O(M_1) \neq O(M_2)$, isto é, $O(M_1) \oplus O(M_2)$ não é vazio. Os elementos desse conjunto são conhecidos como certificados de distinguibilidade para (M_1, M_2):

Definição 4.14 (Certificados de distinguibilidade) *Certificados de distinguibilidade são observações que são possíveis em O para uma das máquinas, mas não para a outra.*

Embora, a equivalência de duas máquinas seja intratável e não possa ser provada (SIPSER, 1997) (assim como para TM, similar ao problema da parada), não-equivalência pode ser estabelecida encontrando-se seu certificado de distinguibilidade:

Proposição 4.6 *Quaisquer duas PeTMs não-equivalentes são distinguíveis por algum certificado de distinguibilidade de tamanho finito.*

Prova: Sejam M_1 e M_2 PeTMs que não são equivalentes por linguagem; isto é, onde $\mathcal{L}(M_1) \neq \mathcal{L}(M_2)$. S.p.g., deve existir uma seqüência de interação $\sigma \in \mathcal{L}(M_1)$ que não está em $\mathcal{L}(M_2)$. Deve existir algum k tal que o prefixo de tamanho k de σ não é um prefixo de nenhuma seqüência de $\mathcal{L}(M_2)$. Seja o menor k possível. Então o prefixo de tamanho k de σ é o certificado de distinguibilidade para (M_1, M_2).

Segue das definições que se duas PeTMs são distingüíveis por alguma observação finita, deve existir algum ambiente finito onde elas são distingüíveis. Para PeTMs amnésicas, ambientes algorítmicos servem para esse propósito.

Proposição 4.7 *Quaisquer duas PeTMs com amnésia (Definição 4.6) não equivalentes são distinguíveis por um certificado de tamanho 1.*

Prova: Sejam M_1 e M_2 PeTMs com amnésia que não são equivalentes por linguagem; seja ω um certificado de distinguibilidade para (M_1, M_2), obtido de acordo com a Proposição 4.6, e seja (i, o) o último par entrada/saída em ω. Da definição de PeTM amnésica (Definição 4.6), pode ser mostrado que em qualquer ambiente capaz de gerar o símbolo i, (i, o) também serve como certificado de distinguibilidade entre M_1 e M_2.

A Proposição 4.7 assegura que ambientes algorítmicos são suficientes para distingüir todas as PeTMs amnésicas não-equivalentes. Já que a classe de TMs tem a mesma expressividade que a classe de PeTMs amnésicas (Proposição 4.1), os ambientes algorítmicos são da mesma forma suficientes para distingüir todas as TMs. O mesmo não se aplica à classe geral das PeTMs; adiante será mostrado que existem pares de PeTMs com certificados de distinguibilidade arbitrariamente longos.

4.7.3 Hierarquia de expressividade infinita

Qualquer ambiente O para a classe de dispositivos C com comportamento B induz um particionamento de C em classes de equivalência, onde os elementos de cada classe parecem equivalentes em O; tais classes são chamadas de classes de equivalência comportamental. Dada uma classe C de dispositivos, um ambiente O_1 é dito ser mais rico que O_2 se menos modelos em C parecerem equivalentes nele.

Definição 4.15 *Um ambiente O_1 é mais rico que O_2 se as classes de equivalência comportamental para O_1 são estritamente mais finas que as de O_2.*

Define-se uma seqüência infinita Θ de ambientes finitos da PeTM, $\Theta = (O_1, O_2, \ldots)$, como a seguir:

Definição 4.16 $\Theta = (O_1, O_2, \ldots)$ *onde, para qualquer k e qualquer PeTM M, $O_k(M)$ é o conjunto de observações de M de tamanho $\leq k$.*

O_k dá a noção relativa do comportamento de M, com relação aos ambientes que podem gerar no máximo k símbolos de entrada. O_1 é um ambiente algorítmico, com a noção mais grossa de equivalência de comportamento.

É fácil ver que para qualquer k, O_k é consistente (Definição 4.11): para qualquer par de PeTM M_1 e M_2, se $B(M_1) = B(M_2)$, então $O_k(M_1) = O_k(M_2)$. É também o caso que, para as classes de PeTMs, para qualquer k, O_{k+1} é mais rico que O_k:

Proposição 4.8 *Para qualquer k, O_{k+1} é maior que O_k*

Prova: Pode ser demonstrado que $O_{k+1}(M_1) \neq O_{k+1}(M_2)$ sempre que $O_k(M_1) \neq O_k(M_2)$. A prova é completada construindo uma seqüência de PeTMs $(M_1, M_2, ...)$ tal que para qualquer k, M_k e M_{k+1} parecem equivalentes para O_k mas não para O_{k+1}. Essas PeTMs são binárias, com entrada 0 e 1, e saídas V e F. Para qualquer k, M ignora a entrada gerando na saída k Fs seguidos por todos os Vs. O mais curto certificado de distinguibilidade para (M_k, M_{k+1}) é de tamanho $k + 1$, terminando com saída V em um caso e F em outro.

A Proposição 4.8 mostra que os ambientes em Θ são cada vez mais ricos, produzindo classes de equivalência cada vez mais finas para comportamentos da PeTM sem atingir um limite.

Teorema 4.5 *Os ambientes em Θ induzem uma hierarquia de expressividade infinita de comportamentos da PeTM, com comportamento da TM como sendo o primeiro elemento da hierarquia.*

Prova: Segue das proposições 4.8 e 4.7.

Todos os ambientes em Θ são finitos. Em contraste, os modelos absolutos de comportamento da PeTM definidos anteriormente não são finitos. As classes de equivalência da PeTM induzidas pelos modelos absolutos podem ser observadas como o limite para a hierarquia de expressividade infinita do Teorema 4.5.

4.8 Considerações sobre a PeTM

Esta seção tem a intenção de mencionar as relações e comparações já produzidas anteriormente em outros trabalhos. Os detalhes de cada demonstração são debatidos em vários artigos. Por exemplo, a equivalência entre as PeTMs e SIMs pode ser analisada em (GOLDIN; WEGNER, 1998), assim como a equivalência das TMs e SIMs. Pode-se relacionar uma Máquina de Turing com uma Máquina Interativa Seqüencial, comparando as palavras e seqüências de palavras dos modelos, respectivamente. A representação dessa relação pode ser encontrada em (WEGNER, 1997b).

A visão manifestada por Wegner sobre os modelos interativos, e bastante contestada no artigo *Why Church's Thesis Still Holds — Some Notes on Peter Wegner's Tracts on Interaction and Computability*, de Prasse e Rittgen (1998). Essa colocação é acrescentada no texto para mostrar outra visão de computabilidade e máquinas interativas.

5 FORMALIZAÇÃO DA PETM-PAR

Neste capítulo é apresentado o desenvolvimento principal do trabalho. Reunindo todos os conceitos apresentados anteriormente, apresenta-se uma variação do modelo de Máquina de Turing Persistente, abordada no texto como Máquina de Turing Persistente Paralela, que estende o número de fitas de trabalho. Apresenta ainda, o comportamento da PeTM-Par modelado por linguagem, função e ambiente.

5.1 Máquina de Turing Persistente Paralela — PeTM-Par

Nesta seção apresenta-se a definição da Máquina de Turing Persistente Paralela — PeTM-Par — estendendo o modelo da Máquina de Turing Persistente, já apresentada no Capítulo 4. Primeiramente, esse modelo é uma extensão da Máquina de Turing Persistente, que possui n fitas de trabalho.

Definição 5.1 (PeTM-Par) *Uma máquina de Turing Persistente Paralela é uma Máquina de Turing Persistente com múltiplas fitas de trabalho, que por sua vez é uma Máquina de Turing com múltiplas fitas e uma fita de trabalho, na qual o conteúdo é preservado entre as sucessivas computações da Máquina de Turing.*

Assume-se que a descrição dada sobre a máquina de Turing no Capítulo 4 é válida, já que a PeTM-Par é uma extensão da PeTM. O conteúdo da fita de entrada é dito estar sob o controle do ambiente (observador), enquanto que as outras fitas são controladas pela PeTM-Par. Além disso, o conteúdo das fitas de entrada e saída é observável enquanto que o das fitas de trabalho não é.

O comportamento de uma PeTM-Par é caracterizado por seqüências de entradas e saídas de símbolos. A equivalência e expressividade são definidas em relação ao comportamento.

A Figura 5.1 pode ser vista como uma PeTM-Par. Os passos da computação nesse modelo não são sempre os mesmos, a saída da PeTM-Par M e o término da computação dependem ambos da entrada e do conteúdo das fitas de trabalho. Assim M define uma função:

$$f_M : (E \times \langle w_1, w_2, \ldots, w_n \rangle) \rightarrow (S \times \langle w_1, w_2, \ldots, w_n \rangle)$$

5.1.1 Computação na PeTM-Par

A notação da computação para uma Máquina de Turing Persistente Paralela é análoga ao da PeTM que, conforme o Capítulo 4 é análoga ao da TM. Então,

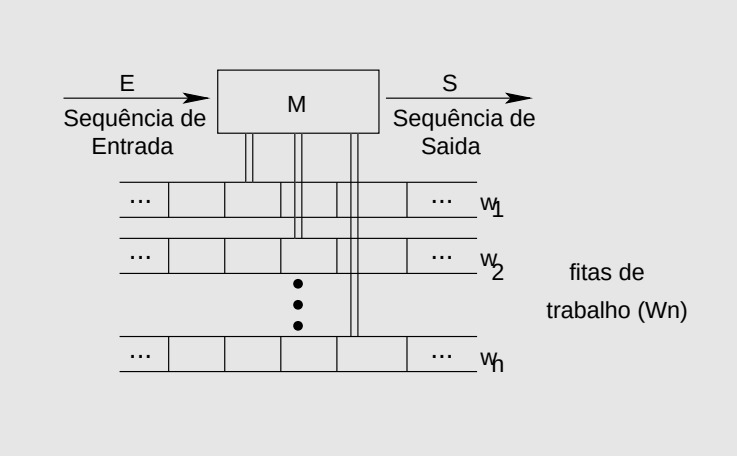

Figura 5.1: Máquina de Turing Persistente Paralela

pode-se dizer que um passo computacional da PeTM-Par corresponde a computações da TM e são baseadas em palavras. A semântica da computação da PeTM-Par é baseada em seqüência, onde a seqüência de símbolos são palavras sobre o alfabeto Σ.

Definição 5.2 (computação) *Dada uma PeTM-Par M, definindo a função f_M e a seqüência de entrada E, onde $E = (e_1, e_2, ...)$, uma computação de M consiste na seqüência de passos produzindo uma seqüência de saída: $S = (s_1, s_2, ...)$. O estado de cada fita de trabalho da PeTM-Par evolui durante a computação. Um* estado global *da máquina pode ser definido como o conjunto de estados das n fitas.*

Conforme a definição de computação dada para a PeTM, assume-se que o estado inicial de cada fita é uma ε (palavra vazia):

$$\begin{aligned} f_M : (e_1, \langle \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \rangle) &= (s_1, \langle w_1, w_2, \ldots, w_n \rangle); \\ (e_2, \langle w_1, w_2, \ldots, w_n \rangle) &= (s_2, \langle w'_1, w'_2, \ldots, w'_n \rangle); \\ (e_3, \langle w'_1, w'_2, \ldots, w'_n \rangle) &= (s_3, \langle w''_1, w''_2, \ldots, w''_n \rangle); \ldots \end{aligned}$$

5.1.2 Seqüência de interações na PeTM-Par

A interação da PeTM-Par com o ambiente é descrita pelas seqüências de interação de entrada e saída. Conforme a seção 4.2.2, uma seqüência de interação da PeTM-Par é análoga àquela definida para uma PeTM, já apresentada na Definição 4.3. A Figura 5.2 representa a seqüência de interação da PeTM-Par.

5.1.3 Operações pertinentes para a PeTM-Par

Para caracterizar de forma clara o paralelismo nas operações sobre as múltiplas fitas, foram definidas operações genéricas para manipulação das mesmas. Nesse modelo, apresentam-se três tipos de operações passíveis de serem executadas por uma PeTM-Par. Essas operações variam em termos de flexibilidade na realização de operações sobre as múltiplas fitas.

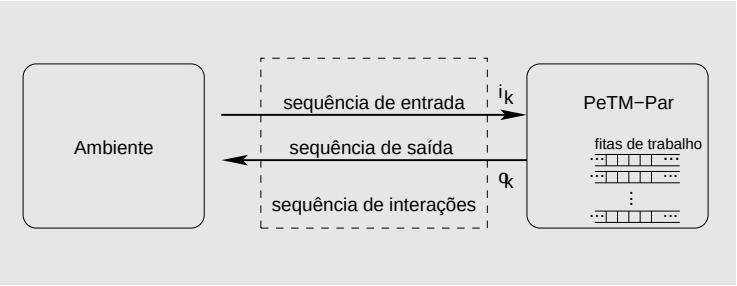

Figura 5.2: Seqüência de Interação da PeTM-Par

As operações definidas são `op`, $\texttt{op}_{sub}$ e `meta-op`. Uma `op` é uma operação simples que tem como entrada n parâmetros que serão aplicados nas n fitas. Outra operação é a $\texttt{op}_{sub}$, que pode ser definida como um conjunto de sub-operações, sendo que cada operação manipula um parâmetro em cada fita. E por fim, define-se uma meta-operação como um conjunto de operações, que no texto será atribuído `meta-op`. Essa operação generaliza a construção dada para as demais, podendo ter n operações atuando em n fitas e possibilitando também manipular m parâmetros.

Definição 5.3 (*op*) *Uma operação* `op` *é uma operação com n parâmetros em n fitas de trabalho. Essa operação difere da* $\texttt{op}_{sub}$ *pois é projetada para casos mais simples, onde não há necessidade de passagem explícita de parâmetros.* `op` *é definida conforme:*

$$\texttt{op}([\langle x, y, \ldots, z\rangle], [\langle w_1, w_2, \ldots, w_n\rangle]) = ([s_{op}], [\langle w_x, w_y, \ldots, w_z\rangle])$$

Definição 5.4 ($\texttt{op}_{sub}$) *Uma operação* $\texttt{op}_{sub}$ *é um conjunto de sub-operações* $\texttt{op}_{sub} = \{op_1, op_2, \ldots, op_n\}$*, definida como segue:*

$$\begin{aligned}\texttt{op}_{sub}(\langle x, y, \ldots, z\rangle)\ (&\\ &[\langle op_1(x), op_2(y), \ldots, op_n(z)\rangle], [\langle w_1, w_2, \ldots, w_n\rangle]) =\\ &([s_{op_{sub}}], [\langle w_x, w_y, \ldots, w_z\rangle])\end{aligned}$$

Definição 5.5 (meta-op) *Uma operação* `meta-op` *é um conjuntos de operações* $\texttt{meta-op} = \{op_1, op_2, \ldots, op_n\}$ *definida como mostrado a seguir:*

$$\begin{aligned}\texttt{meta-op}\ \ &(\langle x_1, y_1, \ldots, m_1\rangle, \langle x_2, y_2, \ldots, m_2\rangle, \ldots, \langle x_n, y_n, \ldots, m_n\rangle)\ (\\ &[op_1, op_2, \ldots, op_n],\\ &[\langle x_1, y_1, \ldots, m_1\rangle, \langle x_2, y_2, \ldots, m_2\rangle, \ldots, \langle x_n, y_n, \ldots, m_n\rangle],\\ &[\langle w_{1_1}, w_{1_2}, \ldots, w_{1_n}\rangle, \langle w_{2_1}, w_{2_2}, \ldots, w_{2_m}\rangle, \ldots, \langle w_{n_1}, w_{n_2}, \ldots, w_{n_m}\rangle])\\ &=\\ &([s_{meta-op}],\\ &[\langle w_{x_1}, w_{y_1}, \ldots, w_{m_1}\rangle, \langle w_{x_2}, w_{y_2}, \ldots, w_{m_2}\rangle, \ldots, \langle w_{x_n}, w_{y_n}, \ldots, w_{m_n}\rangle]))\end{aligned}$$

Wegner (2000), em seus trabalhos, define uma PeTM como sendo uma secretária eletrônica. A partir desse exemplo, propõe-se uma secretária eletrônica com bina modelada por uma PeTM-Par.

EXEMPLO 5.1: Uma Secretária Eletrônica (SE) com bina é uma PeTM-Par, cuja fita contém uma seqüência de mensagens gravadas e cujas operações são: `gravar`, `mostrar` e `apagar`. A função Turing-computável para máquina SE é:

$$f_{SE}(\texttt{gravar}(\langle nu, msg\rangle) \quad ([\langle \texttt{gravar}_1(nu), \texttt{gravar}_2(msg)\rangle], [\langle w_{nu}, w_{msg}\rangle]) = ([ok], [\langle nu, msg\rangle]));$$

$$f_{SE}(\texttt{apagar}([\langle nu, msg\rangle], [\langle w_{nu}, w_{msg}\rangle]) = ([ok], [\langle \varepsilon, \varepsilon\rangle]));$$

Para exibir a saída corretamente da função `mostrar`, utiliza-se uma função f_I. Essa função intercala os elementos de uma fita e de outra.

$$f_I(w_{nu}, w_{msg}) = \mathit{append}(\mathit{head}(w_{nu}), \mathit{append}(\mathit{head}(w_{msg}), f_I(\mathit{tail}(w_{nu}), \mathit{tail}(w_{msg}))))$$

$$f_{SE}(\texttt{mostrar}\ ([\langle nu, msg\rangle], [\langle w_{nu}, w_{msg}\rangle]) = ([f_I(w_{nu}, w_{msg})], [\langle w_{nu}, w_{msg}\rangle]));$$

A computação da máquina SE para a seqüência de entrada (`gravar`($\langle 123, abc\rangle$), `mostrar`, `apagar`, `gravar`($\langle 123, abc\rangle$), `gravar`($\langle 456, def\rangle$), `mostrar`, ...), que gera a seqüência de saída (ok, $[123abc]$, ok, ok, ok, $[123abc456def]$, ...); o estado global da máquina evolui como segue: ($\langle\langle\varepsilon\rangle, \langle\varepsilon\rangle\rangle$, $\langle\langle 123\rangle, \langle abc\rangle\rangle$, $\langle\langle 123\rangle, \langle abc\rangle\rangle$, $\langle\langle\varepsilon\rangle, \langle\varepsilon\rangle\rangle$, $\langle\langle 123\rangle, \langle abc\rangle\rangle$, $\langle\langle 123, 456\rangle$, $\langle abc, def\rangle\rangle$, $\langle\langle 123, 456\rangle$, $\langle abc, def\rangle\rangle$, ...)[1].

5.2 Modelagens de Sistemas na PeTM-Par

Para ilustrar melhor as funcionalidades definidas para a PeTM-Par, esta seção apresenta a modelagem de outros sistemas, que exploram as diferentes operações definidas para a máquina. Nessas modelagens foram escolhidos sistemas que, assim como a secretária eletrônica, apresentam características de interatividade com o ambiente.

5.2.1 Sistema de Computação Par-a-Par

Um tipo de sistema de computação que vêm se popularizando consideravelmente nos últimos anos são os sistemas par-a-par (*peer-to-peer*). Uma das primeiras iniciativas nesse sentido foi o projeto SETI@Home[2], que chamou a atenção da comunidade da Internet demonstrando que é possível obter um alto poder de processamento pela união de computadores pessoais conectados através da Internet. Mas foi a partir do *Napster*[3] que os sistemas par-a-par começaram a ser difundidos. O Napster, e outros sistemas par-a-par como Gnutella[4] e Freenet[5], tornaram-se extremamente populares por permitirem e facilitarem o compartilhamento de arquivos diretamente entre usuários conectados ao sistema. Um

[1] Para clareza de apresentação da evolução da computação da máquina, o conteúdo de cada fita é também mostrado como uma lista de palavras, permitindo a distinção das mesmas. A mesma abordagem será utilizada ao longo deste capítulo

[2] http://setiathome.ssl.berkeley.edu

[3] http://www.napster.com

[4] http://www.gnutella.com

[5] http://freenet.sourceforge.net

sistema par-a-par em funcionamento comporta-se como um grande repositório de dados, onde operações para procura, disponibilização e *download* de arquivos são oferecidas. Ao mesmo tempo, os arquivos estão permanentemente vinculados aos usuários; uma vez que um usuário desconecta-se do sistema, os arquivos que ele disponibiliza tornam-se inacessíveis. Isso confere a um sistema par-a-par uma característica bastante dinâmica e interativa.

Motivado por essa última consideração, apresenta-se nesta Seção a modelagem pela PeTM-Par de um sistema par-a-par fictício.

EXEMPLO 5.2: Seja N um sistema par-a-par que oferece as seguintes operações básicas: `conexao`, `procura`, `lista` e `adiciona`. Considera-se que o sistema já possui outros usuários conectados com os respectivos arquivos, sendo o conteúdo inicial das fitas: $w_{ip} = \langle 1.5, 4.6, 7.3 \rangle$ e $w_{arq} = \langle bb, aa, hh \rangle$.

Sendo *select* uma função que compara os dois primeiros elementos, se forem iguais, retorna o terceiro elemento, senão retorna ε. *select* é definida conforme:

$$\mathit{select}(x, a, b) = \begin{cases} b, & \text{se } a = x \\ \varepsilon, & \text{se } a \neq x \end{cases}$$

A função f_L lista os elementos da fita w_2, cujo correspondente na fita w_1 é igual x

$$f_L(x, w_2, w_1) = \mathit{append}(\mathit{select}(x, \mathit{head}(w_1), \mathit{head}(w_2)), f_L(x, \mathit{tail}(w_1), \mathit{tail}(w_2)))$$

A função Turing-computável para N é:

$$f_N(\texttt{procura}(\langle \varepsilon, arq \rangle) \quad ([\langle \texttt{procura}_1(ip), \texttt{procura}_2(arq) \rangle], [\langle w_{ip}, w_{arq} \rangle]) = ([f_L(arq, w_{arq}, w_{ip})], [\langle w_{ip}, w_{arq} \rangle]))$$

$$f_N(\texttt{adiciona}(\langle ip, arq \rangle) \quad ([\langle \texttt{adiciona}_1(ip), \texttt{adiciona}_2(arq) \rangle], [\langle w_{ip}, w_{arq} \rangle]) = ([ok], [\langle w_{ip} ip, w_{arq} arq \rangle]))$$

$$f_N(\texttt{lista}(\langle ip, \varepsilon \rangle) \quad ([\langle \texttt{lista}_1(ip), \texttt{lista}_2(arq) \rangle], [\langle w_{ip}, w_{arq} \rangle]) = ([f_L(ip, w_{ip}, w_{arq})], [\langle w_{ip}, w_{arq} \rangle]))$$

$$f_N(\texttt{conexao}(\langle ip \rangle, \langle arq \rangle) \quad [\texttt{conectar}, \texttt{disp}], [\langle ip \rangle, \langle arq \rangle], [\langle w_{ip} \rangle, \langle w_{arq} \rangle]) = ([ok], [\langle w_{ip} ip \rangle, \langle w_{arq} arq \rangle]);$$

A computação do sistema N para a seqüência de entrada (`conexão`$(\langle 1.2, xx \rangle)$, `adiciona`$(\langle 1.2, yy \rangle)$, `lista`$(\langle 1.5, \varepsilon \rangle$, `procura`$(\langle \varepsilon, aa \rangle)$, `adiciona`$(\langle 1.2, rr \rangle)$, `lista`$(\langle 1.2, \varepsilon \rangle, \ldots)$, que gera a seqüência de saída (ok, ok, $[aa]$, $[1.5]$, ok $[xxyyrr]$, $\ldots$); o estado global do sistema evolui como segue: ($\langle\langle 1.5, 4.6, 7.3 \rangle, \langle bb, aa, hh \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2 \rangle, \langle bb, aa, hh, xx \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2, 1.2 \rangle, \langle bb, aa, hh, xx, yy \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2, 1.2 \rangle, \langle bb, aa, hh, xx, yy \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2, 1.2 \rangle, \langle bb, aa, hh, xx, yy \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2, 1.2, 1.2 \rangle, \langle bb, aa, hh, xx, yy, rr \rangle\rangle$, $\langle\langle 1.5, 4.6, 7.3, 1.2, 1.2, 1.2 \rangle, \langle bb, aa, hh, xx, yy, rr \rangle\rangle$).

5.2.2 Conta bancária

Outro sistema interativo que pode ser modelado, no qual podem ser demonstradas operações realizadas em paralelo é o sistema de manutenção de conta corrente em um banco. Esta Seção mostra a modelagem das operações `transf`, `saldo`, `deposito` e `saque`, sendo executadas por um cliente (ambiente) em sua conta bancária (máquina).

EXEMPLO 5.3: Um banco qualquer B disponibiliza operações bancárias para seus clientes num caixa automático. Os clientes possuem conta corrente (cc) com uma poupança (p) vinculada. As operações disponíveis são `transf`, `depósito`, `saque` e `saldo`. A função Turing-computável para B é:

$$f_B(\texttt{saldo}([\langle c, p\rangle], [\langle w_c, w_p\rangle]) = ([\langle w_c, w_p\rangle], [\langle w_c, w_p\rangle]));$$

A operação `depósito` utiliza duas funções auxiliares, *soma* e *diferença*. A função *soma* recebe dois elementos e retorna um terceiro que representa a soma dos dois. A função *diferença* recebe dois elementos e retorna um terceiro que representa a diferença dos dois elementos. Essas funções são definidas como:

$$\mathit{soma}(x, y) = z \quad \mathit{diferença}(a, b) = c$$

$$\begin{aligned} f_B(\texttt{deposito}(\langle c, p\rangle) \quad & (\\ & [\langle \texttt{deposito}_1(c), \texttt{deposito}_2(p)\rangle], [\langle w_c, w_p\rangle]) = \\ & ([ok], [\langle \mathit{soma}(w_c, c), \mathit{soma}(w_p, p)\rangle])); \end{aligned}$$

$$\begin{aligned} f_B(\texttt{saque}(\langle c, \varepsilon\rangle) \quad & (\\ & [\langle \texttt{saque}_1(c), \varepsilon\rangle], [\langle w_c, w_p\rangle]) = \\ & ([ok], [\langle \mathit{diferença}(w_c, c), w_p\rangle])); \end{aligned}$$

$$\begin{aligned} f_B(\texttt{transf}(\langle c, p\rangle) \quad & (\\ & [\langle \texttt{transf}_1(c), \texttt{transf}_2(p)\rangle], [\langle w_c, w_p\rangle]) = \\ & ([ok], [\langle \mathit{soma}(w_c, c), \mathit{diferença}(w_p, p)\rangle]))); \end{aligned}$$

A computação da máquina B para a seqüência de entrada (`deposito` $(\langle 10, 20\rangle)$, `saldo`, `tranferencia`$(\langle 10, 10\rangle)$, `saque`$(\langle 5, \varepsilon\rangle)$, `saldo`, ...), que gera a seqüência de saída (ok, $[10, 20]$, ok, ok, $[15, 10]$); o estado global da máquina evolui como segue: $(\langle\langle\varepsilon\rangle, \langle\varepsilon\rangle\rangle$, $\langle\langle 10\rangle, \langle 20\rangle\rangle$, $\langle\langle 10\rangle, \langle 20\rangle\rangle$, $\langle\langle 20\rangle, \langle 10\rangle\rangle$, $\langle\langle 15\rangle, \langle 10\rangle\rangle$, $\langle\langle 15\rangle, \langle 10\rangle\rangle$, ...).

5.3 Modelos de Comportamento da PeTM-Par

Para qualquer PeTM-Par M, será definido três diferentes modelos de comportamento. Baseados na visão dada por Wegner, pode-se atribuir a PeTM-Par os seguintes modelos de comportamento: baseado em linguagem, baseado em funções e baseado em ambiente. A definição de cada comportamento está na

Seção 4.3. Na seqüência do trabalho, propõem-se essas definições de comportamento.

Um passo computacional da PeTM-Par M é modelado por uma função f_M, onde E, S e $\langle w \rangle$ são definidos sobre o alfabeto Σ^*.

$$f_M : (E \times \langle w_1, w_2, \ldots, w_n \rangle) \rightarrow (S \times \langle w_1, w_2, \ldots, w_n \rangle)$$

5.4 Modelo baseado em linguagem da computação da PeTM-Par

De acordo com a definição 4.4, a linguagem da PeTM-Par pode ser apresentada da mesma forma que a linguagem da PeTM. Isto porque a definição de linguagem de ambas as máquinas, está baseada na seqüência de interação.

5.4.1 Linguagem na TM

A definição de linguagem de uma Máquina de Turing M, denotada por $L(M)$, é o conjunto de todas as palavras pertencentes a um dado alfabeto Σ^* aceitas por M (MENEZES, 2001).

5.4.2 Equivalência e Expressividade na PeTM-Par

A definição de linguagem da PeTM pode ser estendida a PeTM-Par, assim sendo, a definição de equivalência e expressividade dada na Seção 4.5, pode ser aplicada diretamente para mostrar como fica o comportamento da PeTM-Par baseada em linguagem.

Similarmente, para as variações apresentadas para PeTM com amnésia, PeTM com memória finita e PeTM com estado finito, as mesmas definições e provas podem ser aproveitadas no caso da PeTM-Par. Por exemplo, no caso da PeTM-Par com amnésia, bastaria apagar o conteúdo das fitas persistentes ao final de cada passo computacional, como evidenciado para a PeTM. Nesse caso, a PeTM-Par apresenta a mesma expressividade que seqüências de computações da TM (conforme Teorema 4.1).

5.5 Modelo da PeTM-Par baseado em funções

Essa seção apresenta o comportamento da PeTM-Par modelado por função definida recursivamente. O comportamento da PeTM-Par está baseado no comportamento da PeTM, diferenciando-se para dar tratamento adequado às n fitas. O mapeamento Ξ para uma PeTM-Par é definido a seguir.

5.5.1 Mapeamento Ξ

O mapeamento Ξ emprega funções, que estão definidas ao longo da seção. Primeiramente é necessário definir duas funções sobre pares, a *1st* e *2nd*, retornado o primeiro e segundo componente de cada par, respectivamente. Conforme Goldin (2000), define-se ainda três funções sobre seqüências, *head*, *tail* e *append*; A função *head*(σ) retorna o primeiro símbolo de uma seqüência σ, a função *tail*(σ) retorna o resto da seqüência, e a função *append*(s, σ) concatena um símbolo s para o início da seqüência σ, retornando uma nova seqüência.

Vale ressaltar que a PeTM-Par comporta-se de acordo com a PeTM, mas le-

vando em consideração as n fitas. Então observações sobre as seqüências de entrada, estão esclarecidas na Seção 4.6.

Seja uma PeTM M onde f_M é a correspondente função computável:

$$f_M : (E, \langle w_1, w_2, \ldots, w_n \rangle) = (S, \langle w'_1, w'_2, \ldots, w'_n \rangle)$$

Se ι é seqüência de entrada e $\langle w_1, w_2, \ldots, w_n \rangle$ são os estados correntes de M, o primeiro passo computacional de M produzirá $f_M(\mathit{head}(\iota), \langle w_1, w_2, \ldots, w_n \rangle)$. Seja s o primeiro símbolo a sair e $\langle w'_1, w'_2, \ldots, w'_n \rangle$ são os novos estados de M, podendo ser visto como um estado global de M, sendo definido conforme:

$$s = \mathit{1st}(f_M(\mathit{head}(\iota), \langle w_1, w_2, \ldots, w_n \rangle))$$

$$\langle w'_1, w'_2, \ldots, w'_n \rangle = \mathit{2nd}(f_M(\mathit{head}(\iota), \langle w_1, w_2, \ldots, w_n \rangle))$$

Então pode-se definir a função $frec_M$ que recebe uma seqüência de entrada ι e os estados $\langle w_1, w_2, \ldots, w_n \rangle$ e retorna uma seqüência de saída; $frec_M$ é definida recursivamente conforme:

$$frec_M(\iota, \langle w \rangle) = \mathit{append}(s, frec_M(\mathit{tail}(\iota), \langle w'_1, w'_2, \ldots, w'_n \rangle))$$

Embora $frec_M$ retorne com sucesso a seqüência de saída, que recebe os estados como uma das entradas, que não é o que se quer. O mapeamento desejado, Ξ_M, é obtido por pegar os estados iniciais $\langle w_1, w_2, \ldots, w_n \rangle$ de M em $frec_M$:

$$\forall x, \Xi_M(x) = frec_M(x, \langle w_1, w_2, \ldots, w_n \rangle)$$

Teorema 5.1 *Para quaisquer PeTM-Pars M_1 e M_2, $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ se e somente se $\Xi_{M_1} = \Xi_{M_2}$*

5.6 Equivalência da PeTM-Par e PeTM baseadas em função

Uma PeTM é equivalente a uma PeTM-Par conforme a prova demonstrada a seguir. A prova consiste em mostrar que, a partir de uma PeTM-Par M qualquer, é possível construir uma PeTM M' que realiza o mesmo processamento, isto é, M' simula M. A demonstração apresenta a seqüência de passos para converter uma PeTM-Par qualquer em uma PeTM equivalente. A idéia central da construção é a concatenação das fitas de M de modo que se transforme em uma única fita em M'.

São usadas as funções *head*, *tail* e *cons* apresentadas por Hein (2002). A função *head* retorna o primeiro elemento de uma lista e a função *tail* retorna o restante da lista. A função *cons* constrói uma lista, que tem como entrada um elemento e uma lista, retornando uma nova lista. Exemplo: Seja w uma lista, $w = \langle a, b, \ldots, z \rangle$, então $\mathit{head}(\langle a, b, \ldots, z \rangle) = a$, $\mathit{tail}(\langle a, b, \ldots, z \rangle) = \langle b, \ldots, z \rangle$ e $\mathit{cons}(a, \langle b, \ldots, z \rangle) = \langle a, b, \ldots, z \rangle$

Nesta prova, considera-se uma fita de trabalho da PeTM-Par como sendo uma lista de palavras, $w_1 = \langle a, b, \ldots, z \rangle$, onde utiliza-se das funções *head* e *tail*.

Prova: Seja M uma PeTM-Par qualquer. Seja M' uma PeTM construída a partir de M como segue:

alfabeto Seja Σ o alfabeto de símbolos da máquina M. Seja Σ' o alfabeto da máquina M', então $\Sigma' = \Sigma \cup \{\sharp\} / \sharp \notin \Sigma$, onde $\sharp$ é o símbolo de concatenação.

conc A função *conc* recebe uma lista de palavras e retorna uma nova lista que representa a concatenação das palavras da lista intercaladas por $\sharp$. Exemplo: $l = \langle a, b, c \rangle$; $conc(l) = a\sharp b\sharp c\sharp$

f_{pos} **(Concatenação)** Esta função recebe listas de palavras e produz uma única lista cujas palavras são formadas pela concatenação das listas de palavras anteriores.

$$\begin{aligned} f_{pos}(\langle w \rangle) &= cons(conc(\langle head(w_1), head(w_2), \ldots, head(w_n) \rangle), \\ &\quad f_{pos}(tail(w_1), tail(w_2), \ldots, tail(w_n))) \end{aligned}$$

Ex: $f_{pos}(\langle abc, def, ghi \rangle, \langle jkl, mno, pqr \rangle) = (\langle abc\sharp jkl\sharp, def\sharp mno\sharp, ghi\sharp pqr\sharp \rangle)$

f_{pre} **(Separação)** Esta função recebe uma fita com palavras formadas pela concatenação $\sharp$ (oriundas da função f_{pos}) e retorna uma lista de n fitas; ou seja, realiza o processo inverso ao da função f_{pos}.

$$l_{fita} = \langle l_1, l_2, \ldots, l_n \rangle$$

A função *fita* retorna o primeiro elemento do par, que é virá a ser uma das fitas de trabalho da PeTM-Par.

$$\begin{aligned} f_{pre}(l) &= fita((\langle head(l_1), head(l_2), \ldots, head(l_n) \rangle), \\ &\quad f_{pre}(\langle tail(l_1), tail(l_2), \ldots, tail(l_n) \rangle)) \end{aligned}$$

A cada aplicação da f_{pre} em l, $fita$ retorna o primeiro elemento do par, sendo uma lista de palavras, que são as palavras de cada fita. Sucessivas aplicações da f_{pre}, faz com que *fita* retorne as outras fitas de trabalho da máquina PeTM-Par.

Ex: $f_{pre}(\langle abc\sharp jkl\sharp, def\sharp mno\sharp, ghi\sharp pqr\sharp \rangle) = (\langle abc, def, ghi \rangle, \langle jkl, mno, pqr \rangle)$

Dadas as funções da PeTM e da PeTM-Par, respectivamente:

$$f_{PeTM} : (E \times W_{PeTM}) = (S \times W'_{PeTM})$$

$$f_{PeTM-Par} = (E \times W_{PeTM-Par}) = (S \times W'_{PeTM-Par})$$

As relações entre as duas, que permitem a construção da PeTM a partir da PeTM-Par, são dadas por:

$$W'_{PeTM} = f_{pos}(W'_{PeTM-Par})$$

$$W_{PeTM-Par} = f_{pre}(W_{PeTM})$$

Onde:

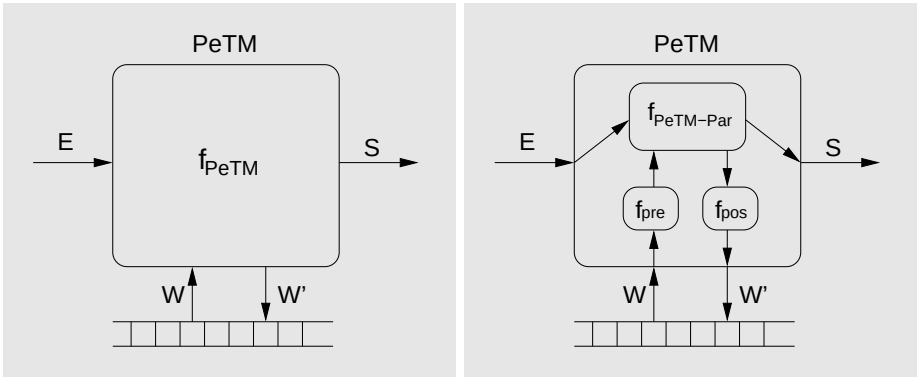

Figura 5.3: Construção da PeTM a partir da PeTM-Par

E	seqüência de entrada;
S	seqüência de saída;
W_{PeTM}	fita de trabalho da PeTM;
$W_{PeTM-Par}$	fitas de trabalho da PeTM-Par, representadas por $\langle w_1, w_2, \ldots, w_n \rangle$;

Concluindo, a transformação da PeTM-Par em PeTM utiliza a função $f_{PeTM-Par}$ já definida, mas incrementada por duas funções, f_{pre} e f_{pos}, que podem ser vistas como duas etapas a mais em cada passo computacional. Para cada passo computacional toma-se o conteúdo da fita de trabalho W da PeTM, transforma-se esse conteúdo em múltiplas fitas para poder ser aplicada a função da PeTM-Par, e torna-se a transformar o conteúdo das múltiplas fitas de trabalho em uma única palavra (W').

Assim, uma PeTM simula uma PeTM-Par para qualquer seqüência de entrada $e \in \Sigma^*$. A construção de uma PeTM a partir de uma PeTM-Par não precisa ser mostrada, pois decorre das definições anteriores. Se a PeTM-Par tiver apenas uma fita de trabalho, deve comportar-se como uma PeTM. Conseqüentemente, sabe-se que as máquinas tem a mesma expressividade. A Figura 5.3 ilustra o mecanismo utilizado na prova. A função da PeTM é construída a partir da função original da PeTM-Par com o auxílio das duas funções f_{pre} e f_{pos}.

EXEMPLO 5.4: Esse exemplo demonstra a contrução de uma PeTM a partir de uma PeTM-Par usando os passos descritos acima. Seja a PeTM-Par M do exemplo 5.1.3 e a PeTM M' do exemplo 4.2.

O primeiro passo é gerar o alfabeto Σ' da PeTM, que é $\Sigma' = \Sigma \cup \{\sharp\}$, sendo $\Sigma = \{a, b, \ldots, z, 0, 1, \ldots, 9\}$

De acordo com a aplicação das funções acima, a evolução da computação para uma dada seqüência de entrada (`gravar`$(\langle 123, abc \rangle)$, `mostrar`, `apagar`, `gravar`$(\langle 123, abc \rangle)$, `gravar`$(\langle 456, def \rangle)$, `mostrar`, ...), vai gerar a seqüência de saída (ok, $[123abc]$, ok, ok, ok, $[123abc456def]$, ...); o estado da máquina PeTM evolui como segue: ($\sharp\sharp$, $123\sharp abc\sharp$, $123\sharp abc\sharp$, $\sharp\sharp$, $123\sharp abc\sharp$, $123\sharp abc\sharp 456\sharp def\sharp$, $123\sharp abc\sharp 456\sharp def\sharp$, ...).

Para ilustrar mais claramente o processo de conversão das fitas de trabalho da PeTM e PeTM-Par, a evolução dos estados em ambas as máquinas é mostrada na

Tabela 5.1: Relação entre a evolução de estados da PeTM-Par e da PeTM

PeTM-Par		**PeTM**
ε	ε	$\sharp\sharp$
123	abc	$123\sharp abc\sharp$
123	abc	$123\sharp abc\sharp$
ε	ε	$\sharp\sharp$
123	abc	$123\sharp abc\sharp$
123, 456	abc, def	$123\sharp abc\sharp, 456\sharp def\sharp$

Tabela 5.1.

5.7 Modelo da PeTM-Par baseado em Ambiente

Na modelagem baseada em ambiente da PeTM-Par, podem ser aplicadas diretamente as definições e conseqüentes proposições e provas já apresentadas na Seção 4.7.1 para a PeTM. Isto porque as mesmas são todas definidas a partir de uma noção de observação, estabelecida na definição 4.13, conforme detalhado a seguir.

A noção de observação estabelece que qualquer prefixo finito de uma seqüência de interação de uma PeTM constitui uma observação da mesma. Como conseqüência, define-se a noção de ambiente finito da PeTM como sendo o mapeamento das PeTMs em conjuntos de observações das mesmas.

Conforme a definição apresentada da PeTM-Par, pode-se perceber que as definições relativas às seqüências de entradas e saídas permanecem inalteradas em relação àquelas da PeTM. Portanto, sendo as noções de observação e ambiente obtidas a partir de considerações sobre a seqüência de interação, conclui-se que as mesmas definições podem ser aplicadas diretamente à PeTM-Par.

Vale ressaltar que a definição de Certificado de distinguibilidade é também aplicável à PeTM-Par. De acordo com a Definição 4.14, o certificado de distinguibilidade é uma observação do comportamento de uma máquina M_1 em um ambiente O que não é observável em uma máquina M_2 no mesmo ambiente O. Portanto, dadas duas PeTM-Pais não equivalentes, é possível encontrar um certificado de distinguibilidade de tamanho finito.

A hierarquia de expressividade infinita também pode ser incluída. Aplicando-se a Definição 4.15 para PeTM-Par, chega-se a mesma conclusão. A Proposição 4.8 mostra que os ambientes em Θ são cada vez mais ricos, produzindo classes de equivalência cada vez mais finas para comportamentos da PeTM-Par sem atingir um limite. Essa inclusão é possível porque já foi mostrado acima que a PeTM-Par e a PeTM comportam-se da mesma forma neste ambiente.

5.8 Considerações para a PeTM-Par

Por estarem fora do escopo deste trabalho, as relações e equivalências das SIMs e TMs com a PeTM-Par não foram contempladas. Entretanto, é possível examinar alguns resultados e comparações em outros trabalhos. A dissertação

limitou-se a apresentar a equivalência da PeTM-Par e PeTM, não percorrendo outros modelos.

Todas as considerações expostas na Seção 4.8, servem para a PeTM-Par. Essa afirmação é possível porque neste capítulo foi demonstrada a equivalência da PeTM-Par e PeTM, nos três comportamentos apresentados. Assim sendo, cada comparação e relação obtida é passível de ser adaptada para a PeTM-Par.

6 CONCLUSÃO

A mudança de sistemas fechados para sistemas interativos é uma conseqüência de uma mudança no paradigma de algoritmos para interação. Na área de Teoria da Computação Interativa, devido à necessidade de capturar vários aspectos do sistema, como por exemplo interação, é fundamental elaborar novos formalismos ou detalhar os existentes para tratar tais aspectos. Como já mencionado na Introdução, a Ciência da Computação está em constante expansão nos mais variados aspectos, acompanhando a evolução tecnológica, e portanto, freqüentemente torna-se necessário o desenvolvimento de novos formalismos, que são importantes no auxílio à modelagem de sistemas.

Nesta Dissertação, buscou-se apresentar uma contribuição à área da Teoria da Computação Interativa no âmbito de modelos de computação baseados na Máquina de Turing. Como forma de continuidade ao trabalho desenvolvido principalmente por Peter Wegner e Dina Goldin, apresentou-se uma extensão para a Máquina de Turing Persistente, proposta por esses pesquisadores, pela adição de múltiplas fitas persistentes de trabalho, em contraste com a existência de uma única fita persistente no modelo por eles originalmente proposto.

A primeira parte deste trabalho, Capítulo 2, teve como objetivo contextualizar conceitos e definições clássicas da Teoria da Computação, como Máquina de Turing, Algoritmo, Máquina de Turing Paralela, entre outros. No Capítulo 3, explorou-se os conceitos da Teoria da Computação Interativa, sendo extremamente importante para analisar as relações existentes com a Teoria da Computação Clássica, e ao mesmo tempo servindo de base para a apresentação do restante do trabalho.

É importante salientar que as duas Teorias foram apresentadas em seus aspectos fundamentais e quando pertinente foi feita relação entre conceitos comuns ou equivalentes nas duas. Entretanto, essa relação não tem caráter de comparação; as duas tratam dos mesmos conceitos, porém em contextos diferentes. Dessa forma, procurou-se evidenciar os pontos importantes e necessários para compreensão do trabalho aqui desenvolvido.

No Capítulo 4, é descrita a Máquina de Turing Persistente, a qual serviu como inspiração para a construção da Máquina de Turing Persistente Paralela, que por sua vez é definida no Capítulo 5. Através de construções e demonstrações, foram expostas a definição da PeTM-Par e exemplos de aplicações modeladas segundo essa definição. Igualmente, foi apresentada formalmente a verificação de equivalência de expressividade entre os dois modelos (PeTM e PeTM-Par), o que demonstra que a adição de fitas persistentes não aumenta a expressividade da Máquina de Turing Persistente.

6.1 Resultados Obtidos e Contribuições

Como resultado, neste trabalho pode-se observar que nem sempre o fato de acrescentar-se recursos a uma máquina implica no aumento de sua expressividade. O que foi demonstrado no Capítulo 5.

Acredita-se que duas das principais contribuições do trabalho são $i)$ a extensão de um modelo para exploração do paralelismo interno em uma PeTM (resultando na PeTM-Par), o qual permite aplicação direta de exemplos, conforme mostrado no Capítulo 5, e assim disponibiliza alternativas para o estudo do processamento paralelo e seus modelos formais, e $ii)$ a demonstração de equivalência entre a PeTM e a PeTM-Par, o que igualmente permite a aplicação à segunda de diversas definições e teoremas já desenvolvidos para a primeira, facilitando seu estudo e desenvolvimentos futuros.

Outras contribuições podem ainda ser identificadas; em geral, os métodos e ferramentas para formalização da computação na PeTM-Par desenvolvidos nesse trabalho podem servir como uma base para a compreensão da Teoria da Computação Interativa e como o paralelismo pode ser especificado em modelos teóricos.

6.2 Trabalhos futuros

Como atividades futuras, pode-se vislumbrar, desde já, ao menos duas linhas de pesquisa:

- a exploração de múltiplas fitas de trabalho persistentes em um modelo multi-seqüencial: se pensar em uma PeTM, como um modelo estendido para contemplar múltiplas seqüências de entrada, caracterizará o que se denomina uma máquina multi-seqüencial (MIM), que segundo Wegner, apresenta maior expressividade do que a SIM. Uma continuação natural do trabalho desenvolvido nesta Dissertação é a verificação da expressividade da PeTM-Par quando inserida no modelo multi-seqüencial. Efetivamente, o exemplo dado na seção 5.2.1, que ilustra um sistema par-a-par, quando analisado com profundidade, reflete um sistema com característica multi-seqüencial, pois a existência de vários usuários acessando o sistema caracteriza exatamente a noção de múltiplas seqüencias de entrada. Portanto, acredita-se que o assunto possa ser explorado mais profundamente em outro trabalho;

- durante o desenvolvimento da Dissertação, realizou-se um breve contato, por email, com Dina Goldin, a fim de obter sua avaliação sobre o trabalho sendo realizado. A pesquisadora, efetivamente, mostrou interesse no desenvolvimento do trabalho, e adicionalmente colocou que uma linha de pesquisa que certamente despertaria interesse no contexto da TCI é a de modelos distribuídos. Ora, o trabalho aqui apresentado explora o paralelismo interno em uma PeTM, e portanto acredita-se que uma atividade complementar seria certamente a exploração de distribuição ao processamento interno da máquina. Explorar o conceito de distribuído, é verificar como várias PeTMs interagem, estando geograficamente ou logicamente separadas.

REFERÊNCIAS

ACIÓLY, B. M.; BEDREGAL, B. R. C.; LYRA, A. **Introdução à Teoria da Computação**: linguagens formais e computabilidade. Natal: Ed. UnP, 2000.

AHO, A. V.; HOPCROFT, J. E.; ULLMAN, J. D. **The Design and Analysis od Computer Algorithms**. Massachusetts: Addison Wesley, 1974.

ANDREWS, G. R. **Concurrent Programming**: principles and practice. Redwood City: Benjamin/Cummings, 1991. 637p.

BIRD, R. S. **Programs and Machines**: an introduction to the theory of computation. London: John Wiley, 1976. 214p.

DIVERIO, T.; MENEZES, P. **Teoria da Computação**: máquinas universais e computabilidade. 2.ed. Porto Alegre: Sagra-Luzzatto, 2000. (Livros Didáticos, v.5).

FERBER, J. **Multi-Agent Systems**: an introduction to distributed artificial intelligence. Harlow: Addison-Wesley, 1999.

GOLDIN, D.; KEIL, D. Interaction, Evolution, and Intelligence. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2001, Seoul, Korea. **Proceedings...** [S.l.: s.n.], 2001.

GOLDIN, D.; KEIL, D.; WEGNER, P. **A Historical Perspective of Interactive Computing**. Disponível em: <http//www.cs.brown.edu/people/pw>. Acesso em: mar. 2002.

GOLDIN, D. Q. Persistent Turing Machines as a Model of Interactive Computation. In: FOUNDATIONS OF INFORMATION AND KNOWLEDGE SYSTEMS, 1., 2000, Burg, Germany. **Proceedings...** Berlin: Springer, 2000. p.116–135. (Lecture Notes in Computer Science, v.1762).

GOLDIN, D.; WEGNER, P. **Persistence as a Form of Interaction**. Brown: Brown University, 1998. (CS 98-07).

GOLDIN, D.; WEGNER, P. **Behavior and Expressiveness of Persistent Turing Machine**. Brown: Brown University, 2000. (CS-99-14).

HEIN, J. L. **Discrete Structures, Logic, and Computability**. 2nd.ed. Boston: Jones and Bartlett, 2002. 943p.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to Automata Theory, Languages, and Computation**. 2nd.ed. Boston: Addison-Wesley, 2001. 521p.

HWANG, K.; BRIGGS, F. A. **Computer Architecture and Parallel Processing**. New York: McGraw-Hill, 1984. 846p.

JACOBS, B.; RUTTEN, J. A Tutorial on (Co)Algebras and (Co)Induction. **Bulletin of the European Association for Theoretical Computer Science**, [S.l.], v.62, June 1997.

JOYCE, D. E. **Hilbert's Mathematical Problems**. Disponível em: <http://aleph0.clarku.edu/~djoyce/hilbert/toc.html>. Acesso em: out. 2001.

KLEENE, S. C. **Mathematical Logic**. New York: John Wiley, 1967. 398p.

LEEUWEN, J. van; WIEDERMANN, J. On Algorithms and Interaction. In: INTERNATIONAL SYMPOSIUM MATHEMATICAL FOUNDATIONS OF COMPUTER SCIENCE, 25., 2000. **Proceedings...** Berlin: Springer, 2000. p.99–112. (Lecture Notes in Computer Science, v.1893).

LEEUWEN, J. van; WIEDERMANN, J. On the power of interactive computing. In: IFIP THEORETICAL COMPUTER SCIENCE, 2000, Sendai, Japan. **Proceedings...** Berlin: Springer, 2000. p.619–623. (Lecture Notes in Computer Science, v.1872).

LEEUWEN, J. van; WIEDERMANN, J. The Turing Machine Paradigm in Contemporary Computing. In: ENGQUIST, B.; SCHMID, W. (Ed.). **Mathematics Unlimited — 2001 and Beyond**. [S.l.]: Springer-Verlag, 2001. p.1139–1155.

LEWIS, H.; PAPADIMITRIOU, C. **Elements of the Theory of Computation**. 2nd.ed. Upper Saddle River: Prentice Hall, 1998. 361p.

MANNA, Z. **Mathematical Theory of Computation**. Rehovot, Israel: McGraw-Hill, 1974. (McGraw-Hill computer science).

MENEZES, P. B. **Linguagens Formais e Autômatos**. 4.ed. Porto Alegre: Sagra-Luzzatto, 2001. (Livros Didáticos, v.3).

MILNER, R. **Communication and Concurrency**. Englewood Cliffs: Prentice Hall, 1989. 260p.

MINSKY, M. L. **Computation**: finite and infinite machines. Englewood Cliffs: Prentice Hall, 1967.

PRASSE, M.; RITTGEN, P. Why Church's Thesis Still Holds — Some Notes on Peter Wegner's Tracts on Interaction and Computability. **Computer Journal**, [S.l.], v.41, n.6, p.357 – 362, 1998.

PY, M. X.; TOSCANI, L. V.; LAMB, L. C.; DIVERIO, T. A. Learning Parallel Computing Concepts via a TuringMachine Simulator. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 13., 2001, Pirenópolis, Goiás. **Proceedings...** Brasília: Departamento de Ciência da Computação da UNB, 2001. p.134–139.

REISIG, W. **Petri Nets**: an introduction. New York: Springer-Verlag, 1985.

SCHÖNING, U.; PRUIM, R. Hilbert's Tenth Problem. In: **Gems of Theoretical Computer Science**. Berlin, Germany: Springer-Verlag, 1998. p.15—24.

SCOTT, D. Continuous Lattices. In: LAWVERE, F. W. (Ed.). **Toposes, Algebraic Geometry and Logic**. Berlin: Springer-Verlag, 1972. v.274, p.97–136.

SIPSER, M. **Introduction to the Theory of Computation**. Boston: PWS, 1997. 396p.

STERLING, T. L. **Beowulf Cluster Computing with Linux**. Cambridge: MIT Press, 2002.

TOSCANI, L.; VELOSO, P. **Complexidade de Algoritmos**: análise, projetos e métodos. Porto Alegre: Sagra-Luzzatto, 2001. (Livros Didáticos, v.13).

TURING, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. **Proceedings of the London Mathematical Society**, [S.l.], v.42, n.2, p.230–265, 1936. Corrigido no v.43, p.544–546.

VAN EMDE BOAS, P. Machine Models and Simulations. In: LEEUWEN, J. van (Ed.). **Handbook of Theoretical Computer Science**. Amsterdam: Elsevier Science, 1990. v.A, p.1–66.

WEGNER, P. Why Interaction is More Powerful than Algorithms. **Communications of the ACM**, New York, v.40, n.5, p.80–91, May 1997.

WEGNER, P. Interactive Software Technology. In: TUCKER JR., A. B. (Ed.). **The Computer Science and Engineering Handbook**. Boca Raton: CRC Press, 1997.

WEGNER, P. Interactive Foundations of Computing. **Theoretical Computer Science**, Amsterdam, v.192, n.2, p.315–351, Feb. 1998.

WEGNER, P. **Peter Wegner's Banquet Speech**. Palestra no 13º European Conference on Objetct-Oriented Programming, 14–18 de junho de 1999, Lisboa, Portugal. Disponível em: <http://ecoop99.di.fc.ul.pt/wegner.pdf>. Acesso em: set. 2001.

WEGNER, P. **A Research Agenda for Interactive Computing**. Disponível em: <http://www.cs.brown.edu/people/pw>. Acesso em: set. 2001.

WEGNER, P.; GOLDIN, D. Interaction as a Framework for Modeling. In: CHEN, P. P.; AKOKA, J.; KANGASSALO, H.; THALHEIM, B. (Ed.). **Conceptual Modeling**: current issues and future directions. Berlin: Springer, 1999. p.243–257. (Lecture Notes in Computer Science, v.1565).

WEGNER, P.; GOLDIN, D. **Mathematical Models of Interactive Computing**. Brown: Brown University, 1999. (CS-99-13).

WEGNER, P.; GOLDIN, D. Coinductive Models of Finite Computing Agents. **Electronic Notes in Theoretical Computer Science**, [S.l.], v.19, 2000.

WEGNER, P.; GOLDIN, D. **Interaction, Computability, and Church's Thesis**. Aceito para publicação no British Computer Journal. Disponível em: <http://www.engr.uconn.edu/ dqg/papers>. Acesso em: dez. 2001.

WIEDERMANN, J. **Parallel Turing Machine**. Utrecht, Netherlands: Department of Computer Science of University of Utrecht, 1984. (RUU-CS-84-11).

WIEDERMANN, J. Weak Parallel Machines: a new class of physically feasible parallel models. In: INTERNATIONAL SYMPOSIUM MATHEMATICAL FOUNDATIONS OF COMPUTER SCIENCE, 17., 1992, Prague, Czechoslovakia. **Proceedings...** Berlin: Springer, 1992. p.95–111. (Lecture Notes in Computer Science, v.629).

WIEDERMANN, J. Quo Vadetis, Parallel Machines Models. In: LEEUWEN, J. van (Ed.). **Computer Science Today**. Berlin: Springer, 1995. p.101–114. (Lecture Notes in Computer Science, v.1000).

WORSCH, T. On Parallel Turing Machines with Multi-head Control Units. **Parallel Computing**, Netherlands, v.23, p.1683–1697, 1997.

WORSCH, T. Parallel Turing Machines with One-head Control Units and Cellular Automata. **Theoretical Computer Science**, [S.l.], v.217, n.1, p.3–30, 1999.

ZALTA, E. N. **Stanford Encyclopedia of Philosophy**. Disponível em: <http://plato.stanford.edu/entries/church-turing>. Acesso em: ago. 2001.