UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Renato Corrêa Vieira

Descrição de Comportamentos Robóticos Utilizando uma Abordagem Gramatical e sua Implementação através de Redes Neurais Artificiais

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Mauro Roisenberg, Dr.
Orientador

Florianópolis, Abril de 2004

Descrição de Comportamentos Robóticos Utilizando uma Abordagem Gramatical e sua Implementação através de Redes Neurais Artificiais

Renato Corrêa Vieira

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Sistemas de Conhecimento e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Raul Sidnei Wazlawick, Dr.

Coordenador do Curso

Banca Examinadora

Prof. Mauro Roisenberg, Dr.

Orientador

Prof. Olinto José Varela Furtado, Dr.

Co-Orientador

Prof. Jorge Muniz Barreto, Dr.

Prof. Jovelino Falqueto, Dr.

Prof. Paulo Martins Engel, Dr.

"Um impulso de algum tipo precede o ato generativo da mente, e através dessa vontade de buscar e encontrar o conhecimento, o conhecimento em si nasce." (Santo Agostinho)
"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original." (Albert Einstein)

Agradecimentos

Agradeço primeiramente a Deus e a padroeira dos paraeneses Nossa Senhora de Nazaré por permitirem, à luz da fé, manter a chama acesa da força de vontade e determinação em realizar meus sonhos.

Se torna propício agradecer aqueles que me ajudaram a iniciar esta jornada, como os amigos e professores Luis Lacerda, Gustavo Campos, Daniel Martins e Polyana Fonseca.

Aos amigos e amigas feitos em Florianópolis, Marcelo Henrique, ao jovem Alexandre, Gilberto, Luciene (pelas suas inúmeras dicas de LaTeX), Gisele, Humberto, Peri, Fernanda, D. Maria e todos os outros que me acolheram e deram suporte a minha estadia durante estes dois anos nesta ilha maravilhosa.

Também não posso deixar de agradecer aos amigos da terrinha Serge, Claudinho, Fabrício, Artur Tupiassú, Márcio Passos, Diego Pacheco e outros que porventura não consiga lembrar neste momento. Embora distantes, mantenho grande estima e apreço por vocês.

Devo agradecer também ao grande amigo Marcelo Tenório, tanto pela sua dedicação em me ajudar com o desenvolvimento gráfico deste trabalho, como pela sua sincera e leal amizade ao longo destes dois últimos anos.

De certo, vale o agradecimento ao meu grande e eterno amigo Anderson Rodrigues pelas intensas discussões filosóficas e científicas, além de sua palavra sempre incentivadora e amiga.

Agradeço a Universidade Federal de Santa Catarina e a todos os professores que a fazem uma das universidade mais gabaritadas deste país.

É importante ressaltar e agradecer a participação do Prof. Jorge Muniz Barreto que com sua grande experiência científica contribuiu com valiosas sugestões para este trabalho.

Destaco ainda a participação fundamental do meu co-orientador Prof. Olinto Furtado

pela sua valiosa contribuição na área de linguagens formais, passo este, primordial para o desenvolvimento deste trabalho.

Sobretudo, agradeço ao meu orientador Mauro Roisenberg pelo seu exemplo de profissionalismo em uma universidade pública cada vez menos valorizada pelo sistema. Também destaco sua competência e os ensinamentos repassados a mim que efetivaram a concretização deste trabalho. Valeu Professor!!!

Com muito orgulho agradeço a minha namorada Izauria Zardo pelo amor que nutre por mim, pela compreensão da minha ausência em virtude da dissertação, por me fazer sorrir e me ensinar o caminho da garra quando desanimava.

A minha irmã, Renatinha, pelo seu jeitinho meigo e carismático, e que embora ausente fisicamente, está sempre presente no meu coração.

Ao meu pai, pelo exemplo que é de vencedor, de quem muito me orgulho por sua garra e visão de mundo. Também agradeço seus importantes ensinamentos ao longo de minha vida.

Qualquer quantidade de palavras seria pouca para agradecer a minha mãe pelo seu incentivo, amor, carinho e sobretudo confiança depositada em mim. Sem você, mamãe, certamente não teria condições de realizar este trabalho.

Publicações

VIEIRA, R. C.; ROISENBERG, M. & FURTADO, O. J. V. Formal languages aspects as a tool for representation and implementation of behavior-based robotics. *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Singapore, 2004.

VIEIRA, R. C.; DE ALMEIDA E SILVA, F. & ROISENBERG, M. Redes neurais hierárquicas para controle robótico. *IV Congresso Brasileiro de Computação - CBComp*, Itajai-SC, 2004.

ROISENBERG, M.; BARRETO, J. M.; DE ALMEIDA E SILVA, F.; VIEIRA, R. C. & COELHO, D. K. Pyramidnet: A modular and hierachical neural network architecture for behaviour based robotics. *ISRA - International Symposium on Robotics and Automation*, Queretaro, Mexico, 2004.

VIEIRA, R. C.; TENÓRIO, M. B.; ROISENBERG, M. & BORGES, P. S. S. Comparação entre redes neurais artificiais e rough sets para classificação de dados. *VI Brazilian Conference on Neural Networks*, São Paulo-SP, 2003.

VIEIRA, R. C. & ROISENBERG, M. Redes neurais artificiais: Um breve tutorial. Relatório Técnico. *UFSC/INE/L3C-01/2003, Laboratório de Conexionismo e Ciências Cognitivas - L3C, Universidade Federal de Santa Catarina* - Florianópolis - SC, 2003.

BARRETO, A. S.; VIEIRA, R. C.; NASSAR, S. M & BORGES, P. S. S. Redes Bayesianas e Produção de Conhecimento: Uma abordagem de Data-Mining em dados de um concurso vestibular. *IT CONFERENCE*, Cuiabá - MT, 2002.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AA	Agentes Autônomos
AF	Autômato Finito
ALL	Autômato Linearmente Limitado
AFD	Autômato Finito Determinístico
AFND	Autômato Finito Não Determinístico
AFSM	Augmented Finite State Machine
AP	Autômato de Pilha
BL	Behavior Language
BMP	Binary Mapping Perceptron Module
ELA	Esclerose Lateral Amiotrófica
ER	Expressões Regulares
GLC	Gramática Livre de Contexto
GR	Gramática Regular
GSC	Gramática Sensível ao Contexto
K	Potássio
LER	Linguagem Enumeravelmente Recursiva
LLC	Linguagem Livre de Contexto
LR	Linguagem Regular
LSC	Linguagem Sensível Contexto
LTP	*Long Term Potentiation*
MT	Máquina de Turing
Na	Sódio
NNPDA	Neural Network Pushdown Automata
NNS	Neural Network Stack
PFA	Padrão Fixo de Ação
RAP	Reactive Action Packages
RBC	Robótica Baseada em Comportamento
RIPE	Robot Independent Programming Language
RNA	Rede Neural Artificial
RS	Robot Schemas
RSC	Rede Seqüencial em Cascata

Resumo

A Robótica Baseada em Comportamentos (RBC) se baseia na emergência de comportamentos robóticos de modo a garantir inteligência e autonomia nas ações que um agente deve descrever para alcançar seus objetivos.

Desta forma, surge a necessidade de se achar uma maneira formal de representar estes comportamentos robóticos, mantendo características como complexidade, concisão e compactação na representação. Neste trabalho se afirma que as linguagens contidas na hierarquia de Chomsky são capazes de representar esta variada gama de comportamentos robóticos.

Tendo em vista a formalização de assuntos pertinentes a robótica, neste trabalho também é feita uma nova definição formal dos agentes autônomos (AAs), os quais aparecem como tendo seu funcionamento tal qual uma máquina de Turing.

Além disso, procura-se fazer paralelos com modelos encontrados na natureza para se encontrar inspirações de como os comportamentos robóticos podem ser implementados. Assim, é proposta a utilização das redes neurais artificiais (RNAs) para a implementação dos comportamentos robóticos descritos pelas linguagens de Chomsky.

Palavras Chave: agentes autônonomos, hierarquia de Comsky, comportamentos robóticos, redes neurais artificiais.

Abstract

The Behavior-Based Robotics has its foundations in the emergence of robotic behaviors, and aims at providing intelligence and autonomy to actions performed by agents in search of their goals.

To attain that objective, it becomes necessary to find a proper way to represent robotic behaviors, although keeping some specific features, such as complexity and conciseness. In this work it shall be asserted that the languages pertaining to Chomsky Hierarchy are adequate to represent the great variety of robotic behaviors.

Taking into account the need to formalize the topics related to Robotics, a new strict definition of autonomous agents (AAs) is developed, where the AAs are represented as a Turing machine.

In addition, this work makes an effort to establish a parallel with natural models in order to achieve an insight of how the robotic behaviors can be set up. Hence, it is proposed to use artificial neural networks (ANNs) to implement robotic behaviors represented by the Chomsky Languages.

Keywords: autonomous agents, Chomsky hierarchy, robotic behaviors, artificial neural networks

Capítulo 1

Introdução

1.1 Motivação

No decorrer da história da humanidade é salutar o desejo humano da criação de artefatos que apresentem características autônomas, tendo em vista fatores como otimização, comodidade e de um modo geral, resolução de problemas difíceis e incômodos de serem executados, como tarefas repetitivas, desgastantes, perigosas, etc.

Esta característica intrínseca do ser humano de criar entidades autônomas remonta desde mitos gregos, como o do escultor Pigmalião, que criou em forma de estátua a mais bela das mulheres e a chamou de Galatéia. No entanto, uma solene frustração recaiu sobre o nobre escultor, pois apesar de vislumbrante, sua obra era inanimada. Assim, pediu a Afrodite que a transformasse em uma mulher, e enquanto ele dormia, Afrodite atendeu seu pedido. Galatéia se tornou uma mulher e casou-se com seu criador [85].

Na mesma linha, também existe a lenda da criatura feita de barro chamada Golen, que tinha propósitos de proteção a comunidade judaica e realização de trabalhos braçais. Nesta lenda já surge a necessidade da criação de uma linguagem que descrevesse o comportamento da criatura, em que palavras mágicas fixadas na testa da criatura regiam seu comportamento. Isto era feito utilizando as palavras “EMET” (verdade) ou “MET” (morte), de modo que enquanto a palavra “EMET” estivesse em vigência, Golen fazia tudo que seu mestre pedia, e quando a tira de papel fosse trocada para “MET” a criatura era desativada, ou seja, este mecanismo funcionava como um botão “liga-desliga”. [100].

Na Idade Média um alquimista chamado Paracelsus dizia-se capaz de criar pequenos seres batizados de *homunculus* que o ajudavam para tarefas braçais. Não se pode deixar de citar os autômatos provenientes da relojoaria e da mecânica de precisão, como o pato de Jacques de Vaucanson que continha, em apenas uma asa, mais de 400 peças articuladas, de modo a imitar os movimentos de um pato [41].

Em época contemporânea, na literatura os livros “Frankestein” de Mary Shelley, “O Caçador de Andróides” de Philip Dick, os romances de Isaac Asimov como “O Homem Bicentenário”, dentre outros, são sucesso entre os mais variados nichos de leitores. Além da literatura, o cinema obteve virtuoso sucesso de bilheterias com filmes como “Guerra nas Estrelas”, “Blade Runner”, “Matrix” e “Inteligência Artificial”.

Observa-se em todos estes casos da ficção científica, que a emergência de comportamentos robóticos é resultado de uma comunicação através de linguagem natural. Isto dá margem a uma série de dificuldades, como a ambiguidade e a necessidade de uma grande quantidade de senso comum, de modo que ainda hoje em dia, este tratamento via linguagem natural, ainda permanece no âmbito da ficção.

Fora do escopo fictício e lendário, aplicações de robôs que requerem autonomia podem ser achadas no âmbito de exploração espacial, como o robô chamado “PAW” que tem como objetivo colher amostras do solo de Marte, bem como obter dados da atmosfera do planeta vermelho em busca de resquícios de vida marciana. [1]

Apesar de fora do enfoque deste trabalho, é importante citar trabalhos de vida artificial como os do biólogo Thomas Ray [86][87][88] que criou um sistema baseado em uma máquina virtual em que organismos digitais (programas), seguindo algumas heurísticas da evolução e genética, deveriam se reproduzir e competir entre si de modo a garantir memória e tempo de processamento.

Finalmente, pode-se citar o ambicioso projeto do robô COG que foi projetado para simular os sentimentos humanos [20] utilizando uma alta carga de processamento e lançando mão de idéias, outrora vistas como ficção científica, como a inspiração do aprendizado do robô nas chamadas “Child Machines”, profetizadas por Allan Turing em seu artigo C*omputing Machinery and Intelligence* de 1950 [104].

Estudos das últimas décadas trazem à tona a utilização de técnicas de inteligência artificial, como na abordagem da Robótica Baseada em Comportamento em que a emergência

de comportamento inteligente e autônomo se dá através de uma implementação comportamental, seguindo diferentes arquiteturas de controle. [21][91][68][65].

Um dos grandes problemas da área da robótica baseada em comportamentos é como descrever as ações que um robô deve efetivar, isto é, como descrever os comportamentos robóticos através de linguagens que possam representar os mais variados tipos de comportamentos robóticos, sendo ao mesmo tempo precisas e formais.

As linguagens para descrição de comportamentos encontradas atualmente na literatura [51][50][62][70][76][21] se baseiam em fatores como arquitetura de hardware e linguagens de programação específicas, o que corrobora certas restrições nos projetos de Agentes Autônomos, como uma dependência do aparato físico do robô, bem como problemas de portabilidade sob o aspecto da coarctação em uma determinada linguagem de programação.

Desta forma, surge a necessidade de um formalismo para a descrição de comportamentos robóticos, de modo a prover uma linguagem comum e completa para a expressão destes comportamentos.

Aliado a isto, da mesma maneira que estudos de etologia [69][35][4] classificam os comportamentos dos seres vivos seguindo diferentes níveis de complexidade, torna-se conveniente idealizar um modo de classificar os comportamentos robóticos utilizando algum modelo que consiga representar a complexidade dos diferentes espectros destes comportamentos.

As linguagens de Chomsky e seus autômatos correspondentes são modelos de descrição formal bastantes poderosos, onde suas características hierárquicas conseguem englobar a complexidade das estruturas a serem representadas.

Assim, utilizando diferentes níveis da hierarquia de Chomsky, com suas linguagens e autômatos peculiares, se tem uma estrutura ao mesmo tempo formal e flexível para a representação de comportamentos robóticos, respeitando diferentes facetas de complexidades inerentes a estes comportamentos.

Por outro lado, de posse de um modelo de representação dos comportamentos robóticos, surge a necessidade de encontrar uma ferramenta que implemente o formalismo adquirido com a hierarquia de Chomsky.

Haja vista que os comportamentos emergidos pelos seres vivos se dão através do funci-

onamento do sistema nervoso central, que possui como elementos atômicos os neurônios, e conseqüentemente as estruturas de redes neurais que conseguem efetivar um mapeamento entre as informações captadas do ambiente pelo seu aparato sensório e a emergência de um comportamento adequado naquele contexto ambiental, se torna interessante achar junto à inspiração biológica das redes neurais, modelos artificiais destas redes que consigam implementar os comportamentos robóticos propostos.

Além disso, as redes neurais artificiais são estruturas que suportam imperfeições no ambiente, possuem tolerância a falhas, são capazes de fazer generalizações e interpolações, ou seja, trazem características convenientes para aplicações na robótica.

Assim, neste trabalho se propõe a utilização das redes neurais artificiais como modelos para a implementação de comportamentos robóticos, seguindo a representação destes comportamentos sob a ótica das linguagens de Chomsky.

1.2 Objetivos

1.2.1 Objetivo Geral

O foco principal desta dissertação é mostrar que os diferentes níveis da hierarquia de Chomsky podem representar comportamentos robóticos, aplicando as redes neurais artificiais como modelos para a implementação destas linguagens.

1.2.2 Objetivos Específicos

- Fazer um levantamento das bases científicas para o estudo de comportamentos no âmbito biológico e mental, posteriormente analisar os paradigmas de arquiteturas robóticas, elucidando exemplos das mesmas;

- Estudar algumas das linguagens para descrição de comportamentos existentes na literatura, alçando suas qualidades e limitações;

- Fazer um estudo sobre os diferentes níveis da hierarquia de Chomsky, expondo suas características em níveis de gramáticas e autômatos, e seqüencialmente efetuar a modelagem de comportamentos robóticos utilizando as idéias de Chomsky;

- Realizar uma formalização da definição de um agente autônomo, sob o prisma da máquina de Turing.

- Encontrar na literatura modelos de redes neurais artificiais que implementem os diferentes níveis das linguagens de Chomsky, mostrando a plausibilidade das RNAs na implementação destas linguagens.

- Acrescentar uma nova camada à arquitetura PiramidNet, na qual a emergência de comportamentos mais complexos será através de modelos de RNAs que implementem autômatos de pilha e autômatos linearmente limitados.

1.3 Estrutura da Dissertação

No **capítulo 2** serão abordados aspectos gerais da robótica baseada em comportamentos, entornando algumas bases biológica no estudo dos comportamentos animais, para em seguida discutir os principais paradigmas da RBC, exemplificando arquiteturas para implementação destes enfoques.

Com o intuito de fazer um levantamento bibliográfico sobre as linguagens para descrição de comportamentos, o **capítulo 3** engloba diferentes linguagens que modelam comportamentos nos diferentes paradigmas da RBC.

O **capítulo 4** faz um estudo dos diferentes níveis da hierarquia de Chomsky, abordando os aspectos dos dispositivos geradores e reconhecedores de linguagem, fornecendo antes uma terminologia básica para o estudo destas teorias. Por conseqüência, diferentes comportamentos robóticos são abordados, fazendo a devida classificação deles em seus respectivos níveis da hierarquia de Chomsky. Além disso, uma discussão sobre a descrição formal de um agente como uma máquina de Turing é realizada.

No **capítulo 5**, é proposta uma nova definição para os agentes autônomos, bem como para os comportamentos emergidos por estes. Esta definição é mantida sob a ótica formal da máquina de Turing.

As redes neurais artificiais como mecanismos capazes de implementar comportamentos robóticos que seguem a hierarquia de Chomsky são tratadas no **capítulo 6**, onde é discutido os modelos de RNAs apropriados para fazer esta implementação. Também é

feita uma contextualização dos mecanismos da memória humana com os autômatos da hierarquia de Chomsky. Finalmente, se tem um acréscimo de camadas a arquitetura PiramidNet, através do incremento de módulos que permitam a implementação via RNAs de autômatos de pilha e autômatos linearmente limitados.

No último capítulo são comentados as conclusões obtidas com este estudo, objetivando realçar se as proposições idealizadas neste trabalho foram cumpridas.

Capítulo 2

Agentes Autônomos e Robótica Baseada em Comportamento

"Para entender como aprendemos e nos lembramos, ou de onde vem o pensamento, a consciência, e o auto conhecimento, exige-se um salto tão grande quanto o dado por Galileu quando ele ignorou a resistência do ar, ou Newton, quando estendeu a gravidade da superfície da Terra à Lua, ou Einstein, quando redefiniu o tempo" (Leon N. Cooper)

2.1 Introdução

Ao se tratar de Agentes Autônomos, torna-se relevante definir o termo agente. Russel & Norvig [94] apresentam uma definição que apesar de simples, consegue ser bastante global. *"Um agente é tudo que pode ser visto percebendo seu ambiente através de sensores e atuando no ambiente através de atuadores."* [1]

Geralmente, os ambientes do mundo real tendem a oscilar entre diferentes estados, ou seja, a configuração espacial, climática, luminosa, e as relações entre entidades (obstáculos, marcos de orientação, etc.) em geral são passíveis de mudanças no decorrer do tempo, tornando complexo o desenvolvimento de agentes autônomos. Além da questão da dinâmica

[1] Na próxima sub-seção uma definição mais aprofundada e menos genérica será apresentada, onde os agentes mantém alguma entidade de cunho cognitivo para que possam realizar o mapeamento entre *percepção* e *ação*.

do ambiente, existem situações em que o ambiente é pouco conhecido, difícil de ser acessado, perigoso e até mesmo hostil. Como exemplos se pode citar pesquisas em águas submersas no ambiente gélido da Antártida [79], a exploração do vulcão do Monte Erebus [111] e o projeto do Robô Nômade ("Nomad Robot") que navega em busca de meteoritos [59]. Outro exemplo que vem chamado atenção é o desenvolvimento de robôs para a exploração de Marte no que tange aspectos como análise da atmosfera, material do solo, condições climáticas e resquícios de vida. [109][99][1]

2.2 Autonomia e Inteligência

Dado o contexto da dinâmica ambiental, convém dotar os agentes de certa independência para que possam lidar de maneira mais próxima da ótima com as "adversidades" do ambiente, surgindo assim a idéia de autonomia.

Os agentes que apresentam autonomia, isto é, Agentes Autônomos apresentam uma entidade cognitiva que tem a responsabilidade de fazer um mapeamento entre os dados oriundos dos sensores para as ações a serem executadas pelos atuadores. A capacidade de autonomia de um agente está intimamente relacionada com a flexibilidade e coerência deste mapeamento.

Segundo FOGEL[37] apud Roisenberg[89] *"pode-se dizer que um requisito necessário para dotar um agente de autonomia é dotá-lo de inteligência. Esta inteligência deve tornar o agente capaz de se adaptar, ou seja, modificar o mapeamento entre sensações e ações de maneira a manter ou melhorar o seu desempenho de operação no ambiente"*.

Desta forma, construir um agente que consiga fazer emergir comportamentos, onde a mutabilidade do ambiente não interfira no desempenho global dos objetivos propostos para o agente, ou seja, o mapeamento estar preparado para quais forem as percepções sensórias, de modo a tomar uma decisão adequada, é o que define o fato do agente ser autônomo.

Roisenberg [89] sintetiza claramente esta discussão ao definir um AA. *"Um Agente Autônomo é um sistema computadorizado capaz de extrair informações do seu ambiente (eventualmente este ambiente é computacional) e, através de alguma capacidade cognitiva, mapear as informações extraídas em ações que, eventualmente, podem afetar o*

ambiente de modo a alcançar os objetivos para os quais foi projetado".

Duas considerações importantes em relação ao escopo deste trabalho devem ser consideradas:

- Agentes serão enfocados sob a ótica da robótica.
- No restante deste trabalho muito será tratado sobre comportamentos, uma concepção genérica do que são, está no resultado que a função de mapeamento irá fazer emergir para o ambiente.

2.3 Bases científicas para o estudo do comportamento

Para abordar a robótica baseada em comportamentos, é interessante ressaltar brevemente algumas ciências afins que estudam o comportamento animal. Este breve levantamento tem o objetivo de justificar a abordagem baseada em comportamentos como um paradigma plausível para o desenvolvimento de soluções na robótica.

2.3.1 Neurofisiologia

O ser humano contém um conjunto de sensores e atuadores, como exemplos dos primeiros podemos citar a visão, olfato, audição, paladar e tato. Dentre os atuadores, os exemplos principais são os membros inferiores e superiores.

A geração de um comportamento se dá quando um sensor recebe informação do ambiente e esta é encaminhada via nervos aferentes para o sistema nervoso central, mais especificamente para o tálamo, (exceto sinais oriundos do olfato) que agrega todos os sinais antes de passar ao córtex central. No córtex central existem diversas regiões com funções específicas e que são ativadas através dos sinais enviados pelo tálamo. Ao chegar no córtex a informação é processada e encaminhada através de impulsos nervosos para os nervos eferentes que irão fazer os atuadores efetuarem o comportamento.

Convém ressaltar que esta foi uma discussão marginal sobre o processo de funcionamento do cérebro no que tange a emergência comportamental, nem mesmo quesitos como o papel do cerebelo de "agrupador lógico" de sinais nervosos para o desempenho de comportamentos mais complexos foi considerado. No entanto, o intuito foi comentar (mesmo

que de forma breve) como o processo de elaboração de um comportamento ocorre, posto a relativa relevância disso para este trabalho. Caso o leitor se interesse em saber mais sobre o funcionamento fisiológico do cérebro, uma leitura em [55], [69] e [32] será de grande valia.

Outro ponto importante a ser comentado no que tange esta sub-seção é a inspiração biológica que estes estudos trazem à tona, principalmente com o estudo das redes neurais artificiais, que serão vistas capítulo 5. Além da arquitetura Piramidnet de controle hierárquico e paralelo de comportamentos robóticos que será introduzida neste mesmo capítulo na seção 2.5.4.

2.3.2 Psicologia

Segundo PANI[78] apud Arkin[8], desde os estudos de Weber e Fechner que estabeleceram relações entre a intensidade dos estímulos com a percepção sensória, a psicologia vem tentando estudar o funcionamento do comportamento humano no que concerne a seus aspectos psíquicos.

O estudo do comportamento teve um grande salto através da teoria do behavorismo de Watson [110] que realizou estudos sobre a influência do meio no comportamento animal e humano a partir de um programa de estímulo e resposta, proclamando que todo estímulo eficaz provoca sempre uma resposta imediata, de alguma espécie. Assim, Watson definia a psicologia como sendo *"a ciência que estuda o comportamento observável, mensurável e possível"*.

Outra vertente da área psicológica é a teoria da Gestalt [54]. Nesta teoria propõe-se que o conhecimento se produz porque existe no ser humano uma capacidade interna inata que predispõe o sujeito ao conhecimento; há uma super valorização da percepção como função básica para o conhecimento da realidade. Assim, a entidade cognitiva da percepção proporciona o processo do resposta do indivíduo, frente a um dado estímulo. Em suma, pode-se concluir que a maneira que um estímulo é percebido será determinante para o comportamento resultante.

Posteriormente, surge a teoria cognitiva em que o comportamento é fruto resultante de ações recíprocas entre o indivíduo e o meio, definindo a cognição como *"a atividade de conhecer: a aquisição, organização e utilização do conhecimento"* [74].

Esta abordagem, segundo Arkin[8], mantém algumas asserções que justificam este enfoque como intimamente relacionado a processos computacionais:

- Uma série de subsistemas processa as informações ambientais (ex. Estímulo → atenção → percepção → processos de pensamento → decisão → resposta).
- Os subsistemas individuais transformam os dados sistematicamente.
- Processamento de informação em pessoas está fortemente correlacionado com o que acontece em computadores.

A aplicação destas teorias da psicologia na robótica não deve ser necessariamente um prisma a ser seguido, mas buscar fontes de inspiração em estudos tão bem amparados filosoficamente pode ser uma fonte de sucesso na criação de arquiteturas robóticas.

2.3.3 Etologia

Em 1973 os cientistas Karl Von Frisch, Konrad Lorenz e Nikolas Tinbergen foram agraciados com o prêmio Nobel de Medicina e Fisiologia devido aos seus trabalhos pioneiros que originaram a ciência denominada de Etologia, que vem a ser o estudo do comportamento animal, geralmente quando observado em seu habitat natural.

Apesar de toda a fundamentação vista até o presente momento sobre o estudo dos comportamentos, cabe aqui uma clássica e abrangente definição de comportamento animal, segundo J. B Messenger [69]

> *"Pode-se dizer que (o comportamento animal) abarca todas as atividades de procurar comida, evitar perigo, acasalar-se e mesmo criar uma prole. Todas essas atividades requerem órgãos dos sentidos que coletam informações, principalmente sobre as alterações no mundo externo, um sistema nervoso que processa estas informações e sistemas efetores, tais como glândulas, músculos ou fotóforos que traduzem a saída (output) do sistema nervoso. O que caracteriza o comportamento tipicamente, entretanto, é o fato de manifestar-se através da atividade - ou inatividade - do sistema muscular. Uma anêmona ao fechar-se, um leão ao dar o bote, um chimpanzé que*

sorri, uma prima-dona ao cantar: de algum modo estão se comportando ao executarem tais atos."

Outro ponto passível de discussão é o fato da emergência de inteligência frente ao âmbito comportamental. Por vezes no mundo animal os comportamentos surgem com fins de executar tarefas que vão desde o instinto por sobrevivência, passam por agregações de comportamentos com cunho reprodutivo e chegam ao ápice de desenvolvimento através de exemplos como junção de comportamentos em sociedade com objetivo de otimização de tarefas que se tomadas individualmente seriam sub-ótimas e até mesmo inoperantes.

Além disso, surgem as chamadas classes de comportamentos, que variam em nível de complexidade, indo desde comportamentos puramente reflexivos, até comportamentos com alto teor cognitivo, como os comportamentos racionais. A seguir serão comentadas estas diferentes classes de comportamentos.

2.3.3.1 Taxias

Comportamentos deste tipo tendem a adaptar o organismo para padrões que estão muitas vezes relacionados com questões funcionais deste mesmo organismo, frente aos estímulos ambientais. No texto de Dethier & Stellar [35], é dado o seguinte exemplo:

"Um organismo pode se orientar em uma fonte de luz de tal forma que os dois olhos recebam estímulo igual. Se a fonte for deslocada lateralmente, a orientação mudará porque um olho está recebendo agora mais iluminação que o outro. Se um olho for removido ou pintado, o organismo irá se mover continuamente em círculos, como se estivesse tentando igualar a luz nos dois olhos. Esta orientação, guiada contínua e especificamente por estímulos externos, é chamada de taxia."

2.3.3.2 Reflexos

A Esclerose Lateral Amiotrófica (ELA) [93] é uma degeneração progressiva dos neurônios motores no cérebro (neurônios motores superiores) e na medula espinhal (neurônios motores inferiores), ou seja, estes neurônios perdem sua capacidade de funcionar adequadamente, isto é, de transmitir os impulsos nervosos.

Em uma primeira análise clínica o médico que deseja diagnosticar a ELA, faz alguns testes como pressionar a planta do pé com um objeto moderadamente rígido, movendo de baixo para cima em direção aos dedos do pé. Caso ocorram contrações nos dedos e movimentação do dedão para trás há um grande indício de distúrbio no neurônio motor superior.

Outro teste conveniente é o teste patelar padrão, onde o médico bate com certa força no joelho do paciente, e este deve balançar seu membro inferior para frente.

Finalmente, pode-se utilizar o teste do umbigo, onde o médico executa pequenos golpes na região inferior e superior ao umbigo, e o resultado deve ser uma contração nos músculos do abdome e uma movimentação involuntária do umbigo durante o estímulo.

Estes testes de auxílio ao diagnóstico, baseiam-se no comportamento reflexivo (também chamado de comportamento de arco-reflexo), eles geralmente são respostas de partes do corpo, frente a algum estímulo que não é necessariamente o que está em contato maior com o estímulo ambiental . Da mesma forma que nas taxias, a intensidade e duração deste comportamento estão conectadas proporcionalmente à duração e intensidade do estímulo ambiental que incitou os comportamentos em questão.

2.3.3.3 Reativos ou Padrão Fixo de Ação (PFA)

Roisenberg [89] faz uma análise bastante sucinta sobre este tipo de comportamento: *"Esta classe de comportamento é formada por uma série de comportamentos estereotipados como resposta a um dado estímulo. O estímulo que dispara o comportamento é geralmente mais complexo e específico que o necessário para disparar um comportamento reflexivo. A resposta, por sua vez, envolve uma seqüência temporal de ações que se desenrolam até o seu final, mesmo que o estímulo disparador não esteja mais presente. As respostas que compõem o comportamento reativo podem estar relacionadas de um modo intrincado, no qual cada resposta componente é disparada pelo final da ação precedente ou por algum estímulo ou sinal proveniente do ambiente e que é alcançado como resultado da ação precedente. Se qualquer dos sinais disparadores de uma ação da sequência for inibido, todas as ações seguintes não serão disparadas, mesmo que os sinais seguintes sejam apresentados."*

O exemplo a seguir extraído de ALCOCK[4] apud [57] ilustra bem este tipo de com-

portamento através da tática de *codebreaking* que consiste de algumas espécies aproveitarem o PFA de outras.

> *"Um code breaker de primeira é o besouro Atemeles pubicollis, o qual deposita seus ovos em ninhos da formiga Formica polyctena. A larva desenvolvida produz um odor atraente, ou feromônio, que induz as formigas trabalhadoras a levar o besouro parasita para a incubadora, onde o verme regala-se com os ovos e larvas de formigas. Não contente com esta despensa, o parasita imita a súplica por comida típica da larva de formiga batendo com partes de sua boca nas mandíbulas de uma formiga trabalhadora. Esta ação é um liberador (releaser) que dispara a regurgitação (um PFA) da formiga. A larva eventualmente transforma-se em besouro adulto que também imita o liberador para as formigas trabalhadoras adultas alimentarem-no."*

2.3.3.4 Instintivo ou Motivado

O comportamento instintivo ou motivado parece ser algo intrínseco do organismo de um dado indivíduo. Ele obtém suas características funcionais através de fatores como estímulos externos, hormônios e influências nervosas centrais excitatórias.

Também pode ser atrelado ao conceito de instinto, a sensação de saciedade que se obtém perante a realização de um objetivo para o qual aquele comportamento motivacional foi desencadeado.

Assim, é uma justificativa razoável para necessidades físicas, que por vezes confundidas por emoções obliteradas, em que ao efetivarmos sentimos uma grande sensação de dever cumprido, ou melhor de saciedade.

Convém ressaltar que estas reações não competem apenas ao ser humano. Um exemplo clássico disso é o ganso cinzento que recupera um ovo que rolou fora do ninho, empurrando-o entre suas pernas com o lado inferior do bico e pode continuar estes movimentos zelosos de empurrar até o ninho, mesmo que o ovo tenha rolado para fora do seu alcance [35] [12].

2.3.3.5 Racional

Esta abordagem geralmente concerne sobre aspectos inerentes a espécies de alto degrau na escada filogenética. Os comportamentos desencadeados não dependem mais de processos de nível endócrino, nem necessariamente do estado do organismo frente ao ambiente.

Surge aqui a questão sobre entidades abstratas simbólicas que representam o estímulo-resposta como apanágio do próprio interior do organismo, existindo uma alta relevância no que tange a capacidade cognitiva, do qual acredita-se ser bem mais elevada no *Homo sapiens sapiens*.

Embora, seja inevitável o antropocentrismo quando se reflete sobre o comportamento racional, existem diversos estudos que incitam a uma reflexão sobre as características da racionalidade animal.

Uma das experiências foi a do psicólogo alemão Wolfgan Kohler que testou o chimpanzé Sultão para solucionar o problema de alcançar um pedaço longínquo de banana conectando pequenos bastões (HILGARD e ATKINSON[46] apud [57]).

Outro estudo focou o problema do contorno, o qual o animal deve ser capaz de contornar uma parede para chegar ao alimento, sem bater na parede antes (tentativa e erro), mostrou que macacos e chimpanzés foram capazes de solucionar o problema em sua primeira tentativa [89].

O leitor interessado pode buscar em [95][96] mais experimentos que corroboram a necessidade de uma reflexão sobre a classificação etológica atual.

2.4 Robótica Baseada em Comportamento

Ao unir o âmbito dos agente autônomos com as bases dissertadas nas seções anteriores se obtém uma inspiração plausível para a chamada Robótica Baseada em Comportamento ou Agentes Autônomos Baseados em Comportamento.

Este paradigma capacita o projeto de agentes autônomos que são capazes de emergir comportamentos inteligentes através do mapeamento entre percepção e ação. Os comportamentos robóticos podem ser classificados mesmo através das bases dos estudos de etologia, isto é, eles seguem complexidades diferentes, e o que vai determinar a emergência deles é o *modus operandi* da sua interação com o ambiente.

Duas questões que concernem a RBC são o favorecimento ao paralelismo e a descentralização. Comportamentos podem emergir em conjunto, isto é, não se restringem a uma execução serial, bem como se dividem em módulos de comportamentos destinados a tarefas específicas que podem ser disparados tanto por estímulos provenientes do ambiente, ou mesmo por sinais de outros comportamentos, não sendo necessário um controlador central para a manipulação da informação. [68].

A RBC pode ser analisada com vistas ao enfoque dado à implementação, sendo classificada em três vertentes: deliberativa, reativa e híbrida.

2.4.1 Abordagem Deliberativa

O projeto de um AA utilizando esta abordagem é feito através de um reconhecimento por completo do ambiente para a construção de um mapa que irá dirigir o robô na sua navegação pelo ambiente.

Através de um pré-processamento sensorial que cuida de extrair informações úteis do ambiente, há um processo de mapeamento de um modelo de mundo para que as ações sejam planejadas antes de serem, de fato, executadas. [65]

O agente é projetado utilizando uma representação do mundo em que ele irá atuar, ou seja, é implantada uma entidade cognitiva que controlará os comportamentos através de planos relacionados com o mapa do mundo que o agente contém.

A grande problemática ao se adotar esta vertente é o fato dos ambientes do mundo real estarem sempre sofrendo mudanças. Assim, dada esta dinâmica ambiental, a tarefa de organizar planos abrangentes suficientemente para que o robô possa obter autonomia torna-se uma tarefa bastante difícil

2.4.2 Abordagem Reativa

Quando comparada com a vertente deliberativa, pode-se dizer que a proposta reativa é o outro extremo. Não há construção de modelos simbólicos do ambiente, os comportamentos são disparados diretamente ante aos estímulos recebidos. As arquiteturas contidas nesta abordagem tem uma facilidade maior nos quesitos de flexibilidade e adaptabilidade, haja vista que os comportamentos são implementados para reagirem sem se aterem à

questões geográficas e espaciais, como geralmente são projetados nas modelagens que utilizam mapas imutáveis do ambiente.

Geralmente ao se utilizar esta abordagem implementa-se comportamentos mais simples, como “seguir-parede”, “ir-em-direção de ponto de luz”, “desviar obstáculo”, e por vezes a simplicidade se torna tão veemente que tudo que se consegue construir são agentes com pouca inteligência, posto que eles não tem um nível cognitivo. Apesar de alguns estudos promissores de agentes dotados de pouca inteligência cooperando para alcançar objetivos globais [72], [11][81], em muitas aplicações se perde muito devido a falta de uma entidade cognitiva.

2.4.3 Abordagem Híbrida

Conforme foi visto, enfoques deliberativo e reativo apresentam vantagens e desvantagens, assim uma proposta que unisse as vantagens de cada abordagem é realmente uma boa direção a ser seguida para a construção de agentes autônomos.

Um dos modos para hibridizar o projeto de um AA é fazendo um mapeamento inicial do ambiente, servindo como base para um primeiro planejamento. Ao executar o plano, caso o robô depare com obstáculos que não constavam no mapa, são ativados comportamentos reativos, de modo a contornar tais empecilhos. Após, este tratamento, um novo plano é feito utilizando o mapa.

Outra maneira de unir reatividade e deliberação é a criação de mapas em tempo de execução. Este processo é feito através da coleta das informações sensoriais que são utilizadas para a construção de mapas locais que posteriormente serão unidos em um mapa global do ambiente [65]. Uma discussão mais aprofundada sobre a questão de quais abordagens utilizar para a construção de agentes autônomos pode ser vista em [3] [6] [7].

2.5 Arquiteturas

As arquiteturas desenvolvidas para o projeto de agentes autônomos baseados em comportamento servem para estruturar os componentes inerentes aos comportamentos e organizar a maneira como eles interagem. Isto passa por objetos que vão desde a percepção e ra-

ciocínio, passando pela topologia entre componentes e culminando em processos de ação com funcionalidades específicas.

Ante os paradigmas da RBC vistos nas subseções anteriores, as arquiteturas propostas geralmente se enquadram em uma destas abordagens, seguindo alguns aspectos como suporte ao paralelismo, modularidade, robustez e orientação ao ambiente [8]. A seguir algumas arquiteturas serão comentadas, considerando as três abordagens da RBC, isto é, reativa, deliberativa e híbrida.

2.5.1 Subsumption Architecture

Rodney Brooks do Massachusetts Institute of Technology (MIT) foi o responsável por um grande avanço da RBC. Em seu artigo *"Elephants Don't Play Chess"*[21], ele despontou para a comunidade com idéias inovadoras de uma arquitetura reativa e organizada hierarquicamente, denominada de Subsumption Architecture.

Esta arquitetura decompõe a inteligência em comportamentos individuais, gerando módulos, que coexistem e cooperam entre si, deixando comportamentos mais complexos emergirem, além disso cada módulo individualmente é capaz de gerar comportamentos.

Existe uma organização em camadas como pode ser visto na Figura 2.1, sendo que cada camada tem uma funcionalidade comportamental frente aos comportamentos e objetivos a serem desempenhados pelo agente.

Comportamentos podem suprimir outros comportamentos com prioridades menores, assim como sinais de estímulos e resposta podem ser suprimidos ou inibidos por outros comportamentos ativos. Convém tecer um comentário sobre a existência de uma relação desta supressão de comportamentos com o Problema da Seleção de Ação, em que um robô deve tomar uma decisão sobre que comportamento emergir, frente a diferentes estímulos oriundos do ambiente, em [33] este problema é fortemente discutido.

Cada camada tem a função de desempenhar de maneira assíncrona uma determinada tarefa, bem como, uma camada de maior nível hierárquico tem a capacidade de inibir a entrada sensória ou suprimir a saída da sua camada imediatamente inferior, afim de garantir prioridade no disparar de algum comportamento.

As camadas são constituídas por um conjunto de máquinas de estado finito extendi-

Figura 2.1: Estrutura da Subsumption Architecture

das[2](AFSM) que possuem um conjunto de registradores e temporizadores conectados a máquinas de estado finito convencionais. A chegada de uma mensagem, ou a expiração de um temporizador pode despertar uma mudança de estado no interior das máquinas de estado finito. Este esquema é representado na figura 2.2.

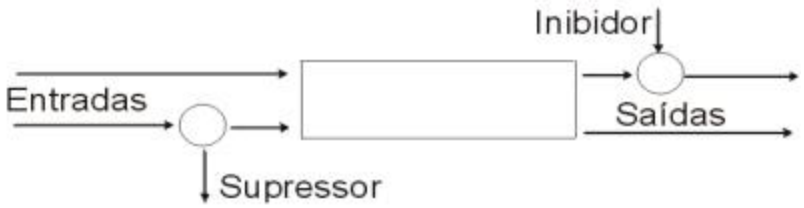

Figura 2.2: Visão detalhada de uma camada da Subsumption Architecture: as entradas podem ser suprimidas e as saídas inibidas por outros comportamentos

Cada comportamento pode agrupar múltiplos processos que geralmente acabam sendo implementados como uma única máquina de estado finito extendida. Podem existir mensagens de transição, supressão e inibição dentro de um comportamento, bem como entre comportamentos.

ALVES descreve o comportamento destas máquinas sucintamente em [29].

> *"Cada máquina de estado finito possui uma quantidade de estados, um ou dois resgistradores internos, um ou dois relógios internos e acesso às máquinas simples que podem fazer cálculos tais como soma de vetores. São ativadas através de mensagens que recebem, e uma mudança de estado ocorre quando chega uma determinada mensagem ou quando o tempo estipulado para este estado expira. Não há memória global compartilhada. As entradas*

[2]do inglês Augmented Finite State Machine

de cada máquina de estado finito podem ser suprimidas e as saídas podem ser inibidas por outras máquinas."

2.5.2 SAUSAGES

Esta é uma arquitetura relacionada com a abordagem deliberativa, pois funciona como espécie de mediador entre o planejamento e execução [44].

Para executar um plano, o robô necessita detectar que aquele é o momento da execução do plano, entretanto por vezes isso é uma tarefa difícil frente a diversidade dos ambientes. Para contornar este problema, mantendo a confiabilidade do sistema como um todo, são desenvolvidos os chamados *Annotated Maps*. Eventos arbitrários como a passagem por uma intersecção em uma estrada são geralmente problemas de difícil monitoramento que são tratados por estes *Annotated Maps* que criam um mapa *a priori* para para a monitoração da posição do veículo. O plano nesta estrutura é essencialmente um sistema de produção posicional, com regras do tipo *"na posição x, execute ação y"*, ao invés de produções do tipo *"Se evento x então execute ação y"*. Segundo o autor, este tipo de representação reduz os recursos necessários para executar uma missão, além de devidamente permitir cenários complexos.

A arquitetura SAUSAGES foi projetada para servir como uma ponte entre o planejamento e a percepção. Esta arquitetura pode ser utilizada para integrar módulos de percepção com uma minimização do planejamento envolvido. Além disso, contém uma ampla facilidade no que tange a utilização com planejadores externos que podem gerar, monitorar e ajustar "planos SAUSAGES" de maneira a criar comportamento inteligente e reativo.

Um plano é composto de unidades semânticas discretas, chamadas de *links* que podem ser vistas como um único passo dentro de um plano. O *link* tem um conjunto de regras de produção associadas a ele, que são adicionadas em um sistema de regras de produção geral como mostra a Figura 2.3.

A estrutura de planos pode se tornar complexa, fazendo com que os *links* se relacionem com um grafo. Além disso, os *links* podem ser sub-divididos em *sub-links* com suas próprias regras de produção. Por exemplo, o *link* atravessar-a-rua, poderia ser composto por *sub-links* como *olhar-para-os-lados*, *verificar-cor-do-semáforo*, *assegurar-se-*

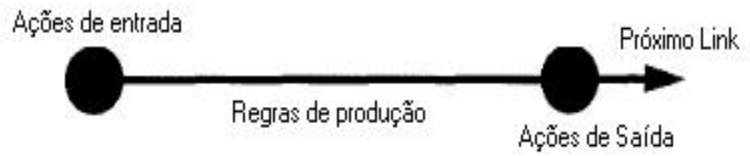

Figura 2.3: Link SAUSAGES

quanto-a-declividade-do-asfalto, etc.

Segundo [44], esta arquitetura mantém flexibilidade tanto para pesquisas em percepção e atuação, utilizando poderosas construções, como os *annotated maps*, para rapidamente gerar sistemas que executam diferentes missões com um mínimo de trabalho em planejadores de alto nível. Além disso, também permite pesquisas em planejamento e interação, pois pode-se utilizar SAUSAGES para interagir com um robô em nível simbólico ao invés de se tratar em níveis mais baixo, o que a acarretaria a necessidade do projetista trabalhar diretamente no âmbito de *pixels* e *motor*.

2.5.3 RAP - Reactive Action Packages

Como exemplo de abordagem híbrida, esta arquitetura fundamentalmente leva em consideração a dificuldade de um plano abranger todas as eventualidades que possam acontecer, dada a diversidade do ambiente. Então como princípio básico esta arquitetura propõe que os planos devem ser formulados de maneira incompleta, isto é, deixando espaço para que os detalhes impostos pela dinâmica ambiental sejam tratados em tempo de execução.

Para tanto, o plano deve ser orientado à tarefas, que em [36] são tratadas como a melhor maneira planejada de atingir determinado objetivo. Assim, uma tarefa é utilizada como índice em tempo de execução, de modo a permitir a busca da decisão (método) adequada a ser tomada em função do objetivo.

Além disso, a estrutura da tarefa também deve conter um teste para determinar quando a tarefa deve ser considerada para execução, este teste contém dois parâmetros: uma verificação se a tarefa é satisfeita na situação atual e uma janela temporal que certifica em que momento a tarefa é relevante. Assim, através do teste de satisfação e da janela temporal juntos, um mecanismo de seleção de tarefas decide se determinada tarefa deve ser considerada para execução. Um RAP acaba sendo uma representação que agrupa e

descreve todas as maneiras conhecidas para realizar uma tarefa em diferentes situações.

Apenas um RAP é necessário para cada tipo de tarefa, pois ele contém os métodos, e descreve os contextos em que eles são apropriados, no entanto podem haver diferentes tarefas instanciadas para um mesmo RAP. A Figura 2.4 ilustra um sistema RAP.

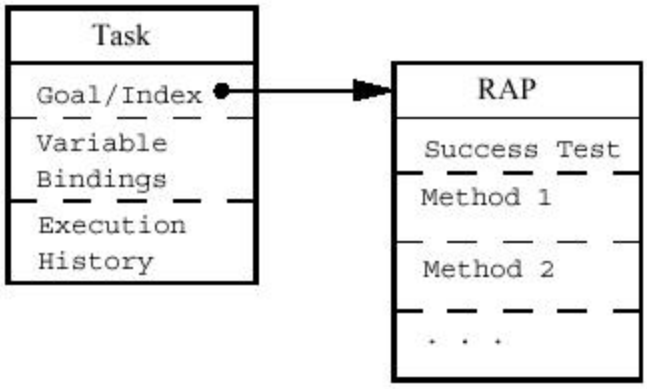

Figura 2.4: Sistema RAP

2.5.4 PiramidNet

No intuito de desenvolver uma arquitetura de robótica baseada em comportamento inspirada biologicamente, de modo a prover autonomia frente a diversidade do ambiente, emergindo inteligência através de padrões comportamentais interagindo em conjunto e de forma modular, Mauro Roisenberg [90][105] desenvolveu a denominada arquitetura PiramidNet.

O cérebro opera de maneira modular, isto é, regiões do cérebro executam funções diferentes. Enumerando algumas destas regiões, temos: o cerebelo como responsável por atividades que envolvem equilíbrio e coordenação; o diencéfalo formado pelo tálamo e hipotálamo, o primeiro como sendo um concentrador e distribuidor das informações destinadas aos hemisférios cerebrais, ao passo que o segundo abastece o sistema endócrino, enviando, por exemplo sinais, que disparam o processo de produção de adrenalina, causando aquela euforia ao assistir uma final de Copa do Mundo, por exemplo.

Os sistemas modulares apresentam uma estrutura hierárquica regida principalmente pela funcionalidade, e podem apresentar dois enfoques: horizontal e vertical.

No primeiro, o desencadeamento dos processos acontece de maneira serial, ou seja, as funções de diferentes módulos organizados hierarquicamente são realizadas em uma mesma camada funcional e as informações processadas podem ser repassadas às camadas hierárquicas posteriores. Roisenberg [90] versa sobre este tipo de enfoque, comentando que:

> *"A estrutura horizontal é encontrada onde os processos são executados em estruturas modulares de neurônios da mesma camada hierárquica. Este estágio pode, por exemplo, ser identificado no sistema primário, onde simples características de imagens visuais como linhas e arcos são representados em neurônios simples que estão na mesma camada. Estas características são combinadas e são representadas através de neurônios em camadas subsequentes para formar então imagens complexas que são interpretadas pelo córtex"*.

Ao passo que na abordagem vertical, aventa-se o paralelismo intrínseco do cérebro ao executar comportamentos distintos simultaneamente, mesmo que estes comportamentos sejam disparados por diferentes regiões funcionais do cérebro. Por exemplo, podemos nos exercitar em uma bicicleta ergométrica, lendo uma revista e tendo um *walkman* junto ao ouvido tocando música, tudo isto mantendo uma sinergia motora e perceptiva adequada.

Outro exemplo seria o fato dos estímulos visuais como forma, cor, movimento e posição serem processados por mecanismos neurais anatomicamente separados, organizados no magno celular e parvo celular. Os processos de informação visual são integrados através de estruturas convergentes de maneira separada em uma posição mais alta dos níveis hierárquicos, formando uma percepção plana e unitária. [17].

Por tudo isto, o princípio do PiramidNet é utilizar redes neurais artificiais organizadas de maneira modular e hierárquica, construindo uma pirâmide que controla o robô, tendo na base desta pirâmide redes neurais diretas que são responsáveis pelos comportamentos mais simples e reativos, explorando a atuação direta no nível efetor.

Na medida que vamos subindo para o topo da pirâmide a complexidade das redes neurais vai aumentando. Para modelar comportamentos mais complexos (como comportamentos de controle) utiliza-se Autômatos Finitos Determinísticos (AFDs) que podem

ser representados e construídos através de redes neurais artificiais recorrentes.

Os níveis hierárquicos não param por aí, pois comportamentos mais complexos precisarão de estruturas mais poderosas e abrangentes que os AFDs, desta forma, redes mais complexas podem ser colocadas em camadas superiores, de modo a simular comportamentos de autômatos de pilha, por exemplo. Veremos isto com mais detalhes no capítulo 6. A Figura 2.5 ilustra a pirâmide da arquitetura PiramidNet.

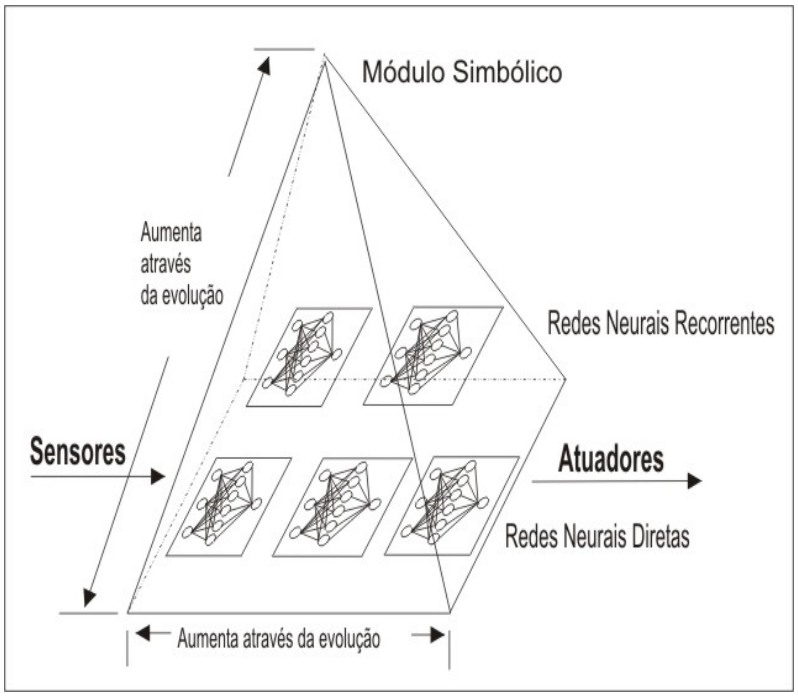

Figura 2.5: Arquitetura PiramidNet

Capítulo 3

Linguagens para Descrição de Comportamentos

"É belo modelar uma estátua e dar-lhe vida, mas é sublime modelar uma inteligência e dar-lhe liberdade"(Vitor Hugo)

3.1 Introdução

Neste capítulo serão abordadas algumas linguagens utilizadas para representação de comportamentos robóticos. Com o advento da explosão da robótica baseada em comportamentos, pesquisadores e fabricantes estão criando diversas formas de representar comportamentos robóticos. Neste ínterim, o objetivo deste capítulo é mostrar a gama de variedades destas respectivas linguagens, isto é, demonstrar que a abrangência para descrição de comportamentos vai desde linguagens baseadas em Lisp, passando por representações que utilizam idéias de programação orientada a objetos e chegando até mesmo a representações no domínio homem-robô.

A escolha destas linguagens que veremos a seguir foi justamente abranger esta gama de variedades, de modo proposital a não comentar sobre diferentes linguagens que utilizem os mesmos conceitos de descrição de comportamentos robóticos.

3.2 Gapps

A linguagem Gapps [51][52] é uma linguagem utilizada para especificar componentes de planejamento orientados a objetivos para robôs móveis.

Existe um compilador Gapps que gera um programa REX [50], o qual é uma linguagem baseada em LISP que será compilado novamente em um circuito digital síncrono, e este será responsável pela implementação propriamente dita do sistema especificado. Esta decomposição horizontal provê uma separação dos níveis de percepção e ação.

A seguir será mostrado um exemplo da implementação de um programa Gapps que tem como objetivo fazer com que o robô segure um martelo e um serrote simultaneamente.

defgoalr (ach (have martelo) (have serrote))

(if (have martelo)

(ach have serrote))

(*if (have serrote)*

(and (maint have serrote)

(ach have martelo))

(if (closer-than martelo serrote)

(ach have martelo)

(ach have serrote))))

No código acima a função *ach* define o objetivo a ser alcançado, ao passo que a função *maint* define um objetivo a ser mantido em um estado particular. Além disso, o predicado *have* serve para certificar que o agente está segurando o objeto.

Além das dificuldades que abordagens baseadas em planos sugerem, como ter que limitar o escopo do ambiente, que já foram tratadas na seção de arquiteturas deliberativas, as linguagens de descrição de comportamentos que utilizam linguagens como Lisp e C++, geralmente implicam em infortúnios como o grande grau de adaptação que o projetista precisa ajustar para adaptar o projeto com a sintaxe da linguagem de programação, o que acarreta um aumento substancial no código fonte, gerando uma tendência para futuros erros no que tange à solução final.

3.3 Robot Schemas (RS)

Esta linguagem se baseia na teoria dos esquemas, na qual esquemas são uma forma de categorizar a percepção sensória como parte fundamental do processo de criação do conhecimento ou experiência. Arkin [8] define um *schema* da seguinte forma

> *"A Schema is the basic unit of behavior from wich complex actions can be constructed: it consists of the knowledge of how to act or perceive as well as the computational process by wich it is enacted"*

Cada esquema possui associado uma descrição comportamental que define como uma instância de determinado *schema* irá se comportar em resposta a uma determinada comunicação.

A linguagem RS [62] provê uma formalização para o comportamento robótico em um nível que possa ser tratado através de uma formalização matemática. Abaixo pode-se ver um exemplo de uma sentença escrita em RS, que trata da movimentação de um robô para uma posição particular.

$$Jmove_{i,\bar{x}}()(x) = [Jpos_i()(x), Jset_{i,\bar{x}}(x)(u), Jmot_i(u)()]^{C,E} \tag{3.1}$$

Esta linguagem apresenta um modelo de computação baseado completamente no domínio de programação do robô, o que traz ao projetista a necessidade de conhecer com profundidade todas as possíveis interações do robô com o ambiente, desta forma a descrição de comportamentos em nível de extrapolação torna-se um percalço de magnitude polinomial a ser superado.

Outro problema desta abordagem é o fato da necessidade de existir um alto grau de sincronismo entre os sinais enviados nos diferentes níveis de instanciações de esquema, isto é, para que as junções de esquema funcionem corretamente é necessário alocar portas de comunicação que trabalhem com mesmos tipos de estruturas em tempo hábil, sendo que efetivar esta diferenciação dos dados sensórios frente a um ambiente do mundo real torna-se uma tarefa bastante penosa e que envolve um custo computacional muito alto.

3.4 Robot Independent Programming Language (RIPE)

Esta linguagem foi desenvolvida pelo Sandia National Laboratories [1]. Caracteriza-se como uma linguagem baseada na programação orientada a objetos, e foi desenvolvida para modelos baseados em planejamento automatizado na programação de robôs [70].

Sua utilização depende de uma poderosa e complexa composição de *hardware* com multiprocessamento hierárquico que emprega processadores distribuídos com funções específicas e pré-definidas.

O enfoque orientado a objetos é o ponto central da RIPE, onde as propriedades inerentes à programação orientada a objetos englobam o próprio sistema robótico e o seu ambiente.

As relações (ações) do robô com os objetos do ambiente compõem o domínio no qual o *software* controlador do robô atua, assim, podem-se criar classes de ações possíveis de serem executadas perante determinadas condições ambientais. Ademais, o ambiente de desenvolvimento para uma implementação utilizando RIPE contém quatro camadas.

A primeira camada é uma camada de baixo nível, contendo os *drivers* de dispositivos que vão desde *drivers* mais simples que consistem de comandos de interface para tarefas como controlar um leitor de código de barras, assim como pode alcançar dispositivos de controle robótico mais sofisticados que exigem um sincronismo complexo.

A camada seguinte lida com controle em tempo real de dispositivos para tarefas como controle de força. Consiste de múltiplos processadores na família Motorola VME-68000, atuando sobre o sistema operacional VxWorks compatível com sistemas UNIX. [113]

Os semáforos oriundos deste sistema operacional permitem acesso imediato aos dados sensórios, e a coordenação dos métodos do robô e dos sensores podem ser executadas em múltiplos processadores. Além disso, a linguagem C++ funciona perfeitamente neste ambiente, o que permite uma maior uniformidade do projeto.

A terceira camada é a responsável pelo controle de todas os dispositivos e atividades do sistema. Sua implementação é feita em uma *workstation* baseada em UNIX. A linguagem C++ é utilizada para a implementação das classes que irão controlar a hierarquia das atividades executadas pelo robô.

[1] http://www.sandia.gov

A escolha da linguagem C++ foi feita por ser uma linguagem já gabaritada e robusta no âmbito das linguagens orientadas a objeto, além do que, ela oferece todas as características necessárias dos conceitos da programação orientada a objetos.

Finalmente, a camada de mais alto nível se preocupa com a programação em nível de tarefas que o robô deve executar, e também executa a modelagem do mundo, planejamento e simulação.

A complexidade de *hardware* envolvida na arquitetura que sustenta a RIPE, aliada ao alto grau de conhecimento da linguagem C++ que o projetista deve ter são as principais desvantagens desta linguagem.

3.5 Representação de Comportamento no Domínio Homem-Robô

Monica Nicolescu e Maja Mataric em [76] apresentam uma abordagem focada na aprendizagem de um robô através de um "professor humano", onde os agentes devem ter a característica de aprender através da interação com humanos ou outros agentes, levando em consideração diversos tipos de interação.

A interação entre o robô aprendiz e o instrutor humano se baseia na premissa que o humano pode atuar como um professor ou como um colaborador para auxiliar no aprendizado do robô. Desta forma, o objetivo geral é fazer com que o robô efetue tarefas através da experiência que ele adquiriu através da interação com o instrutor humano.

Para descrever os comportamentos do robô utiliza-se uma arquitetura baseada em comportamentos que permite a construção das tarefas em forma de redes de comportamentos.

Estas redes se baseiam em *links* que conectam os nós de comportamentos representando dependências de requisitos e conseqüências, de forma que o disparo de um comportamento não depende apenas de seus próprios pré-requisitos, isto é, seus estados ambientais particulares, mas depende também das conseqüências entre seus precursores relevantes.

As peculiaridades da rede de comportamentos geram uma relação causal de pré-condições

seqüenciais que podem ser classificadas em três tipos durante a execução da rede:

Pré-Condições Permanentes: São aquelas que devem ser encontradas durante toda a execução do comportamento. Caso a condição esteja em vigência e por um algum motivo ela seja desabilitada, automaticamente o comportamento é desativado.

Pré-Condições Capacitadoras: Devem estar presentes imediatamente antes da ativação de um comportamento, seu estado pode mudar durante a execução do comportamento, sem influenciar na ativação do comportamento.

Pré-Condições de Ordenação: Estas pré-condições garantem o sincronismo da rede de comportamentos, devendo ser encontradas em algum momento anterior à execução do comportamento.

A figura 3.1 ilustra a rede de comportamentos desta arquitetura, enfocando as condições seqüenciais.

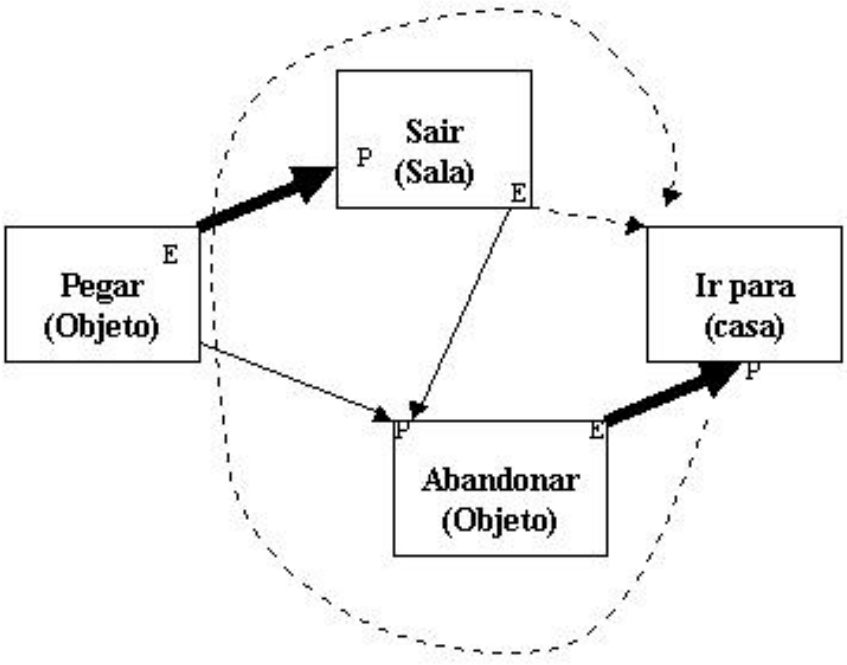

Figura 3.1: Rede de Comportamentos. *P* são pré-condições de entrada e *E* são efeitos da saída

Esta abordagem tem como sua principal desvantagem a dependência do ser humano

para fazer com que o robô aprenda a efetuar tarefas, isto pode ser um grande empecilho em aplicações críticas como viagens espaciais e ambientes hostis como campos minados.

Além disso, o robô não é capaz de inferir novos comportamentos que não foram apresentados por um ser humano no momento do treinamento, o que faz esta estrutura atuar em uma espécie de arquitetura deliberativa por imitação.

3.6 Behavior Language

A linguagem Behavior Language (BL) pode ser denominada como a linguagem oficial para o desenvolvimento de aplicações que utilizam a Subsumption Achitecture. Esta linguagem emergiu dos estudos de Pattie Maes [64] e foi consolidada pelo criador da Subsumption Achitecture, Rodney Brooks em [21].

A BL utiliza uma sintaxe baseada na linguagem Lisp para descrever comportamentos robóticos. Ela foi desenvolvida como uma maneira de englobar as máquinas de estado finito aumentadas da *Subsumption Achitecture* em unidades mais manuseáveis, possibilitando que todas as unidades pudessem ser ativadas ou desativadas de maneira seletiva.

Geralmente os comportamentos robóticos são desenvolvidos pelo projetista de forma que a emergência deles, frente a interação com o ambiente, torne possível a efetivação de objetivos pré-definidos. Neste interim, os objetivos definidos através da BL ficam implícitos na descrição dos comportamentos robóticos, de modo que a funcionalidade do agente aflore de maneira “natural”.

Conseqüentemente, as AFSM não são especificadas diretamente, sendo produto de conjunto de regras de tempo real que são compiladas uma à uma dentro destas máquinas. Finalmente, os comportamentos são gerados através do compartilhamento dos registradores e temporizadores intrínsecos às AFSM que produzem um único conjunto de regras, que nada mais são que os próprios comportamentos.

As regras em tempo real são o apanágio principal da BL, sendo que elas podem agir isoladas ou agrupadas dentro dos comportamentos. Assim, estas condições em tempo real são testadas e quando encontrado um valor verdadeiro para a regra, uma parcela de código baseado em Lisp é ativado.

Convém ressaltar que as regras lidam com constantes e variáveis armazenadas em

registradores, geralmente com palavras de no máximo 8 bits, além disso, existem temporizadores que podem ser monitorados e disparados. Outrossim, não existe um enfoque procedural, de modo que toda a abstração é dada através de macros, regras e comportamentos.

Para especificar uma regra de tempo real são utilizadas duas cláusulas: *whenever* e *exclusive*.

A sintaxe geral para a cláusula *whenever* é a seguinte:

(**whenever** *condition &rest body-forms*)

Para a execução de uma regra um processo é alocado de modo a garantir esta execução. O processo inicia em um estado de espera, então não importando o momento (*whenever*) que a condição de disparo se torna verdadeira, a lista de *body-forms* é avaliada de maneira seqüencial, finalmente o processo volta ao seu estado de espera até que a condição seja novamente satisfeita. Abaixo serão listadas as possíveis condições que a cláusula *whenever* pode assumir:

t: Garante que em um determinado período de tempo o corpo da cláusula *whenever* será incondicionalmente avaliado, este período de tempo geralmente está associado à característica do próprio sistema de subsumpção.

Monostable: A condição será verdadeira para a duração do disparo de um dado temporizador.

(not monostable): A condição é verdadeira apenas enquanto o temporizador não está disparado.

(delay t): A diferença para a condição que utiliza o parâmetro *t* é que aqui podemos escolher o valor do período em que a cláusula *whenever* será avaliada, não dependendo necessariamente das características do sistema.

(received?register): Esta condição é satisfeita se alguma mensagem foi depositada em um denominado registrador desde o começo do estado de espera da cláusula *whenever*.

(and—or &rest forms): Formam um conjunto restrito de combinações lógicas das formas descritas acima.

É interessante observar que todas as condições vistas referem-se explícita ou implicitamente a elementos temporizadores. Todavia, existe uma forte concepção que estes temporizadores não estão relacionados entre si, ou seja, eles são totalmente independentes.

Já as regras que utilizam a cláusula ***exclusive*** tem a possibilidade de simultaneamente monitorar muitas condições, de modo a executar exclusivamente a primeira a se tornar verdadeira, ignorando qualquer outra condição que possa ocorrer em um mesmo tempo de execução.

Sua sintaxe geral é (*exclusive &rest whenever-forms*), onde *whenever-forms* é uma coleção de regras do tipo *whenever*.

Conforme já foi mencionado os comportamentos são definidos como regras de tempo real isoladas ou como grupos destas regras. Para comportamentos definidos por uma única regra, como sintaxe de definição tem-se o seguinte:

(**defmachine** *name declarations rule*), onde:

- O argumento *name* é o nome de uma única AFSM que implementa a regra especificada.
- A cláusula *declarations* serve para especificar períodos monoestáveis e valores iniciais para os registradores.
- rule é a regra propriamente dita que pode ser definida através das cláusulas *defbehavior* e *definterface*

As cláusulas *defbehavior* e *definterface* possuem, respectivamente, as seguintes sintaxes (**defbehavior** name &key inputs outputs decls processes) e (**definterface** name &key inputs outputs decls processes), onde

- *name:* é o nome do comportamento.
- *inputs:*Uma lista de registradores que estão disponíveis como entradas de fora do comportamento.

- *outputs:* Lista de portas de saídas que podem ser conectadas com entidades externas.
- *decls:* São as declarações tendo a mesma sintaxe da cláusula **defmachine**
- *processes:* Lista de regras em tempo real que possuem a mesma sintaxe daquelas descritas anteriormente; podem ser escritas como *whenever*, *exclusive* e até mesmo *defmachine*.

Finalmente para conectar máquinas de estado finito aumentadas e comportamentos é utilizada a cláusula *connect*. A forma geral da sintaxe desta cláusula é dada a seguir:

(**connect** *source dest1 &rest more-dests*)

O parâmetro *source* significa uma porta que pode ser um registrador ou a saída de uma máquina de estados finitos aumentada especificadas com a cláusula *defmachine*, assim como pode ser uma entrada ou saída de um comportamento.

Desta forma a sentença indica que para toda saída de uma porta *source* será entregue uma cópia para os destinos especificados como *dest1* ou para os destinos da lista *more-dests*. Além disso, se a porta de destino for parte de um comportamento, uma cópia da mensagem irá ser entregue para cada regra ou máquina de estados finitos aumentada que referencia à determinada entrada.

É importante ressaltar que a cláusula *connect* também é utilizada para implementar supressões e inibições, que são duas das principais características da *Subsumption Architecture*.

A *Behavior Language* é uma proveitosa ferramenta de programação, no entanto fornece pouco suporte formal para os comportamentos que os robôs descrevem. Além disso, devido a falta de restrições no que tange a comunicação entre as camadas superiores com as inferiores, se torna difícil a especificação e a depuração dos comportamentos.

Novamente surge a desvantagem da utilização do conhecimento por parte do projetista de uma linguagem de programação específica, que neste contexto é a linguagem Lisp, o que sugere um alto grau de adaptação que o projetista precisa estar ajustando para efetuar o projeto, acarretando um aumento substancial no código fonte, o que pode tender para futuros erros na solução final.

Finalmente, esta linguagem se restringe apenas a implementações que se baseiam na *Subsumption Architecture*, o que limita as implementações de agentes autônomos baseados em comportamento a apenas uma arquitetura. Isto traz à tona a passividade de problemas como a geração “manual” dos elementos temporizadores das AFSMs antes da implementação, o que pode acarretar diversos erros nos objetivos do agente, fazendo até mesmo que ele efetue uma tarefa indeterminadamente ou em outro extremo, para temporizadores de baixo valor, pode acontecer de o robô concluir uma tarefa antes do tempo, não chegando assim a um objetivo almejado. Outro problema desta arquitetura é a adaptação para novos comportamentos, uma vez que existe uma dependência hierárquica entre os comportamentos implementados.

Capítulo 4

Linguagens de Chomsky e Comportamentos Robóticos

"Como o xadrez, o jogo de damas e outros, o pensamento consiste em embaralhar peças sem significado de acordo com regras definidas. Dessa manipulação de símbolos, a mente surge sinergicamente como uma imagem num vídeo surge do embaralhamento de 1 e 0 de um computador"(Jerry Fodor)

4.1 Introdução

O lingüista americano Noam Chomsky em seu artigo *On Certain Formal Properties of Grammars*[25] introduziu as idéias da chamada Hierarquia de Chomsky.

Chomsky acreditava que as sentenças de uma linguagem como a lingua inglesa seguiam a estrutura desta hierarquia, mostrando a expressividade das gramáticas, bem como o poderio dos autômatos reconhecedores.[60]

Esta teoria classificava as gramáticas como sendo de quatro tipos: Gramáticas Irrestritas ou gramática de nível 0, Gramáticas Sensíveis ao Contexto (GSC) ou gramáticas de nível 1, Gramáticas Livres de Contexto (GLC) ou gramáticas de nível 2 e finalmente as chamadas gramáticas regulares[1] (GR) ou gramáticas de nível 3. Este aninhamento gramatical induz a classificação na mesma hierarquia das linguagens geradas por estas

[1] As gramáticas regulares neste trabalho serão representadas pelas propridades de geração das expressões regulares, conforme visto na seção 4.4

gramáticas[102]. Desta forma, teremos as Linguagens Regulares (LR) (nível 3), Livres de Contexto (LLC) (nível 2), Sensíveis ao Contexto (LSC) (nível 1) e Linguagens Enumeráveis Recursivamente (LER) (nível 0).

O enquadramento de uma linguagem em um nível da Hierarquia de Chomsky é regido pelo formato das regras de produção das gramáticas.

Para cada nível da hierarquia de Chomsky existe um dispositivo capaz de reconhecer se uma dada cadeia de símbolos pertence ou não a determinada linguagem. Estes dispositivos são chamados de reconhecedores, os quais são: autômatos finitos (AF), autômatos de pilha (AP), autômatos linearmente limitados (ALL) e máquinas de Turing em ordem decrescente dos níveis hierárquicos de Chomsky. A tabela 4.1 mostra um sumário das relações entre as linguagens, com suas respectivas gramáticas geradoras e reconhecedores.

Linguagem	**Gramática**	**Reconhecedor**
Nível 0: LER	Gramáticas irrestritas	Máquina de Turing
Nível 1: LSC	Gramáticas sensíveis ao contexto	Autômato Linearmente Limitado
Nível 2: LLC	Gramáticas livres de contexto	Autômatos de pilha
Nível 3: LR	Expressões regulares	Autômatos Finitos

Tabela 4.1: Relação entre linguagens, gramáticas e reconhecedores da hierarquia de Chomsky

Pelo fato das linguagens serem classificadas na hierarquia de Chomsky de acordo com as limitações impostas nas regras de produção das gramáticas, ao se descer na hierarquia de Chomsky, uma maior flexibilidade é provida por parte das gramáticas, de modo que toda LR é uma LLC, toda LLC é uma LSC, e toda LSC é uma linguagem de nível 0. A figura 4.1 ilustra esta idiossincrasia.

Neste trabalho não iremos abordar a classe de linguagens enumeráveis recursivamente que são estritamente do nível 0 como modelos para descrição de comportamentos robóticos. Esta escolha se dá pelo fato de seu autômato reconhecedor, máquina de Turing, não poder garantir que uma linguagem de nível 0 será ou não decidida, isto é, se uma configuração de aceitação ou de rejeição será de fato encontrada.

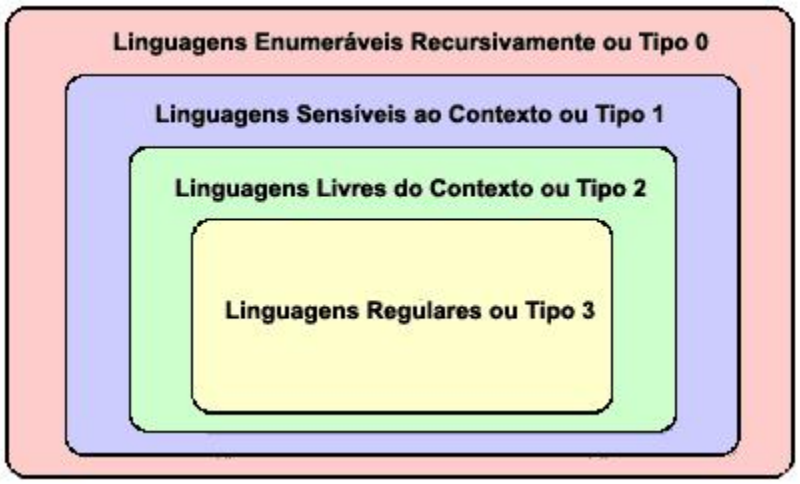

Figura 4.1: Relação entre as linguagens na Hierarquia de Chomsky

Um exemplo disso seria a simples operação de decidir se uma cadeia *w* pertence ou não à uma dada linguagem *L*. Caso $w \notin L$ e se *L* for uma linguagem exclusiva do tipo 0, nunca teremos a garantia de ter esperado tempo suficiente para uma resposta (parada da máquina), dado o tamanho infinito da fita de leitura/gravação que a máquina de Turing possui. Para mais detalhes sobre este assunto, recomenda-se a leitura de [60][102].

Transcendendo o exposto para a área da robótica, seria como imaginar um AA que segue um comportamento que necessitasse de uma máquina de Turing para descrevê-lo, caso o comportamento descrito não estivesse contido na linguagem proposta, este AA iria ficar indefinidamente esperando a resposta de que ação efetivar.

Defende-se então neste trabalho que as linguagens de níveis 1, 2 e 3 da hierarquia de Chomsky, com seus respectivos autômatos, são capazes de modelar comportamentos robóticos, sejam eles simples ou complexos.

Ao tratar de comportamentos englobados pela abordagem reativa da robótica baseada em comportamentos, os autômatos finitos são suficientemente capazes de modelar esta classe de comportamentos, pois neste nível os comportamentos se baseiam apenas no estado atual e na informação sensória captada do ambiente. Além disso, comportamentos com alguma complexidade maior podem ainda ser1 implementados por autômatos finitos extendidos, com temporizadores e registradores, como engloba a *Behavior Language* de Brooks. [21]

No entanto, quando se pensa em hibridizar ou mesmo utilizar uma arquitetura base-

ada em planos, ou seja, fazer emergir comportamentos mais complexos que envolvam tarefas como planejamento de trajetórias e mapeamento de ambientes, o AA cria mapas em nível simbólico, onde a memorização de símbolos, que podem ser marcos[2] oriundos do ambiente torna-se necessária. Neste contexto, precisamos de autômatos que suportem memorização destes dados, bem como tornar possível que estes marcos sejam passo fundamental para os comportamentos emergidos pelo robô. Desta forma, estes comportamentos estariam contidos nos níveis 1 e 2 da hierarquia de Chomsky.

Segundo [68], comportamentos podem ser projetados em variados tipos de implementação. Em geral eles são representados em níveis mais altos do que as próprias ações atômicas dos atuadores, como ”ir-adiante-por-um-incremento X”, ”virar-ângulo-teta”. Desta forma, comportamentos clássicos na literatura são: ”achar-objeto”, ”recarregar-bateria”, ”evitar-colisões”, ”pegar-objeto”, ”evitar-luz”, ”agregar-se com grupo”, ”seguir-parede”, etc.

Neste trabalho a filosofia de descrição de comportamentos em nível mais alto geralmente será seguida, no entanto, por vezes será necessária a decomposição dos comportamentos em níveis de abstração menor. Esta abordagem de decomposição permite a abstração dos detalhes de implementação e naturalmente sugere uma filosofia de projeto *top-down*. Iniciando no nível mais alto, definindo o comportamento global, especificando um coleção de comportamentos mais simples, específicos e concretos, os quais serão utilizados para gerar o comportamento mais complexo desejado [63].

No restante deste capítulo, os três níveis supracitados da hierarquia de Chomsky serão tratados e algumas definições importantes do âmbito das linguagens formais serão brevemente apresentadas, surgindo uma adequação destes conceitos para meandros relacionados aos agentes, e finalmente os comportamentos robóticos serão modelados em termos dos componentes da hierarquia, mostrando a viabilidade da hierarquia de Chomsky e seus respectivos autômatos na modelagem de comportamentos robóticos.

[2] do inglês *landmarks*

4.2 Terminologia Básica

Nas próximas subseções serão brevemente introduzidos alguns conceitos básicos relativos ao estudo da composição das linguagens formais.

4.2.1 Alfabeto

Um Alfabeto é definido como um conjunto finito de símbolos. Por exemplo, o alfabeto da língua portuguesa é V = {a, b, c, d..., z}. No mesmo contexto, o alfabeto binário é representado por {0,1}

4.2.2 Cadeia

Uma cadeia ou string em um alfabeto é uma seqüencia finita de símbolos, com os símbolos justapostos, desta forma *paysandu* é uma cadeia no alfabeto {a, b, ..., z}.

O **comprimento** de uma cadeia é computado através da seqüência de símbolos que a forma. O tamanho de uma cadeia w é denotado por $\|w\|$. Por exemplo, $\|aabb\|$ = 4. A cadeia vazia (não contém símbolos) é denotada por *e*, tendo tamanho igual a 0, logo $\|e\|$ = 0.

Convém ressaltar a existência da chamada *cadeia elementar*, que são as cadeias de comprimento unário, estas cadeias são formadas por um único símbolo, logo, um símbolo por si só, se contido em um alfabeto, representa uma cadeia.

A **Concatenação** de duas cadeias é a cadeia formada pela escrita da primeira cadeia seguida da segunda, sem nenhum espaço no meio. Por exemplo, a concatenação da cadeia x= *compor* com a cadeia y=*tamento* é dada por w=xy ou $w = x \circ y$, isto resultaria em w=*comportamento*.

4.2.3 Fechamento

Dá-se o nome de *fechamento recursivo e transitivo* do conjunto $\sum$ ao conjunto $\sum^*$ formado por todas as cadeias, de qualquer comprimento, que se possa construir com os elementos de $\sum$. Este conjunto inclui a cadeia vazia *e*.

Desta forma também surge o *fechamento transitivo* que é o conjunto $\sum^+$ formado retirando-se a cadeia vazia do conjunto $\sum^*$:$\sum^+ = \sum^+ \cup \{e\}$, onde $e \notin \sum^+$.

4.2.4 Estrela de Kleene

A estrela de Kleene de uma linguagem *L* (vide definição 4.2.5), denotado por L^* é o conjunto de todas as cadeias obtidas pela concatenação de zero ou mais cadeias de *L*.

$L^* = \{w \in \sum^* : w = w_1 \circ \ldots \circ w_k$ para algum $k \geq 0$, e algum $w_1, \ldots, w_k \in L\}$

4.2.5 Linguagem

Uma linguagem em um alfabeto é um conjunto de cadeias de símbolos tomados de algum alfabeto. Isto é, uma linguagem sobre o alfabeto $\sum$ é um subconjunto de $\sum^*$. Esta definição não considera os mecanismos formadores da linguagem, mas apenas a sua extensão. Assim, por exemplo, o conjunto de sentenças válidas da língua portuguesa poderia ser definido como um subconjunto de $\{$a, b, c,.., z$\}^+$. Uma linguagem é finita se suas sentenças formam um conjunto finito. Caso contrário, a linguagem é infinita. Uma linguagem infinita precisa ser definida através de uma representação finita. Por exemplo, a linguagem dos números naturais menores que 10 é finita, e pode ser representada literalmente como, L = $\{$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$\}$. Já a linguagem dos números naturais como um todo não é uma linguagem finita, pois existem uma quantidade infinita de números naturais. Porém, existem mecanismos, como as gramáticas, que permitem expressar esta e outras linguagens infinitas através de representações finitas. [18]

4.2.6 Gramática

É um dispositivo de geração de linguagens que pode ser definida formalmente pela quádrupla G=(*V*,$\sum$,*P*,*S*), onde:

- *V* representa o vocabulário da gramática que corresponde ao conjunto de todos os elementos simbólicos dos quais a gramática se vale para definir as leis de formação das sentenças da linguagem. Os elementos de $V - \sum$ também são chamados conjunto de não-terminais ou de variáveis. Estes elementos não pertencem ao alfabeto,

e são utilizados nas gramáticas para definir construções auxiliares ou intermediárias na formação das sentenças.

- $\sum$ Representa alguns dos elementos de V que são exatamente os símbolos terminais dos quais as sentenças da linguagem são constituídas.

- *P* representa o conjunto de todas as leis de formação utilizadas pela gramática para definir a linguagem. Assim, cada construção parcial, representada por um não-terminal, é definida como um conjunto de regras de formação relativas à definição do não-terminal a ela referente. A cada uma destas regras de formação que compõem o conjunto *P* dá-se o nome de produção da gramática. Sendo que cada produção de *P* tem a forma:

$$\alpha \rightarrow \beta$$

  Onde, α é uma cadeia contendo no mínimo um não-terminal, ou seja, $\alpha \in V^*NV^*$, sendo que *N* representa um conjunto de não-terminais. E β é uma cadeia, eventualmente vazia, de terminais e não-terminais, levando em consideração que $\beta \in V^*$.

- *S* é um elemento de V que dá início ao processo de geração de sentenças. S é dito como o *símbolo inicial* da gramática

Uma gramática é um instrumento formal capaz de construir (gerar) conjuntos de cadeias de uma determinada linguagem. Além disso, são instrumentos que permitem definir, de maneira formal e sistemática, uma representação finita para linguagens infinitas.

Ademais, as gramáticas também podem ser vistas sob a ótica de *sistemas de substituição*, nos quais as produções indicam as substituições possíveis para os não-terminais, de modo que, quando todos estes não-terminais forem eliminados, o resultado será uma sentença da linguagem.

4.3 Terminologia Básica Aplicada aos AAs

De posse da terminologia básica proposta na seção anterior, torna-se necessário dar um enfoque a aplicação desta terminologia para o campo dos AAs.

4.3.1 Alfabeto

O conceito de alfabeto quando aplicado aos AAs se torna bastante extenso, pois ele pode ser: os símbolos que representam a captação sensória do agente, geralmente conjunto de marcos; podem representar os estados dos agentes no autômato de pilha; quando referido à pilha do AP, o alfabeto pode ser considerado como os símbolos utilizados pela memória de trabalho do AA, etc.

4.3.2 Cadeia

A idéia de símbolos concatenados formando cadeias pode ser considerada como a seqüencia de símbolos que o agente recebe do ambiente pelo seu aparato sensório, bem como pode ser considerada uma seqüencia de ações que levam a um comportamento global, como seguir o caminho $< norte >< norte >< leste >< sul >< norte >< oeste >< sul >< sul >$, ou seja ir e voltar pelo mesmo caminho. Na subseção 4.5.2, um exemplo prático deste tipo de comportamento será comentado.

4.3.3 Fechamento e Estrela de Kleene

O *fechamento recursivo e transitivo* de $\sum$ pode ser o conjunto de todos os possíveis valores a serem obtidos pelos sensores, ou a possível sequência de ações que o robô pode descrever para emergir um comportamento. A inclusão da cadeia vazia *e* se faz necessária, pois o robô pode, em algum momento, não estar captando nenhum símbolo através dos seus sensores, da mesma maneira que pode não emergir nenhum comportamento possível.

Já o *fechamento transitivo* de $\sum^*$:$\sum^+ = \sum^+ \cup \{e\}$, onde $e \notin \sum^+$ não permite que o robô fique sem receber nada em seu sensores, bem como pode obrigar que os atuadores efetuem a implementação de uma ação, pois a cadeia vazia *e* não é permitida.

A *estrela de Kleene* de uma linguagem L, é dada por $L^* = \{w \in \sum^* : w = w_1 \circ \ldots \circ w_k$ para algum $k \geq 0$, e algum $w_1, \ldots, w_k \in L\}$.

Sendo que w seria a junção de uma ou mais ações $w = w_1 \circ \ldots \circ w_k$ que formam a emergência de algum comportamento pelo AA.

4.3.4 Linguagem

No âmbito dos AAs, a linguagem é a representação de comportamentos propriamente dita, isto é, a sequência de ações que um AA deve efetivar para a emergência de um comportamento. Exemplificando, se o comportamento do robô for representado pela linguagem $L = wcw^r$, informalmente significaria dizer a ele: "*vá e volte pelo mesmo caminho assim que encontrar um referencial* c".

4.3.5 Gramática

A gramática no contexto dos AAs pode ser considerada o dispositivo de geração de comportamentos, pois pode-se considerar que a entidade cognitiva do robô constitui um conjunto de regras de formação *P*. Ou seja, para uma dada produção $\alpha \rightarrow \beta$ a entidade cognitiva do AA faz um mapeamento de dados oriundos do seu aparato sensório para a execução através de seus atuadores em alguma ação que eventualmente pode estar relacionada com os seus objetivos.

4.4 Nível 3 - Linguagens Regulares

4.4.1 Expressões Regulares

As expressões regulares (ER) são uma forma de representar uma linguagem apenas empregando os símbolos de união ($\cup$), concatenação ($\circ$) e estrela de Kleene (*), ou seja, não necessita de utilização de símbolos não-terminais.

Algumas considerações podem ser feitas sobre as expressões regulares:

- Um terminal *x* forma uma ER, e representa o conjunto que contém apenas a cadeia formada pelo terminal *x* isolado.
- O símbolo *e* forma uma ER, e representa o conjunto que contém apenas a cadeia vazia.
- Se α e β são expressões regulares, a concatenação ($\alpha \circ \beta$) também é considerada uma ER.

- Se α e β são ER, então ($\alpha \cup \beta$) também será uma ER.
- Se α é uma ER, então o fechamento recursivo e transitivo α^* também é considerado um ER.

As ERs formam uma poderosa notação para a definição de linguagens de nível 3, utilizando alguns mecanismos de geração.

Definição 4.1. A relação entre ERs e as linguagens que elas representam é estabelecida por uma função *L*, tal que, se α é uma expressão regular qualquer, então $L(\alpha)$ é uma linguagem representada por α . Isto significa que *L* é uma função de strings para linguagens.

Ademais, a função *L* é definida na seqüencia

1. $L(\oslash) = \emptyset$ e $L(a) = \{a\}$ para cada $a \in \sum$
2. Se α e β são expressões regulares, então $L((\alpha\beta)) = L(\alpha)L(\beta)$
3. Se α e β são expressões regulares, então $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$
4. Se α é uma expressão regular, então $L(\alpha^*) = L(\alpha)^*$

A classe de **linguagens regulares** sobre um alfabeto $\sum$ é definida como consistindo de todas as linguagens *L*, tal que $L = L(\alpha)$ para alguma expressão regular α sobre $\sum$. Isto é, todas as linguagens regulares podem ser descritas como expressões regulares.

4.4.2 Autômatos Finitos

Os dispositivos capazes de reconhecer se uma dada cadeia é ou não uma linguagem regular são conhecidos como autômatos finitos que podem ser de dois tipos: Autômato Finito Determinístico (AFD) e Autômato Finito Não Deteminístico (AFND). Este estudo versará apenas sobre os AFDs, visto que todo AFND pode ser transformado em um AFD [76].

Um autômato finito pode ser visto como uma máquina composta, basicamente pelos seguintes componentes:

Fita de entrada: É dividida em células, sendo que cada célula contém um símbolo que irá ser analisado por um dispositivo; é finita e não existe a possibilidade de se gravar sobre ela; além disso, no estado inicial a palavra a ser processada ocupa toda a fita.

Controle finito e Cabeça de leitura: O controle finito reflete o estado corrente da máquina, possuindo um número pré-definido de estados. A unidade de controle finito acessa a fita de entrada por meio de uma cabeça de leitura (figura 4.2) que acessa o símbolo de uma célula de cada vez, para depois mover-se na fita de entrada, sempre da esquerda para a direita.

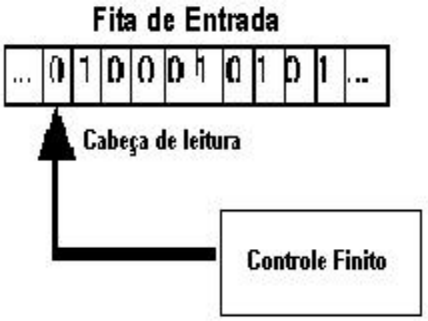

Figura 4.2: Autômato Finito

A leitura do símbolo, aliada ao movimento da cabeça de leitura da esquerda para a direita, estará sempre alterando o estado da unidade de controle finito, conseqüentemente, a cabeça de leitura alcançará o ponto final da fita de entrada. Neste momento o autômato irá aceitar ou recusar o que leu, embasado no estado em que ele está ao final da computação. Se este estado for um dos que foram definidos como um dos conjuntos de estados finais, então ele reconhece e aceita a cadeia de entrada.

O principal apanágio dos AFDs é que a transição de estados depende apenas do seu estado atual e do símbolo que acabou de ser lido na fita de entrada.

Definição 4.2. Um AFD é definido como um quíntuplo $M = (K, \sum, \delta, s, F)$, onde

K é o conjunto finito de estados;

$\sum$ é um alfabeto;

$s \in K$ é o estado inicial;

$F \subseteq K$ é o conjunto de estados finais, e

δ uma função de transição, sendo função de K x $\sum$ para K

Desta forma, para a linguagem regular L=$\{w \in \{a,b\}^*\}$: *w* tem um número par de b's, teríamos o AFD $M = (K, \sum, \delta, s, F)$, onde

$$\text{K=\{q0,q1\}}$$
$$\sum = \{a, b)\}$$
$$s = q_0$$
$$F = \{q_0\}$$

A função de transição de estados δ é dada pela tabela 4.2

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

Tabela 4.2: Tabela de transição de estados

Assim, caso seja apresentada a cadeia *aabba*, teríamos as seguintes transições.

$$\begin{aligned}(q_0, aabba) &\vdash_M (q_0, abba)\\ &\vdash_M (q_0, bba)\\ &\vdash_M (q_1, ba)\\ &\vdash_M (q_0, a)\\ &\vdash_M (q_0, e)\end{aligned}$$

Podemos também representar este AFD pelo seu diagrama de estados que é uma representação que consiste de um grafo dirigido, onde os vértices representam os estados, sendo que cada arco direcionado representa a transição de um estado para outro, tal que $\delta(q, a) = q^{'}$. Estados finais são indicados por um círculo duplo, bem como o estado

inicial é representado por um $>$. A figura 4.3 ilustra o diagrama de estados para o AFD em questão.

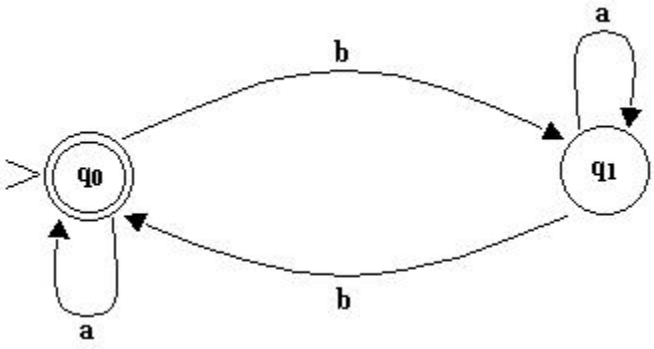

Figura 4.3: Diagrama de Estados

4.4.3 Descrição de comportamento robótico utilizando AFD

A representação de comportamentos robóticos utilizando autômatos finitos é bastante difundida na literatura de AAs, sua utilização pode ser observada em trabalhos como [8], [73], [21], [89] e [33].

Para comportamentos robóticos pode-se imaginar a situação em que um robô, equipado com um sensor de luz e outro de toque, deve vagar por uma sala com o intuito de encontrar e pegar latas de refrigerante perdidas e levar a lixeiras apropriadas que também estão dispostas na mesma sala.

As latas de refrigerante foram colocadas com rótulos vermelhos e os depósitos de lixo foram rotulados com uma faixa azul, de tal forma que pudesse ocupar toda a dimensão da lixeira.

Inicialmente o robô deve vagar pelo ambiente afim de encontrar uma lata de refrigerante, e ao "avistá-la" através dos seus sensores de luz, o comportamento de mover-se em direção à lata deve emergir, sendo que ao tocar na lata seu sensor de toque irá acusar que ele deve pegar a lata.

Ao pegar a lata, o robô deve voltar a vagar, agora em busca de um depósito de lixo, sendo que ao sentir este depósito, ou seja, uma luz azul sendo captada pelos seus sensores, o robô deve ir na direção do depósito até que o sensor de toque seja ativado, quando isto acontecer o robô deve jogar a lata fora e sua tarefa se dá por completa.

Os comportamentos deste robô podem ser representados pelo autômato $M = (K, \sum, \delta, s, F)$, onde

K={vagar-por-lata, mover-para-lata, pegar-lata, vagar-por-lixeira, mover-para-lixeira, jogar-lata-fora}

$\sum$={SL=V, SL≠V, ST=0, ST=1, SL≠Az, SL=Az}

$s = vagar - por - lata$

$F = jogar - lata - fora$

A função δ é mostrada na tabela 4.3.

q	σ	$\delta(q, \sigma)$
vagar por lata	SL≠V	vagar por lata
vagar por lata	SL=V	mover para lata
mover para lata	ST=0	mover para lata
mover para lata	ST=1	pegar lata
pegar lata	SL=Az	mover-se para lixeira
pegar lata	SL≠Az	vagar por lixeira
vagar por lixeira	SL=Az	mover-se para lixeira
vagar por lixeira	SL≠Az	vagar por lixeira
mover-se para lixeira	ST=0	mover-se para lixeira
mover-se para lixeira	ST=1	jogar lata fora

Tabela 4.3: Tabela de transição de estados para comportamentos do robô catador de latinhas.

Na tabela 4.3, SL significa a informação obtida pelo sensor de luz, que para a função de transição pode ter quatro instâncias: vermelho(SL=V), diferente de vermelho (SL≠V), azul(SL=Az) e diferente de azul(SL≠Az). Já o sensor de toque foi convencionado que quando ele operar com o valor 0 (ST=0), ele permanece sem ativação, ao passo que quando for ativados seu status passa a ser ST=1. Este AFD é representado graficamente através do diagrama de estados ilustrado na figura 4.4

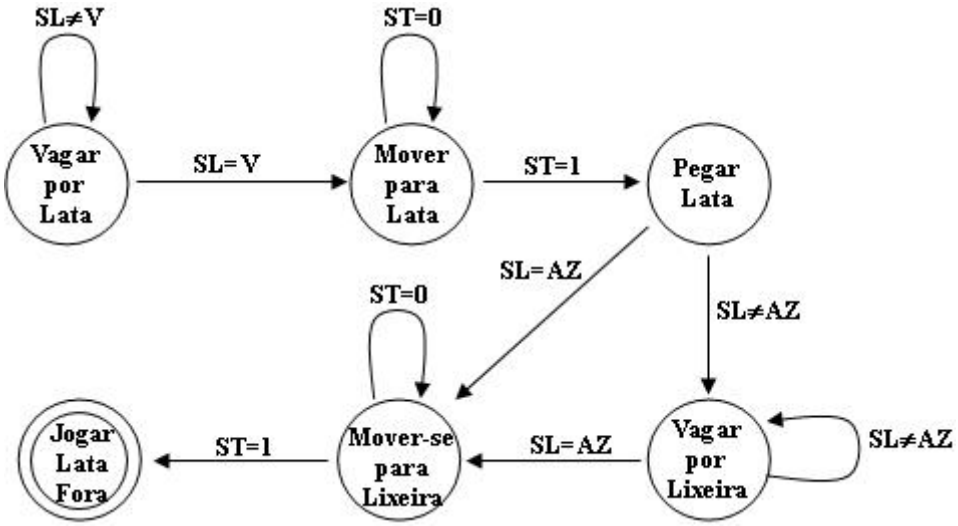

Figura 4.4: AFD que modela comportamentos do robô catador de âmbulas

4.5 Nível 2 - Linguagens Livres de Contexto

4.5.1 Gramáticas Livres de Contexto

As GLCs são dispositivos geradores de linguagens mais complexos que as expressões regulares, pois são baseadas em um completo entendimento da estrutura de ”fazer” strings pertencentes à linguagem[60].

Estas gramáticas têm a forma $A \rightarrow \alpha$, onde A é uma variável ou símbolo não-terminal, e α é uma seqüencia qualquer de V^*, que pode ser a cadeia vazia *e*.

Para uma dada regra $A \rightarrow \alpha$, o não-terminal A pode ser substituído por α não importando o contexto ao qual está inserido, por isso a denominação gramática livre de contexto.

Definição 4.3. Uma gramática livre de contexto é dada pelo quádruplo G=(V,$\sum$,P,S), onde

V é um alfabeto

$\sum$ é o conjunto de terminais, sendo um subconjunto de V.

P é o conjunto de produções, sendo um subconjunto finito de $(V - \sum) \times V^*$, e

S é o símbolo inicial, o qual é um elemento de $V - \sum$

Os membros de $V - \sum$ são chamados de símbolos não-terminais.

Definição 4.4. Sendo G uma gramática livre de contexto que gera todas as cadeias em

L(G), uma linguagem *L* é dita linguagem livre de contexto se *L*=*L*(G) para alguma gramática livre de contexto G.

Como exemplo pode-se considerar a gramática livre de contexto G= {V,∑,P,S}, onde V={S, a, b}, ∑= {a,b}, S=S e P consiste das regras $S \rightarrow aSb$ e $S \rightarrow e$. Uma possível derivação de G seria:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

Desta forma, este dispostivo gerador é capaz de gerar cadeias contidas na linguagem $L(G) = \{a^n b^n, tal\ que\ n \geq 0\}$, a qual é considerada um caso clássico de linguagem livre de contexto.

4.5.1.1 Autômato de Pilha

O dispositivo reconhecedor de linguagens livres de contexto é o autômato de pilha. Este dispositivo é análogo ao AFD, mas inclui uma pilha como memória auxiliar que é independente da fita de entrada e não possui limite de tamanho.

Um AP tem como principais componentes as seguintes estruturas:

Fita de entrada: Análoga a dos AFDs, definidos na seção 4.4.2.

Pilha: Memória auxiliar que pode ser usada livremente para leitura e gravação.

Unidade de controle: Reflete o estado corrente da máquina. Possui dois cabeçotes, a *cabeça da fita de entrada* que faz o acesso a uma célula da fita de entrada de cada vez e movimenta-se exclusivamente para a direita e a *cabeça da pilha* que é uma unidade de leitura e gravação, a qual se move para a esquerda (ou "para cima") ao gravar e para a direita (ou "para baixo") ao ler um símbolo. A leitura exclui o símbolo lido, o qual está no topo da pilha.

Função de transição: Comanda a leitura da fita, leitura e gravação da pilha e define o estado da máquina.

Definição 4.5. Um AP é definido como a sêxtupla M={K,∑,Γ,δ,s,*F*}, onde

K é um conjunto finito de estados,

$\sum$ é um alfabeto (os símbolos de entrada),

Γ é um alfabeto contendo os símbolos da pilha,

s é o estado inicial,

$F \subseteq$K é o conjunto de estados finais, e

δ é a relação de transição que é subconjunto finito de $(K \times (\sum, \cup \{e\}) \times \Gamma^*) \times (K \times \Gamma^*)$

No que tange à função de transição δ, o AP pode sofrer duas novas operações: push e pop. A primeira indica que um símbolo é adicionado ao topo da pilha e a segunda faz a indicação que um símbolo deve ser removido do topo da pilha.

Assim, se $\delta(p, a, \beta) = \{[q, \gamma]\}$ significa que M sempre que está no estado p com β no topo da pilha, pode ler a da fita de entrada [3], substituindo β por γ no topo da pilha e entrando no estado q. Assim, a transição $\delta(p, u, \boldsymbol{e}) = (q, a)$ faz um push a, ao passo que a transição $\delta(p, u, a) = (q, \boldsymbol{e})$ faz um pop a.

Os autômatos de pilha também podem ser representados graficamente, sendo que as possíveis operações são representadas nas figuras 4.5, 4.6 e 4.7.

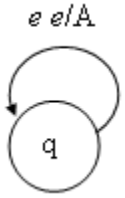

Figura 4.5: Push A

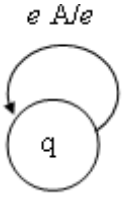

Figura 4.6: Pop A

Na figura 4.5 o autômato lê a string *e* e empilha A, efetivando a transição $\delta(q, \boldsymbol{e}, \boldsymbol{e}) = \{[q, A]\}$[4], já na figura 4.6 a transição indicada é $\delta(q, \boldsymbol{e}, A) = \{[q, \boldsymbol{e}]\}$; finalmente a terceira

[3] se $a = \boldsymbol{e}$, a entrada não é consultada

[4] É importante observar que neste caso não há dependência do símbolo de entrada

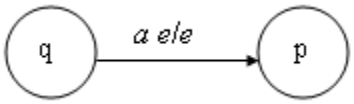

Figura 4.7: Transição equivalente a de um AFD

transição (4.7) indica que a mudança de estado só dependerá do estado atual e do símbolo de entrada, tal qual um AFD, formando assim uma transição $\delta(q, a, e) = \{[q, e]\}$.

De um modo geral, as transições entre estados num AP, funcionam de acordo com a figura 4.8.

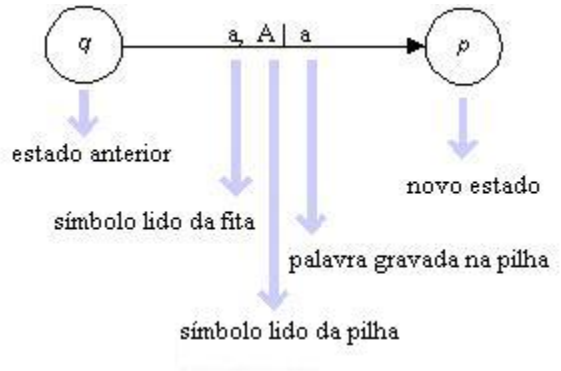

Figura 4.8: Representação gráfica da transição de um AP

Como exemplo pode-se citar o AP que aceita a linguagem $L = \{wcw^R : w \in \{(a, b\}^*\}$.

Para aceitar esta linguagem a máquina precisa ter a capacidade para registrar o processamento de qualquer número de a's e b's. Deste maneira precisamos de um autômato que leia a primeira metade da entrada armazenando os símbolos da fita de entrada (aeb) em uma memória auxiliar, de modo que ao ver um c, o autômato deva começar a desempilhar os símbolos da pilha, até que o símbolo lido da fita de entrada e o símbolo no topo da pilha tenham em seu conteúdo a string vazia *e*[5]. Se isso acontecer, o AP reconheceu a cadeia como sendo uma cadeia em L.

Desta forma este autômato é definido por M=$\{$K,$\sum$,Γ,δ,s,$F\}$, onde

$$\text{K}=\{q_0, q_1\}$$

[5]Embora também se possa projetar um AP que reconheça a linguagem através da chegada a um estado final, neste trabalho iremos utilizar apenas a notação de pilha vazia para reconhecimento de cadeias.

$$\sum=\{a,b,c\}$$

$$\Gamma=\{A,B\}$$

$$s = q_0$$

$$F = \{q_1\}$$

A função de transição δ é definida pelas seguintes regras.

1. $(q_0, a, e) = \{[q_0, A]\}$
2. $(q_0, b, e) = \{[q_0, B]\}$
3. $(q_0, c, e) = \{[q_1, e]\}$
4. $(q_1, a, A) = \{[q_1, e]\}$
5. $(q_1, b, B) = \{[q_1, e]\}$

Este autômato pode ser representado graficamente como ilustrado na figura 4.9 e o processamento da cadeia w=abbcbba neste autômato se dá pela tabela 4.4

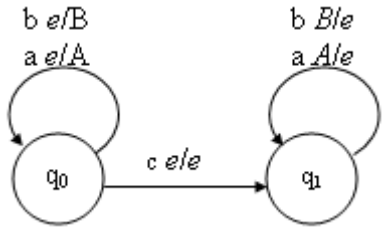

Figura 4.9: AP que reconhece por pilha vazia a linguagem $L = \{wcw^R : w \in \{(a, b\}^*\}\}$

4.5.2 Descrição de comportamentos robóticos utilizando linguagem livre de contexto

Pela maior flexibilidade e expressividade que as linguagens livres de contexto dão à estrutura de um sentença, os comportamentos robóticos representados por este tipo de linguagem vêm a ser mais complexos e interessantes do que os providos pelas linguagens regulares.

Estado	Entrada não lida	pilha	Transição Utilizada
q_0	abbcbba	*e*	-
q_0	bbcbba	A	1
q_0	bcbba	BA	2
q_0	cbba	BBA	2
q_1	bba	BBA	3
q_1	ba	BA	5
q_1	a	A	5
q_1	*e*	*e*	4

Tabela 4.4: Processamento da cadeia abbcbba

Embora em ambientes limitados os comportamentos possam ser representados por AFDs, a utilização de autômatos de pilha torna a representação dos comportamentos que se deseja descrever mais compacta e concisa. Além disso, pode-se observar que quando se pretende generalizar para uma ordem infinita, a representação via AFDs torna-se inviável, necessitando assim de leis de formação de sentenças que gramáticas regulares não podem fornecer.

Para exemplificar comportamentos que são regidos por linguagens livres de contexto, será dado foco a problemas de navegação robótica, onde o robô conterá um pré-mapeamento do ambiente, alcançando um nível além de puramente reativo, passando a possuir uma entidade cognitiva que manifesta características simbólicas relacionadas ao ambiente e ao mapeamento exercido.

Este mapeamento pode ser feito através da utilização de mapas topológicos que possuem a propriedade de fornecer uma representação abstrata do ambiente em que o robô se encontra. Esta abstração é baseada nas informações sensórias que o agente captura durante sua navegação.

Para a utilização de navegação topológica justifica-se a utilização de marcos espaciais ou *landmarks* que segundo Piaget & Inhelder[82] *Apud* Mataric[67] "*Um landmark é uma primitiva especial, servindo como base para representações espaciais*".

Quando o robô encontra um marco no ambiente que já foi definido no seu plano, então

o robô pode utilizá-lo para tomar decisões e para se localizar no ambiente. Além disso, servem também para indicar quando o robô chegou ao fim de um determinado segmento de caminho, bem como identificar que outro caminho está por começar, e o que vem a ser de maior importância para este trabalho: estes marcos podem ser registrados em algum tipo de memória, de modo a serem úteis para emergência de comportamentos peculiares como os que serão vistos nos próximos exemplos.

Um tipo especial de marco são às bifurcações[6] que podem ser intersecções no ambiente que fazem o robô mudar sua direção, escolhendo um caminho possível entre diferentes possibilidades.

A natureza não determinística de um autômato de pilha é capaz de tratar estas entidades, de modo a construir uma espécie de árvore de busca, onde os diferentes caminhos podem ser modelados através do *backtracking* automático que os APs efetivam.

4.5.2.1 Navegação sem bifurcações inconvenientes

Considerando a hipotética situação em que um robô deve percorrer um caminho em busca de um objetivo como uma fonte de luz, restos orgânicos em um terreno devastado por uma explosão, ou mesmo resquícios de água em Marte. Tendo que, após encontrar seu objetivo, ser capaz de voltar pelo mesmo caminho para um ponto inicial.

Para modelar este comportamento de ir e voltar pelo mesmo caminho, surge a necessidade do robô apresentar algum tipo de memória, de modo a guardar os marcos já vistos, para tornar possível a volta pelo mesmo caminho. Desta forma, uma estrutura de pilha é suficiente, pois o robô pode ir empilhando os marcos e na volta desempilhá-los, chegando ao seu destino de modo eficiente. Na figura 4.10, um possível mapeamento para este problema é descrito.

Neste primeiro modelo, todas as direções que o robô pode seguir, levarão ao seu ponto objetivo, portanto as bifurcações apesar de existirem não se tornam um problema tão grande no que tange à busca do robô pelo objetivo. Entretanto um cuidado que se deve tomar ao descrever uma linguagem para representar este comportamento, é o fato de que a volta deve ser feita pelo mesmo caminho que levou o robô ao ponto objetivo **o**.

Os marcos são dispostos em cada esquina do labirinto e são representados por letras,

[6] do inglês gateways

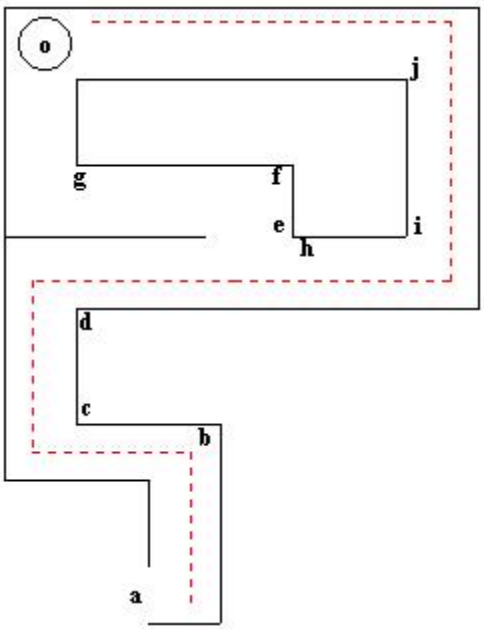

Figura 4.10: Labirinto sem bifurcações incovenientes. A linha pontilhada indica um dos possíveis caminhos válidos

assim o robô vai empilhando os marcos por onde passa, e ao encontrar seu objetivo, começar a desempilhar os marcos, sendo que se no final a pilha tiver um status vazio o robô conseguiu com êxito ir e voltar pelo mesmo caminho.

Desta forma, a linguagem que o robô deve seguir para descrever este comportamento será a linguagem $L = \{wow^R : w \in \sum\}$, onde w será uma cadeia que representa o conjunto de marcos pelos quais o robô irá passar e o símbolo **o** representa o marco final, isto é, o objeto do robô.

A gramática que gera percursos válidos para este problema é a gramática G=(*V*,$\sum$,*P*,*S*), onde

V $= \{A, B, C, D, E, F, G, H, I, J, O\} \cup \sum$

$\sum$=$\{$a,b,c,d,e,f,g,h,i,j,o$\}$

S=A

P consiste das seguintes regras:

$$A \rightarrow aBa$$

$$B \rightarrow bCb$$

$$C \rightarrow cDc$$

$$D \rightarrow dEd \mid dHd$$
$$E \rightarrow eFe$$
$$F \rightarrow fGf$$
$$G \rightarrow gOg$$
$$H \rightarrow hIh$$
$$I \rightarrow iJi$$
$$J \rightarrow jOj$$
$$O \rightarrow o$$

Desta forma, para que o robô siga o percurso representado pela linha vermelha pontilhada na figura 4.10, a seguinte seqüencia de derivação deve ser seguida.

$$\begin{aligned} A &\Rightarrow aBa \\ &\Rightarrow abCba \\ &\Rightarrow abcDcba \\ &\Rightarrow abcdHdcba \\ &\Rightarrow abcdhIhdcba \\ &\Rightarrow abcdhiJihdcba \\ &\Rightarrow abcdhijOjihdcba \\ &\Rightarrow abcdhijojihdcba \end{aligned}$$

O autômato de pilha é dado por M=$\{$K,$\sum$,Γ,δ,s,$F\}$, onde

$$\text{K}=\{q_0, q_1\}$$
$$\sum=\{\text{a,b,c,d,e,f,g,h,i,j,o}\}$$
$$\Gamma=\{\text{Z,A,B,C,D,E,F,G,H,I,J}\}$$
$$s = q_0$$
$$F = \{q_1\}$$

A função de transição δ é definida pelas seguintes regras.

1. $(q_0, a, Z) = \{[q_0, A]\}$
2. $(q_0, b, A) = \{[q_0, BA]\}$
3. $(q_0, c, B) = \{[q_0, CB]\}$
4. $(q_0, d, C) = \{[q_0, DC]\}$
5. $(q_0, e, H) = \{[q_0, EH]\}$
6. $(q_0, f, E) = \{[q_0, FE]\}$
7. $(q_0, g, F) = \{[q_0, GF]\}$
8. $(q_0, h, D) = \{[q_0, HD]\}$
9. $(q_0, i, H) = \{[q_0, IH]\}$
10. $(q_0, j, I) = \{[q_0, JI]\}$
11. $(q_0, o, e) = \{[q_1, e]\}$
12. $(q_1, a, A) = \{[q_1, e]\}$
13. $(q_1, b, B) = \{[q_1, e]\}$
14. $(q_1, c, C) = \{[q_1, e]\}$
15. $(q_1, d, D) = \{[q_1, e]\}$
16. $(q_1, e, E) = \{[q_1, e]\}$
17. $(q_1, f, F) = \{[q_1, e]\}$
18. $(q_1, g, G) = \{[q_1, e]\}$
19. $(q_1, h, H) = \{[q_1, e]\}$
20. $(q_1, i, I) = \{[q_1, e]\}$
21. $(q_1, j, J) = \{[q_1, e]\}$
22. $(q_1, e, Z) = \{[q_1, e]\}$

É importante observar que para superar uma possível situação em que o robô eventualmente retrocedesse em seu percurso, a consulta ao topo da pilha garante que uma sequência inválida não seja reconhecida, e também garante que a volta seja em função do caminho que ele faz em direção ao objetivo. Aliado a isto, a pilha neste AP é inicializada com um símbolo Z, de modo a assegurar que na eventualidade do robô passar pelo ponto inicial mais de uma vez, ele não empilhe os marcos iniciais novamente, o que faria com que o caminho de volta não fosse ideal.

Graficamente este autômato é mostrado na figura 4.11. É importante ressaltar que quando o robô chega no objetivo, e encontra o marco **o**, ele não faz alteração na pilha, apenas muda de estado e começa o processo de desempilhamento.

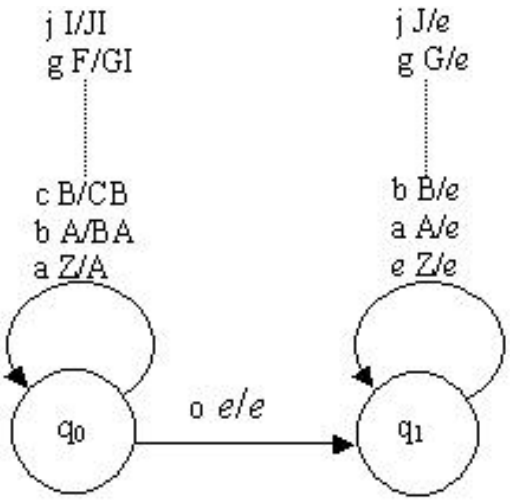

Figura 4.11: Autômato de pilha para labirinto sem bifurcações incovenientes

4.5.2.2 Navegação com bifurcações inconvenientes

Neste exemplo o robô pode se deparar com uma situação que o leva para um estado que o distanciará do objetivo. O AP que modela esta situação deve ser capaz de tratar estes estados, lançando mão de sua característica não determinística para testar situações que podem conduzir o robô a estados válidos no que tange sua busca pelo objetivo. Um possível mapeamento para o problema em questão é ilustrado na figura 4.12.

Neste problema foram colocados marcos em cada cruzamento e em cada beco sem saída (Exs. D e G), além disso, quando o robô percorre um caminho de um marco para

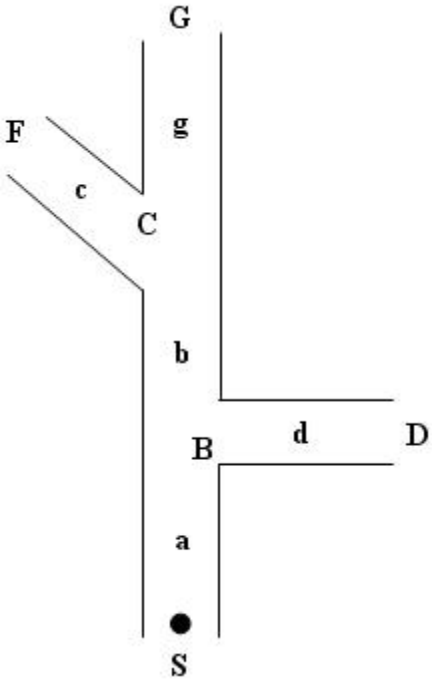

Figura 4.12: Labirinto com bifurcações inconvenientes

outro ele vai guardando em sua memória (pilha) uma denominação para este caminho, que no exemplo estão sendo representadas pelo alfabeto ∑={a,b,c,d,g}.

O robô deve ser capaz de chegar em seu objetivo F, e voltar pelo caminho ótimo, ou seja, se ele tiver passado na ida por caminhos inconvenientes, estes não devem ser gravados na memória (pilha) do robô, fazendo com que na volta ele percorra o caminho inverso apenas pelos melhores trajetos das bifurcações.

A gramática livre de contexto que gera percursos válidos para este problema é a gramática G=(*V*,∑,*P*,*S*), onde

$V = \{S, B, C, D, G, F\} \cup \sum$

∑={a,b,c,d,g}

S=S

P consiste das seguintes regras:

$$S \rightarrow aBa$$

$$B \rightarrow bCb \mid dD \mid e$$

$$C \rightarrow cFc \mid gG \mid e$$

$$D \rightarrow dB$$

$$G \rightarrow gC$$
$$F \rightarrow e$$

Um possível trajeto que o robô deve seguir é dado pela linha tracejada da figura 4.13 e este percurso proposto pode ser gerado pela seguinte derivação.

$$\begin{aligned} S &\Rightarrow aBa \\ &\Rightarrow adDa \\ &\Rightarrow addBa \\ &\Rightarrow addbCba \\ &\Rightarrow addbcFcba \\ &\Rightarrow addbccba \end{aligned}$$

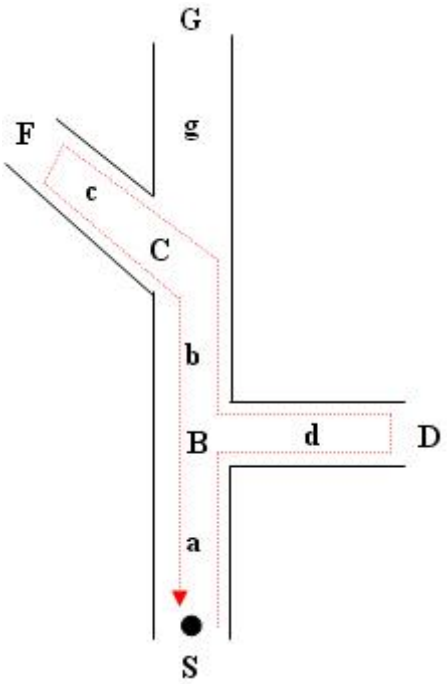

Figura 4.13: Possível percurso

Do mesmo modo, o autômato de pilha que modela este problema é mostrado na figura 4.14.

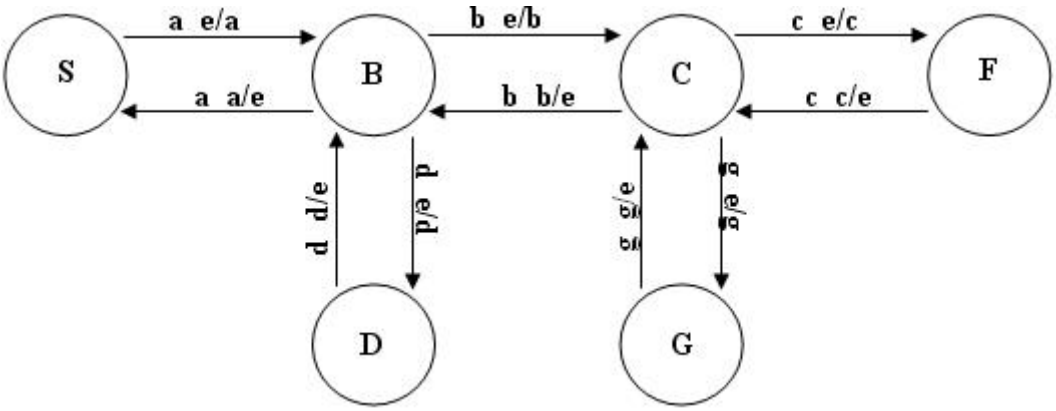

Figura 4.14: Autômato de pilha para labirinto com bifurcações incovenientes

Em termos de reconhecimento de linguagens, o AP apresentado neste exemplo é capaz de reconhecer como válido o percurso *adda*, pois ao apresentar esta cadeia ao AP, a pilha no final da computação estará vazia como mostrado na tabela 4.5. Em termos práticos isto significa que o robô foi e voltou pelo mesmo caminho, mas não chegou ao seu objetivo F. Para contornar este problema, a próxima seção apresenta um labirinto similar, que embora mais complexo, propõe um modelo de autômato que consegue superar este problema.

Estado	Entrada não lida	pilha	Transição Utilizada
S	adda	*e*	-
B	dda	a	a *e*/a
D	da	da	d *e*/d
B	a	a	d d/*e*
S	*e*	*e*	a a/*e*

Tabela 4.5: Processamento da cadeia adda

4.5.2.3 Navegação com garantia de chegada ao objetivo

Neste modelo o labirinto possui um número maior de bifurcações, como ilustrado na figura 4.15, sendo que as regras de derivação propostas na gramática impõem a restrição de que o AA só poderá voltar caso encontre o ponto F, garantindo assim o encontro do seu objetivo.

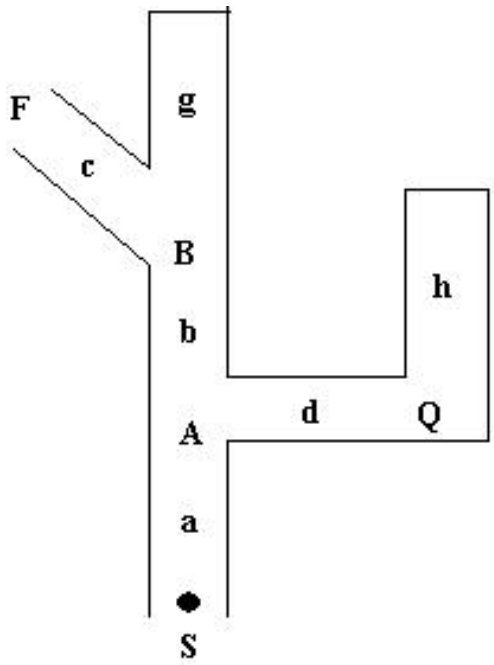

Figura 4.15: Labirinto com mais bifurcações

A formulação da gramática livre de contexto que gera os percursos que este robô deve descrever é dada por G=(*V*,∑,*P*,*S*), onde

V = $\{S, A, B, Q, F\} \cup \sum$

∑={a,b,c,d,g,h}

S=S

P consiste das seguintes regras:

$$S \rightarrow aAa$$
$$A \rightarrow bBb \mid dQ$$
$$B \rightarrow ggB \mid cF$$
$$Q \rightarrow hhQ \mid dA$$
$$F \rightarrow c$$

O percurso válido *adhhdbccba* pode ser gerado pela seguinte seqüencia de transições:

$$\begin{aligned} S &\Rightarrow aAa \\ &\Rightarrow adQa \\ &\Rightarrow adhhQa \end{aligned}$$

$$\Rightarrow \quad adhhdAa$$
$$\Rightarrow \quad adhhdbBba$$
$$\Rightarrow \quad adhhdbcFba$$
$$\Rightarrow \quad adhhdbccba$$

Finalmente o autômato capaz de reconhecer caminhos válidos para este labirinto é dado na figura 4.16.

Este autômato garante que um caminho inválido como *adhhda* não será reconhecido como válido, além do que, para um caminho ser válido o AA precisa necessariamente passar pelo ponto F, bem como voltar ao seu lugar inicial.

Pode-se imaginar um robô enviado para um terreno em busca de resquícios orgânicos em algum ponto objetivo F, de modo que quando tal material fosse encontrado, o robô deveria voltar para a sua base afim de colocar o material em um dispositivo de refrigeração com intuito de manter a conservação necessária para futuras análises biológicas.

4.6 Nível 1 - Linguagens Sensíveis ao Contexto

4.6.1 Gramáticas Sensíveis ao Contexto

As gramáticas sensíveis ao contexto têm esta denominação por permitirem regras da forma $\alpha A\gamma \rightarrow \alpha B\gamma$, isto significa que A pode ser substituído por B, dependendo do contexto em que estiver inserido, ou seja, com α à esquerda e γ à direita.

Além disso, uma produção $\alpha \rightarrow \beta$ de uma GSC deve satisfazer a condição $\|\alpha\| \leq \|\beta\|$, isto implica que o comprimento da cadeia derivada permanece o mesmo ou aumenta no decorrer de cada aplicação das regras. Esta propriedade é conhecida como propriedade de monotonicidade das regras de uma GSC. Esta propriedade garante que não existirá uma regra do tipo $S \rightarrow e$

Formalmente uma GSC é dada pela definição 4.6:

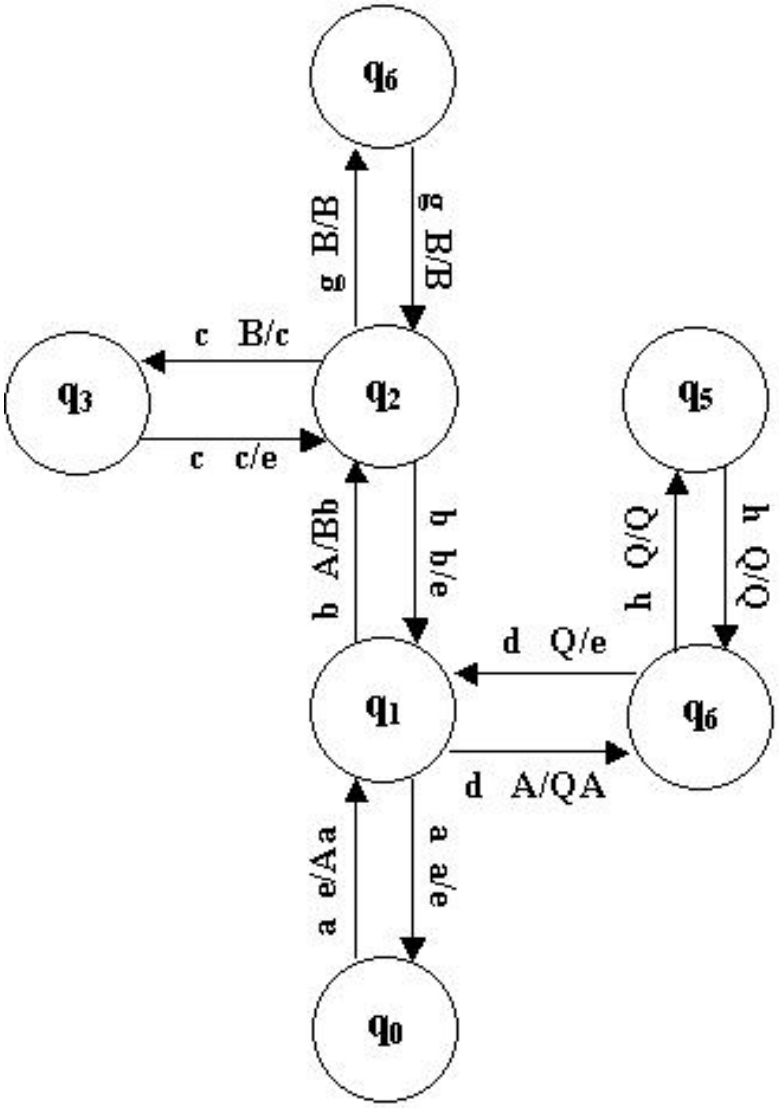

Figura 4.16: AP que modela labirinto com garantia de chegada ao objetivo

Definição 4.6. Uma gramática sensível ao contexto é dada pelo quádruplo G=(*V*,$\sum$,*P*,*S*), onde

V é um alfabeto

$\sum$ é o conjunto de terminais, sendo um subconjunto de *V*.

S é o símbolo inicial, o qual é um elemento de $V - \sum$

P é o conjunto de produções, com cada produção na forma $\alpha \rightarrow \beta$, sendo que $\alpha \; e \; \beta \in V^*$, com $\|\alpha\| \leq \|\beta\|$

Os membros de $V - \sum$ são chamados de símbolos não-terminais.

Definição 4.7. Uma linguagem L é sensível ao contexto se existe uma gramática sensível

ao contexto G, de forma que *L*=*L*(G).

Para exemplificar as gramáticas sensíveis ao contexto, se pode utilizar a geração de cadeias válidas para a LSC $L = \{a^n b^n c^n \parallel n \geq 1\}$. Para isto G seria dada por G=(*V*,$\sum$,*P*,*S*), onde

V={S, B, C, E, a, b, c}

$\sum = a, b, c$

S=S

P consiste das seguintes regras:

$$S \rightarrow abC$$
$$bC \rightarrow bc$$
$$bC \rightarrow bEc$$
$$bE \rightarrow Eb$$
$$aE \rightarrow aaB$$
$$Bb \rightarrow bB$$
$$Bc \rightarrow bcc$$
$$Bc \rightarrow bEcc$$

Para n=3, teríamos as seguintes derivações:

$$\begin{aligned} S &\Rightarrow abC \\ &\Rightarrow abEc \\ &\Rightarrow aEbc \\ &\Rightarrow aaBbc \\ &\Rightarrow aabBc \\ &\Rightarrow aabbEcc \\ &\Rightarrow aabEbc \\ &\Rightarrow aaEbbcc \\ &\Rightarrow aaaBbbcc \\ &\Rightarrow aaabBbcc \end{aligned}$$

$$\Rightarrow \quad aaabbBcc$$

$$\Rightarrow \quad aaabbbccc$$

4.6.2 Autômato Linearmente Limitado

O dispositivo reconhecedor das linguagens sensíveis ao contexto é o autômato linearmente limitado que é uma máquina de Turing não determinística com fita finita, o que garante que a máquina de Turing irá parar dada qualquer entrada, e conseqüentemente será capaz de reconhecer ou não a linguagem. [7]

Definição 4.8. Um autômato linearmente limitado A é a tupla $A =< K, \sum, \Gamma, \delta, s, F >$, onde

K é o conjunto finito e não vazio de estados,

$\sum \subseteq \Gamma$ é o alfabeto de símbolos de entrada,

Γ é o alfabeto de símbolos da fita que contém os marcadores de movimentação da cabeça de leitura/gravação **E** e **D** (esquerda e direita), o símbolo de espaço em branco $\diamondsuit$ e os delimitadores de fita [ ] à esquerda e à direita, respectivamente,

δ é a função de transição que é um mapeamento $K \times \Gamma \rightarrow K \times \Gamma \times \{E, D\}$

$s \in K$ é o estado inicial,

$F \subseteq K$ é o conjunto de estados finais.

Uma transição de um ALL é dada da mesma forma que na máquina de Turing, ou seja, uma transição da forma $\delta(q, a) = (p, b, D)$ indica que ao ler um símbolo a, estando no estado q, a cabeça da fita escreve um b no lugar de a, move-se para a direita e a unidade de controle passa para o estado p.

4.6.3 Descrição de comportamentos robóticos utilizando linguagens sensíveis ao contexto

Com o intuito de demonstrar a aplicação de uma linguagem sensível ao contexto como modelo de descrição de comportamentos robóticos, será dado um exemplo de um robô

[7] Na seção 5.2, a máquina de Turing é brevemente explanada.

que atua em uma linha de montagem de peças automotivas.

O objetivo do robô é fazer a montagem de uma peça que deve ser fixada nos carros através de um parafuso, uma porca e uma arruela. Neste setor da produção o robô deve ir fixando individualmente cada peça em uma quantidade n de carros, isto é, ele deve primeiro encaixar o parafuso, em seguida colocar a arruela e finalmente afixar o parafuso com a porca (figura 4.17).

Figura 4.17: Parafuso, arruela e porca

Os carros nesta linha de produção estão dispostos lado a lado, nas proximidades de um corredor onde o robô caminha. Assim o robô deve passar em cada carro e fixar uma peça de cada vez em todos os carros.

O robô é equipado de um aparato que simula uma mão mecânica para efetuar a composição da peça, uma câmera que identifica os carros através de *templates* pré-programados e uma caixa de ferramentas, onde os parafusos, porcas e arruelas são colocados.

Inicialmente o robô tem sua caixa de ferramentas carregada de parafusos, sendo abastecida em ambos os lados do corredor com arruelas e porcas, nesta mesma ordem.

Em suma, o robô irá percorrer o corredor inicialmente com parafusos que serão colocados nos carros. Ao chegar ao final do corredor os parafusos são trocados por arruelas e o robô volta colocando as arruelas, que novamente ao chegar ao final do corredor, serão trocadas por porcas na caixa de ferramentas e o robô irá afixar as porcas, chegando ao final do corredor com o seu objetivo concluído, podendo passar para uma outra fase desta linha de produção automotiva.

Este comportamento de levar parafusos, arruelas e porcas individualmente para *n* carros pode ser representado pela linguagem $L = \{a^n b^n c^n \mid n \geq 1\}$, onde a, b, e c repre-

sentam os utensílios que o robô deve levar para fixar as peças, que são respectivamente parafusos, arruelas e porcas e *n* representa o número de carros em que o robô irá efetuar a tarefa em questão.

A gramática que gera comportamentos válidos para esta linguagem é a mesma dada no exemplo referente à definição 4.7. Assim, a geração de uma cadeia aabbcc, indica que o AA em questão levou dois parafusos para dois carros, em seguida colocou duas arruelas e finalmente voltou fixando as porcas que faltavam para assegurar a fixação do parafuso. Para a geração desta cadeia, o seguinte processo de derivação foi descrito:

$$\begin{aligned} S &\Rightarrow abC \\ &\Rightarrow abEc \\ &\Rightarrow aEbc \\ &\Rightarrow aaBbc \\ &\Rightarrow aabBc \\ &\Rightarrow aabbcc \end{aligned}$$

O autômato que representa este comportamento é dado pelo ALL $A =< K, \sum, \Gamma, \delta, s, F >$, onde

$K = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{a, b, c\}$

$\Gamma = \{A, B, C, \diamondsuit, E, D, [,]\} \cup \sum$

$s = q_0$

$F = \{q_6\}$

δ é dada pelas seguintes regras

1. $(q_0, [) = \{q_1, [, D\}$
2. $(q_1, a) = \{q_2, A, D\}$
3. $(q_1, B) = \{q_5, B, D\}$
4. $(q_2, a) = \{q_2, a, D\}$

5. $(q_2, B) = \{q_2, B, D\}$
6. $(q_2, b) = \{q_3, B, D\}$
7. $(q_3, b) = \{q_3, b, D\}$
8. $(q_3, C) = \{q_3, C, D\}$
9. $(q_3, c) = \{q_4, C, E\}$
10. $(q_4, a) = \{q_4, a, E\}$
11. $(q_4, b) = \{q_4, b, E\}$
12. $(q_4, B) = \{q_4, B, E\}$
13. $(q_4, C) = \{q_4, C, E\}$
14. $(q_4, A) = \{q_1, A, D\}$
15. $(q_5, B) = \{q_5, B, D\}$
16. $(q_5, C) = \{q_5, C, D\}$
17. $(q_5,]) = \{q_6,], D\}$

Graficamente este autômato é ilustrado na figura 4.18

4.7 Classificação dos Comportamentos

De posse destes exemplos se consegue demonstrar a viabilidade da descrição de comportamentos robóticos utilizando a hierarquia de Chomsky, com suas respectivas linguagens e autômatos. Entretanto, surge a necessidade de uma classificação no âmbito de quais linguagens seriam necessárias para determinados padrões de comportamentos.

O escopo deste trabalho não é definir uma prova formal e matemática para esta classificação, sendo isto alvo de futuros trabalhos que venham a surgir nesta nova linha de pesquisa. Todavia, algumas evidências podem ser aventadas como ponto de partida para esta classificação.

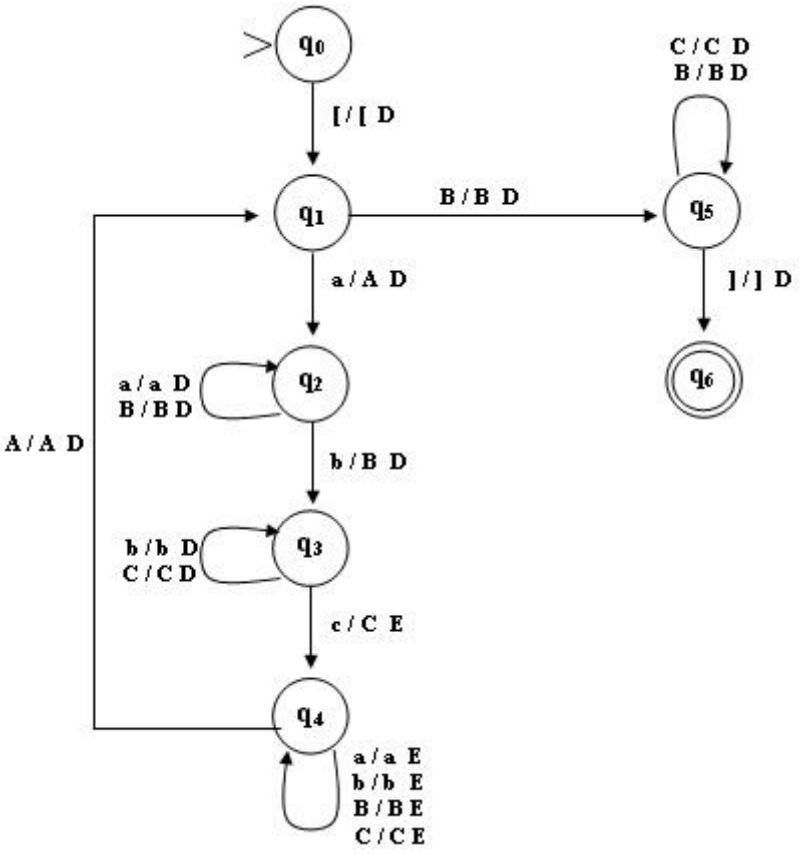

Figura 4.18: Autômato linearmente limitado para robô atuando em uma linha de montagem

Primeiramente se pode citar o fato da memorização e do método de acesso ao dispositivo de memória ser crucial para a escolha de que nível da hierarquia de Chomsky utilizar.

Nos exemplos dos labirintos, às tomadas de decisão que o robô deveria efetivar foram se tornando construtivamente menos elementares, e sempre estas decisões se embasavam em que tipo de informação o mecanismo de memória (pilha) poderia fornecer. Certamente, para problemas deste tipo, onde se necessita uma informação sequencial de um ponto imediatamente anterior a uma tomada de decisão, a estrutura da pilha se torna suficiente, deixando uma linguagem livre de contexto como sendo o modelo relativamente ideal para a representação do comportamento.

Quando o objetivo do AA necessita de um método de acesso a memória que garanta que as informações irão permanecer íntegras até o final de toda a computação, necessita-se de uma linguagem sensível ao contexto. O autômato reconhecedor desta linguagem,

por ser uma máquina de Turing com fita limitada, garante esta memorização permanente das informações, mantendo o acesso a elas em qualquer momento da computação, obviamente, desde que suas regras de transição estejam definidas para tanto.

Este assunto será visto com mais profundidade no próximo cápítulo, onde será mostrado quando se necessita de uma máquina de Turing (mesmo que de fita limitada, como os ALLs) para representar os comportamentos robóticos.

Em paralelo pode-se levar em consideração que existe um teorema chamado de Teorema do Bombeamento (vide [60] e [102]) que garante que uma dada linguagem de menor nível da hierarquia de Chomsky não pode ser representada por uma linguagem de nível superior. Por exemplo, uma livre de contexto não pode ser representada por uma linguagem regular.

Este teorema leva a inferência que os comportamentos são auto-ditados pela própria linguagem, ou seja, quando a representação de um comportamento se dá através de uma linguagem sensível ao contexto, ele não poderá ser feito através de uma linguagem livre de contexto. De fato, não seria possível representar o comportamento visto no exemplo da subseção 4.6.3(problema das peças automotivas) por uma LLC, e muito menos por uma LR, pois isto iria transgredir o teorema do bombeamento.

Capítulo 5

Definição Formal de Agentes Autônomos como máquina de Turing

"One day we will take robots out of the realm of toys and into the realm of thinking machines". ($^{tm} Dan\ Mathias$*)*

5.1 Introdução

Pelo fato deste trabalho levantar idéias que visam a formalização dos comportamentos robóticos, surge a relevância e necessidade de se utilizar um formalismo para a definição dos próprios agentes autônomos e de seus comportamentos de um modo genérico.

Em geral as definições de AAs e de seus comportamentos encontradas na literatura circundam nas definições vistas no capítulo 2 deste trabalho, onde se observa que estas definições se baseiam em conceitos qualitativos, informais e por vezes puramente filosóficos. Uma definição mais formal é encontrada em [89], o qual faz uma definição de AAs e seus comportamentos utilizando Teoria Geral de Sistemas.

Para manter a linha de pesquisa deste trabalho, será dado enfoque as estas definições utilizando a máquina de Turing (MT), pois embora existam os dispositivos reconhecedores mais simples para cada linguagem da hierarquia de Chomsky, a máquina de Turing se comporta como um dispositivo capaz de reconhecer qualquer linguagem dos níveis acima da hierarquia, de modo que através dela será feita uma nova definição formal dos agentes autônomos.

5.2 Máquina de Turing

A máquina de Turing (MT) foi proposta pelo matemático inglês Alan Turing ao analisar a situação de uma pessoa equipada com uma caneta, apagador e uma folha de papel organizada em quadrados. Inicialmente o papel contém somente os dados iniciais do problema, e a pessoa pode interagir com o problema lendo e alterando um símbolo em um quadrado e movendo os olhos para outro quadrado.

Com isso, Turing construiu uma máquina que consegue ser um modelo formal para a noção de algoritmo, além disso a tese de Church-Turing postula que qualquer dispositivo computacional tem seu poderio de computabilidade no máximo equivalente ao da MT. [102]

A máquina de Turing é composta por três componentes que são a fita, a cabeça de leitura/gravação e a unidade de controle. Uma ilustração da MT pode ser visualizada na figura 5.1.

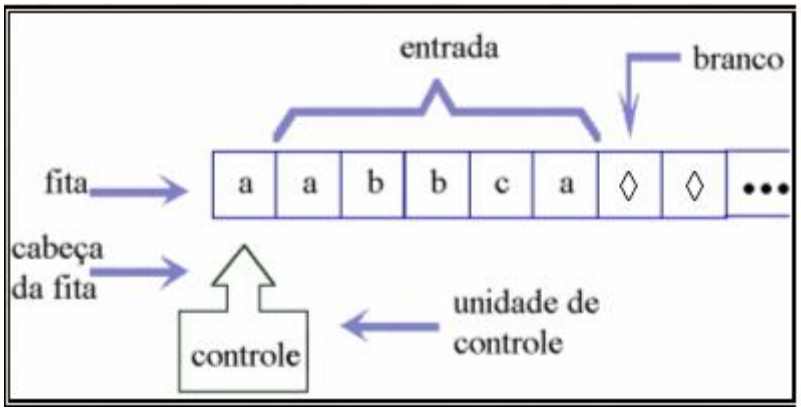

Figura 5.1: Máquina de Turing

A fita da MT pode ser utilizada como dispositivo de entrada, saída e memória de trabalho, tem tamanho infinito à direita e cada célula armazena um símbolo que pode ser do alfabeto de entrada, do alfabeto de símbolos exclusivos da fita ou o símbolo de espaço em branco. Inicialmente, a palavra a ser processada ocupa as células mais à esquerda, ficando as demais com o símbolo de espaço em branco ◊, assim como é mostrado na figura 5.1.

A cabeça de leitura/gravação ou simplesmente cabeça da fita é a responsável pela comunicação entre a unidade de controle e a fita, sendo utilizada tanto para ler o conteúdo,

como para escrever os símbolos na fita. Ela inicialmente está posicionada na primeira célula à esquerda da fita, e seu movimento se dá acessando uma célula de cada vez, em direção à esquerda ou à direita.

A unidade de controle reflete o estado interno corrente da máquina, possuindo um número finito e pré-definido de estados. Esta entidade opera em passos discretos, podendo executar a cada passo uma troca do estado atual, bem como, através da cabeça da fita, gravar um símbolo na célula vigente da fita, movendo a cabeça para a direita ou esquerda.

Definição 5.1. Uma **Máquina de Turing** é a quíntupla $M =< K, \sum, \Gamma, \delta, s, F >$, onde

K é o conjunto finito e não vazio de estados,

$\sum \subseteq \Gamma$ é o alfabeto de símbolos de entrada,

Γ é o alfabeto de símbolos da fita que contém os marcadores de movimentação da cabeça de leitura/gravação **E** e **D** (esquerda e direita) e o símbolo de espaço em branco $\diamondsuit$.

δ é a função de transição que é um mapeamento $K \times \Gamma \rightarrow K \times \Gamma \times \{E, D\}$

$s \in K$ é o estado inicial,

$F \subseteq K$ é o conjunto de estados finais.

A computação de uma MT inicia com a máquina em um estado $q_0 \in K$, com a cabeça da fita na célula mais à esquerda. As transições consistem de três ações: mudar o estado corrente, escrever um símbolo na célula lida pela cabeça de leitura/gravação e mover a cabeça para à esquerda ou direita. Por exemplo, a transição $\delta(q, a) = (p, b, D)$, pode ser entendida como M no estado q, lendo o símbolo a na fita, deve escrever o símbolo b na mesma célula onde estava a, passar para o estado p e mover a cabeça da fita para à direita. As transições em uma MT também podem ser definidas através de um grafo finito direto como ilustrado na figura 5.2.

Uma MT pode parar sob três circustâncias:

- **Estado Final:** A máquina chega a um estado final, aceitando a cadeia de entrada.
- **Transição Indefinida:** Nenhuma transição δ se enquadra na configuração atual da máquina, de modo que a máquina pára e rejeita a entrada.
- **Limite à Esquerda:** A transição requer um movimento à esquerda e a cabeça da fita já se encontra na célula mais à esquerda, desta forma a máquina pára, e a entrada é rejeitada.

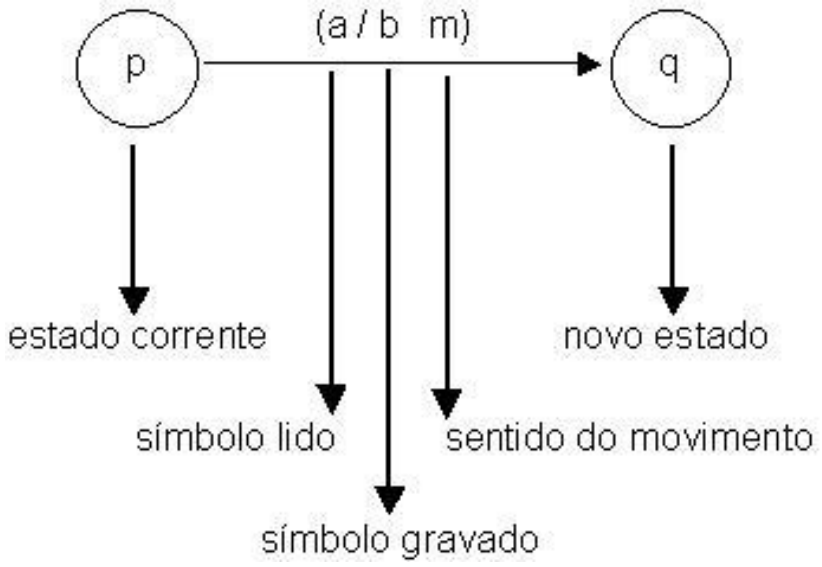

Figura 5.2: Representação em forma de grafo da transição de uma MT

A máquina de Turing em algumas situações pode ficar rodando indefinidamente a fim de tentar reconhecer uma cadeia. As linguagens que contém estas cadeias são chamadas de enumeráveis recursivamente.

Neste contexto, é importante ressaltar que neste trabalho conjectura-se que os comportamentos robóticos são regidos pelos níveis 1,2 e 3 da hierarquia de Chomsky, não adentrando assim nas linguagens enumeráveis recursivamente, o que garante que para os problemas vistos neste trabalho a MT irá parar para todas as entradas. Para mais detalhes sobre linguagens recursivamente enumeráveis e questões de computabilidade da MT, consultar [60] e [102].

A máquina de Turing pode ser utilizada sob dois enfoques: como um dispositivo reconhecedor genérico de linguagens e como processadora de funções.

Na primeira abordagem um conjunto de estados finais é especificado e a MT deverá aceitar ou rejeitar a cadeia apresentada na fita de entrada.

Um segundo enfoque é utilizá-la como processadora de funções, onde ela é capaz de computar funções matemáticas, tendo o valor contido na fita de entrada como parâmetro da função. Este tipo de aplicação da MT contém apenas um estado final que representa o término da computação da função, indicando que a máquina conseguiu ter sucesso em seu processamento. Caso a máquina pare em um outro estado, ela não foi capaz de computar a função, pois suas regras de transições não comportam tal situação.

Novamente se deve realçar que a MT pode ficar rodando indefinidamente. No primeiro caso isto acontece para algumas linguagens enumeráveis recursivamente, visto que não há garantia que a MT irá parar para uma dada cadeia de entrada, e no segundo enfoque também não há garantia que para uma dada função a MT irá parar em seu estado final.

5.3 Agente Autônomo como Máquina de Turing

Como inspiração para as definições em questão pode-se partir da premissa que um agente autônomo também pode ser considerado um mecanismo de computação que recebe uma entrada, efetua algum processamento e emite uma saída.

Desta forma, analogamente, neste trabalho visa-se definir um AA como sendo uma MT, onde a fita é o ambiente onde o agente atua, o alfabeto dito de entrada contém os símbolos que representam tanto as entradas que os sensores conseguem captar, como a representação simbólica dos comportamentos emergidos pelo agente.

A cabeça de leitura/gravação representa os sensores ao efetuar operações de leitura, também se comportando como um mecanismo atuador quando estiver atuando em operações de gravação.

Já o mecanismo de controle de estados representa a entidade cognitiva do robô, a qual é orientada a fazer o mapeamento cognitivo entre os dados provenientes dos sensores (comportamento de leitura da cabeça) e a emergência dos comportamentos robóticos (comportamento de escrita da cabeça).

Em suma, a emergência de um comportamento se dá na medida em que a máquina de Turing computa os dados contidos na recepção sensória, integra-os cognitivamente através do controle de estados, e computa uma saída que será o comportamento emergido.

Definição 5.2. Um agente definido por uma máquina de Turing pode ser dado pela tupla $A =< K, \sum, \Gamma, \delta, s, F >$, onde

K é o conjunto finito e não vazio de estados cognitivos do agente.

Γ é o alfabeto de símbolos da fita que contém os marcadores de movimentação da cabeça de leitura/gravação **E** e **D** (esquerda e direita) e o símbolo de espaço em branco $\diamondsuit$

$\sum \subseteq \Gamma$ é o alfabeto de símbolos que representam os dados captados pelo aparato sensório do agente e as ações que podem ser efetuadas por ele. Este alfabeto é o conjunto

de símbolos que representam a interação do agente com o ambiente.

δ é a função de transição que realiza o mapeamento entre a recepção sensória, os estados cognitivos do agente e o comportamento emergido, se apresentando na forma $\delta : K \times \Gamma \rightarrow K \times \Gamma \times \{E, D\}$

s é o estado cognitivo inicial do agente

$F \subseteq K$ é o conjunto de estados cognitivos finais que o agente pode alcançar, indicando que o comportamento emergido é válido frente às informações sensórias providas pelo ambiente.

Definição 5.3. O **comportamento observável** emergido por um agente é o mapeamento cognitivo da função $\delta : K \times \Gamma \rightarrow K \times \Gamma \times \{E, D\}$ de uma entrada dada por uma cadeia de um ou mais símbolos sensórios $w \in K$ para uma saída de símbolos $w' \in K$.

Definição 5.4. Um **comportamento observável válido** é um comportamento observável que tem sua cadeia de saída em um dos estados cognitivos finais $F \subseteq K$.

De fato, o fator decidibilidade de uma dada cadeia de símbolos não garante que a parada da MT seja um estado final válido, portanto para que se tenha um comportamento observável válido é necessário que o estado cognitivo do agente ao final da computação seja em um estado final válido.

Outro fator importante que se deve considerar é a questão do aprendizado dos AAs. A máquina de Turing suporta a representação deste aprendizado, pois as operações sobre sua fita permitem que a saída computada pela MT (comportamento) possa ser representada de maneira concomitante com a entrada, haja vista a memória infinita ou suficientemente grande para armazenar todos os símbolos necessários à computação da máquina.

Elucidando o páragrafo anterior, existe a possibilidade pragmática de se criar uma MT com fita infinita à esquerda e à direita separadas por um símbolo qualquer Y que controla a fronteira entre as partes esquerda e direita. Na figura 5.3, um exemplo desta fita é ilustrado, onde as células de índice negativo representam a parte direita da fita, e as de índice positivo, a parte esquerda

Com este aparato pode-se convencionar que os símbolos provenientes dos sensores estariam à esquerda de Y e o comportamento desempenhado pelo agente estaria à direita de Y, desta forma garantindo o mapeamento entre a recepção sensória e a emergência do

comportamento em uma mesma estrutura de memória. Para um estudo mais aprofundado da computação destas máquinas “incrementadas”, recomenda-se consultar [60] e [102] [1].

No exemplo do labirinto apresentado na subseção 4.5.2.3, a garantia de que o robô fez o caminho correto, é de que no final de sua computação a pilha dever estar vazia, no entanto, os passos que ele percorreu ao ir ao objetivo e voltar para a origem não foram gravados permanentemente em lugar algum, pois o método de acesso a pilha do AP obriga a exclusão das informações caso se queira a pilha vazia no final do processo.

Entretanto, com uma máquina de Turing de duas fitas seria possível que o caminho percorrido pelo robô pudesse ser gravado de maneira permanente, de modo que se insistíssemos em fazê-lo percorrer o labirinto novamente, ele já teria os passos de um momento anterior gravados na fita, realizando mais facilmente o percurso em questão.

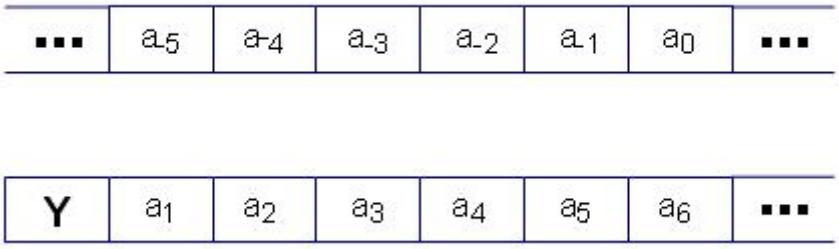

Figura 5.3: Fita infinita nos dois sentidos

A proposição que o aprendizado pode ser representado por uma máquina de Turing, faz com que ela aja como um modelo de memória de longa duração (vide seção 6.4), ou seja, ela permite a possibilidade de conter uma cadeia de entrada e o resultado de sua computação em um único meio, que no caso é sua fita.

Os autômatos de pilha também permitem certa memorização, no entanto, seu dispositivo de armazenamento (pilha) se comporta como uma espécie de *memória de trabalho* (vide seção 6.4), pois o meio de acesso a ele acontece de uma maneira que não permite a memorização permanente das entradas e saídas concomitantemente, tornando-se então, apenas uma memória de trabalho ou de curta duração.

Finalmente, os AFDs são dispositivos que não possuem nem uma memória auxiliar,

[1] Apesar destas máquinas trazerem uma maior facilidade de implementação, elas não aumentam a potencialidade da máquina de Turing dita universal, pois as mesmas operações podem ser realizadas por esta. Para mais detalhes sobre os teoremas que circundam esta afirmação, consultar [102].

muito menos memória permanente, acarretando que os comportamentos que eles são capazes de descrever são comportamentos mais simples, que em geral são ações puramente reflexivas.

Capítulo 6

Modelos de Redes Neurais Artificiais para implementação das Linguagens de Chomsky

"Qualquer coisa que possa ser descrita exaustivamente e sem ambiguidade, qualquer coisa que possa ser plenamente e sem ambiguidade colocada em palavras, é, ipso facto, realizável por uma rede finita adequada"(John Von Neumann)

6.1 Introdução

No capítulo 4 foram apresentadas formas de se representar comportamentos robóticos utilizando os diferentes níveis da hierarquia de Chomsky. Feita esta representação surge a necessidade de encontrar métodos ou arquiteturas de controle capazes de implementar estes comportamentos regidos pelas linguagens de Chomsky.

Ao se imaginar a navegação de um AA sobre um ambiente dinâmico, como um corredor de uma universidade, torna-se inviável construir um algoritmo convencional que consiga modelar todas as situações, como pessoas transitando em coordenadas diferentes, objetos mudando de posição, produtos de limpeza sobre o chão em determinado momento, etc.

Além disso, o tratamento algorítmico convencional não é compatível com os dados analógicos que os sensores capturam do ambiente, necessitando assim, de uma discretização

dos valores lidos para um nível simbólico. Ademais o paradigma computacional baseado em procedimentos ou objetos não tem um mecanismo automático de tratamento de falhas, devendo às eventuais incertezas estarem sempre previstas pelo projetista.

Os seres vivos conseguem prosperar em um ambiente dinâmico e por vezes hostil, realizando o controle de sua navegação e o tratamento das "imperfeições" do ambiente de maneira inata. A emergência de comportamentos se dá através do sistema nervoso central, o qual tem como seus elementos atômicos os neurônios que formam redes neuronais responsáveis por um mapeamento cognitivo entre as informações recebidas do ambiente, e a forma como atuar nele.

Se a emergência de comportamento dos seres da natureza ocorre de forma inata e inteligente, aliada ao fato de possuírem em seus comportamentos o objetivo de prosperar na execução de suas tarefas (que pode ser até mesmo o ato de sobreviver), torna-se promissor estudar meios de implementar modelos inspirados biologicamente.

Em diversos experimentos [24][103][26][42][15][16] as RNAs conseguem aprender a estrutura de uma dada linguagem, conseguindo induzir regras de derivação gramaticais, bem como reconhecer sentenças válidas de uma dada linguagem. Isto se torna de grande utilidade para aplicações no âmbito da robótica, pois auxilia na decisão que um AA deve tomar para a efetivação de um dado comportamento.

Neste contexto, justifica-se a utilização das redes neurais artificiais como forma de dotar um AA com características inteligentes e autônomas.

Para tanto as redes neurais artificiais devem ser modelos que consigam implementar os diferentes níveis da hierarquia de Chomsky, pois neste trabalho conjectura-se que os comportamentos robóticos podem ser representados pelas linguagens dispostas nesta hierarquia.

Nas seções deste capítulo, as redes neurais artificiais serão introduzidas ao leitor, dispondo conhecimento a respeito das suas principais características e potencialidades, para em seguida mostrar modelos já consagrados na literatura de redes neurais que conseguem agir como autômatos. Isto é fundamental para este trabalho, pois serão dados métodos para que agentes autônomos dotados cognitivamente de redes neurais artificiais consigam descrever comportamentos robóticos regidos pelos diferentes níveis da hierarquia de Chomsky.

Além disso, algumas considerações relativas a questões de computabilidade das redes neurais serão explanadas, frente sua suposta equivalência à máquina de Turing.

Finalmente, retomando à arquitetura PiramidNet, explanada na seção 2.5.4, a qual se inspira nas características modulares e hierárquicas do cérebro, pretende-se incluir novas camadas que representarão as RNAs que implementam níveis mais baixos da hierarquia de Chomsky, provendo assim uma gama de comportamentos maior.

6.2 Redes Neurais Artificiais

O ser humano é dotado de complexos circuitos neuronais que constam de variadas conexões entre seus neurônios (sinapses) interagindo entre si de modo a fazer emergir comportamento inteligente. Sendo assim, surge a idéia de se tentar modelar computacionalmente estas conexões neurais, de modo a incitar a emergência de comportamento inteligente em máquinas. Neste contexto, surgem as redes neurais artificiais que são inspiradas na própria natureza das redes de neurônios e sinapses biológicas. Esta idéia de modelagem cerebral forma a vertente da Inteligência Artificial, chamada Inteligência Artificial Conexionista. [108]

As RNAs também surgem para os cientistas da computação como um novo paradigma de programação, baseada em exemplos, não necessitando de algoritmo explícito, aproveitando a potencialidade das RNAs como entidades flexíveis à generalização, efetivando inferências corretas mesmo para dados não vistos pela rede anteriormente.

Além disso, as redes neurais são um paradigma computacional que inere o paralelismo em seu processamento, onde cada neurônio pode ser considerado uma entidade que executa certo processamento.

Para aplicações na robótica, as RNAs apresentam a interessante característica de suportar ruídos atrelados as imperfeições no ambiente. Outro ponto importante que sedimenta a utilização das RNAs é sua tolerância a falhas, ou seja, geralmente conseguem chegar ao resultado desejado mesmo com mal funcionamento de algum componente.

Apesar do pouco conhecimento que ainda se tem do cérebro humano, aliado à dificuldade de modelar mesmo o que já sabemos, as pesquisas em redes neurais tem se mostrado bastantes promissoras em diversas áreas, como engenharia, computação e até mesmo nas

neurociências. Sendo também de grande utilidade para problemas como reconhecimento de padrões, agrupamento, previsão de séries temporais e obviamente na robótica.

6.2.1 Neurônio Biológico

Um neurônio típico é composto por um corpo celular ou soma, um axônio tubular e várias ramificações arbóreas conhecidas como dendritos. Os dendritos formam uma malha de filamentos finíssimas ao redor do neurônio. Ao passo que o axônio consta de um tubo longo e fino que ao final se divide em ramos que terminam em pequenos bulbos que quase tocam os dendritos dos outros neurônios na fenda sináptica. Na figura 6.1 é mostrada a ilustração de um neurônio.

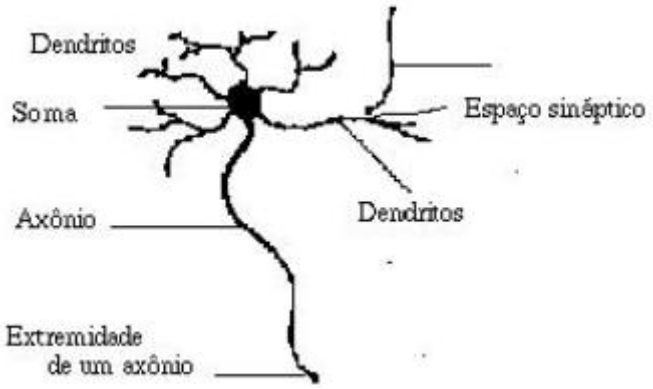

Figura 6.1: Neurônio Biológico

O pequeno espaço entre o fim do bulbo e o dendrito é conhecido como sinapse, através da qual as informações se propagam. O número de sinapses recebidas por cada neurônio varia de 100 a 100.000, sendo que elas podem ser tanto excitatórias como inibitórias. A figura 6.2 representa uma possível sinapse entre dois neurônios.

Figura 6.2: Sinapse

A célula nervosa tem um potencial de repouso devido aos íons Na^+ e K^- estarem

em concentrações diferentes dentro e fora da célula, de modo que qualquer perturbação na membrana, estimulada por elementos como luminosidade, reações químicas e pressão provocam alteração nas concentrações dos íons Na^+ e K^- [12].

A alteração na concentração dos íons Na^+ e K^- gera um trem de pulsos elétricos que se expande localmente nas proximidades dos dendritos. Dependendo da intensidade do estímulo, este trem de pulso pode exceder um certo limiar no corpo celular e gerar um sinal com amplitude constante ao longo do axônio. Na fronteira do momento do disparo do neurônio, é gerado um potencial de ação que impulsiona o fluxo do sinal gerado pelo corpo celular para outras células

O pulso elétrico gerado pelo potencial de ação libera neurotransmissores que são substâncias químicas contidas nos bulbos do axônio. Estes neurotransmissores são repassados para os dendritos do neurônio seguinte. Assim, quando o conjunto de neurotransmissores que chegam aos dendritos de um determinado neurônio atinge um certo limiar, eles disparam de novo um potencial de ação que vai repetir todo o processo novamente.

Convém ressaltar que as sinapses podem ser excitatórias, facilitando o fluxo dos sinais elétricos gerados pelo potencial de ação, como podem também ser inibitórias, as quais possuem a característica de dificultar a passagem desta corrente.

Esta seção teve o intuito de apenas contextualizar a inspiração biológica pela qual as redes neurais artificiais são idealizadas. Para mais informações acerca do funcionamento do sistema neural, o leitor pode consultar [75] e [58] que foram utilizados como literatura base para esta seção.

6.2.2 Neurônio Artificial

Apesar dos esforços em se modelar os neurônios biológicos, tudo que se conseguiu até hoje foi uma aproximação elementar. Neste trabalho mostraremos um modelo adequado com o proposto por Azevedo[9] *apud* [89]. A figura 6.3 mostra este modelo.

Neste modelo as entradas do neurônio u_j podem ser saídas de outros neurônios, entradas externas, um bias (entrada especial) ou qualquer combinação destes elementos. Estas entradas são ponderadas pelos pesos w_{ij} que são inspirados na força da conexão sináptica entre os neurônios i e j.

Desta forma, temos o chamado net do neurônio que é geralmente o somatório de todas

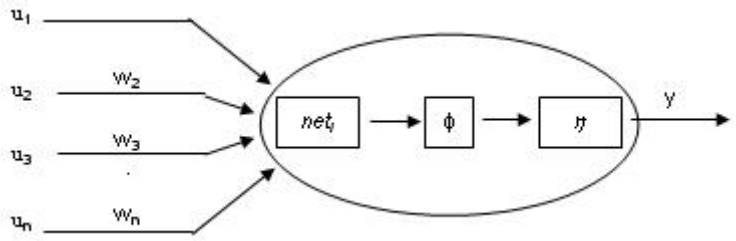

Figura 6.3: Modelo de Neurônio Artificial

as entradas multiplicadas pelos seus respectivos pesos. Ou seja:

$$net_i = \sum_{1}^{n} u_j w_{ij}$$

A função de ativação ϕ só existe em neurônios dinâmicos que são neurônios onde seus estados futuros são determinados tanto pelo net de entrada como pelo estado atual do neurônio, ou seja, este tipo de neurônio tem uma certa "memória"que permite guardar informação de estados anteriores. Entretanto, em grande parte da literatura considera-se a função de ativação dependente apenas das entradas atuais e dos pesos, tornando os neurônios em entidades estáticas[89].

A função de saída η é quem produz a saída do neurônio e normalmente tem forma contínua e crescente, de tal sorte que seu domínio geralmente se encontra no âmbito dos números reais. Geralmente utiliza-se como função de saída as funções lineares, sigmoidal ou logística e a função tangente hiperbólica.

6.2.3 Topologia das RNAs

Para a vasta maioria dos problemas práticos um único neurônio não é suficiente, sendo assim utilizam-se neurônios interconectados, sendo que a decisão de como interconectar os neurônios é uma das mais importantes decisões a se tomar no projeto de uma rede neural artificial. Ademais, a forma como os neurônios estão conectados influi diretamente na capacidade da rede para resolver determinada classe de problemas[106].

No tocante de como os neurônios se interligam, é conveniente ressaltar a utilização de camadas intermédiárias (ou ocultas) que permitem as RNAs implementarem superfícies de decisão mais complexas. Estas camadas permitem que seus elementos se organizem de tal forma que cada elemento aprenda a reconhecer características diferentes do espaço

de entrada. Assim, o algoritmo de treinamento deve decidir que características devem ser extraídas do conjunto de treinamento. O problema em utilizar camada escondida é que o aprendizado se torna muito mais difícil

As redes neurais artificiais podem ser diretas ou recorrentes, sendo que a principal diferença entre elas é que na primeira os neurônios não recebem realimentação em suas entradas, ou seja, seu grafo não tem ciclos.

Atualmente as redes neurais diretas são as mais comuns, principalmente pelo advento da popularização do algoritmo de treinamento *backpropagation*. Este tipo de rede pode ser considerado um aproximador universal de funções, sendo que seu nível de precisão dependerá principalmente do número de neurônios, bem como da escolha eficiente do conjunto de exemplos. O formato de uma RNA direta é ilustrado na figura 6.4.

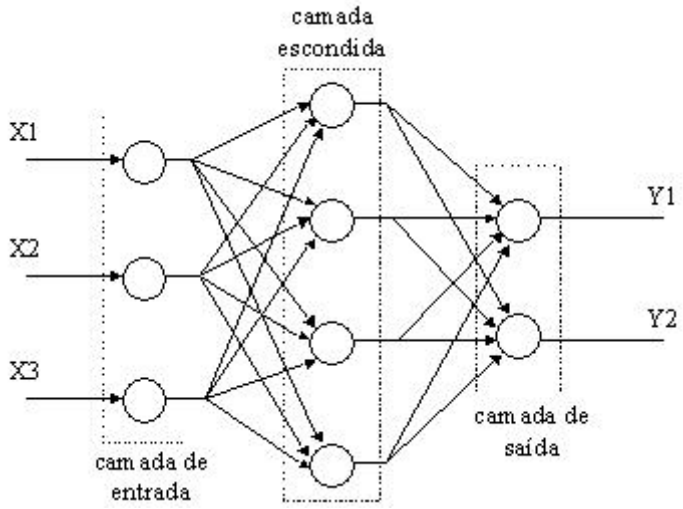

Figura 6.4: Rede Neural Direta

As redes recorrentes são aquelas que contém pelo menos um ciclo de realimentação (figura 6.5). Neste tipo de rede podem ser acoplados elementos de atraso-unitário que em conjunto com as características não-lineares dos próprios neurônios faz a rede operar em um domínio não-linear, fato que segundo [89] provê a este tipo de rede uma capacidade para tratamento de problemas que exijam representações de estados, tais como processamento de linguagem, controle, processamento adaptativo de sinais e predição de séries temporais.

Diferentes arquiteturas de redes recorrentes serão tratadas nas próximas seções, pois devido a sua maior flexibilidade em lidar com dados mais complexos, em domínios não-

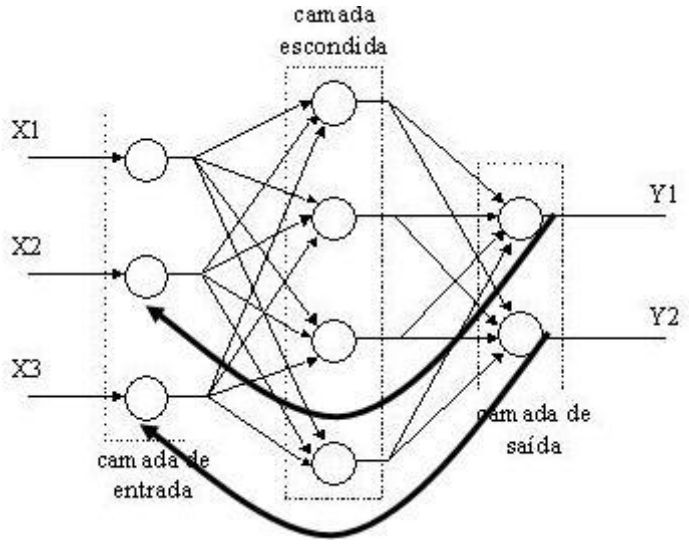

Figura 6.5: Rede Neural Recorrente

lineares, elas serão combinadas em diferentes configurações afim de conseguir implementar o funcionamento dos diferentes autômatos da hierarquia de Chomsky.

Convém ressaltar também que o conhecimento das redes neurais artificiais se dá através da ponderação que os pesos da conexão entre os neurônios de diferentes camadas trocam entre si. Ou seja, encontrar solução para um determinado problema utilizando RNA seria, resumidamente, encontrar a melhor topologia de rede, ajustando corretamente os pesos das conexões entre os neurônios.

6.3 Implementação de Autômato de Estado Finito utilizando Redes Neurais Artificiais

Diversos estudos encontrados na literatura versam sobre a implementação de autômatos finitos utilizando RNAs [5] [43][26][66]. Por considerar os modelos mais robustos, no que tange à capacidade de extrapolação e estabilidade da rede, neste trabalho serão discutidos principalmente os estudos descritos em [77] e [39]. Estes trabalhos diferem principalmente quanto à forma que os pesos estão relacionados na rede. Ao introduzir ao leitor redes de ordem maior do que 1 (seção 6.3.2), as potencialidades das diferentes maneiras de se conectar os pesos serão elucidadas.

Como justificativa para utilização de redes recorrentes para implementação de autô-

matos finitos, pode-se utilizar o teorema proposto em [92].

Teorema: Todo autômato finito pode ser representado por uma rede neural recorrente.

Prova: A prova será feita de modo construtivo utilizando neurônios binários, ou seja, com valores de saída binários. Um autômato finito pode ser descrito através das seguintes equações:

$$x(t+1) = \phi(x(t), u(t)) \tag{6.1}$$

$$y(t) = \gamma(x(t), u(t)) \tag{6.2}$$

A função γ pode ser implementada tomando todas as possíveis saídas e utilizando uma quantidade *n* de neurônios binários para codificar estas saídas. Além disso, deve-se utilizar um conjunto de neurônios cujas saídas codifiquem os estados do autômato. Os neurônios de entrada recebem o valor $u(t)$ no instante considerado e o estado do autômato $x(t)$ atrasados por um conjunto de elementos de retardo z^{-1}. Em tom ilustrativo, este modelo de rede é ilustrado na figura 6.6

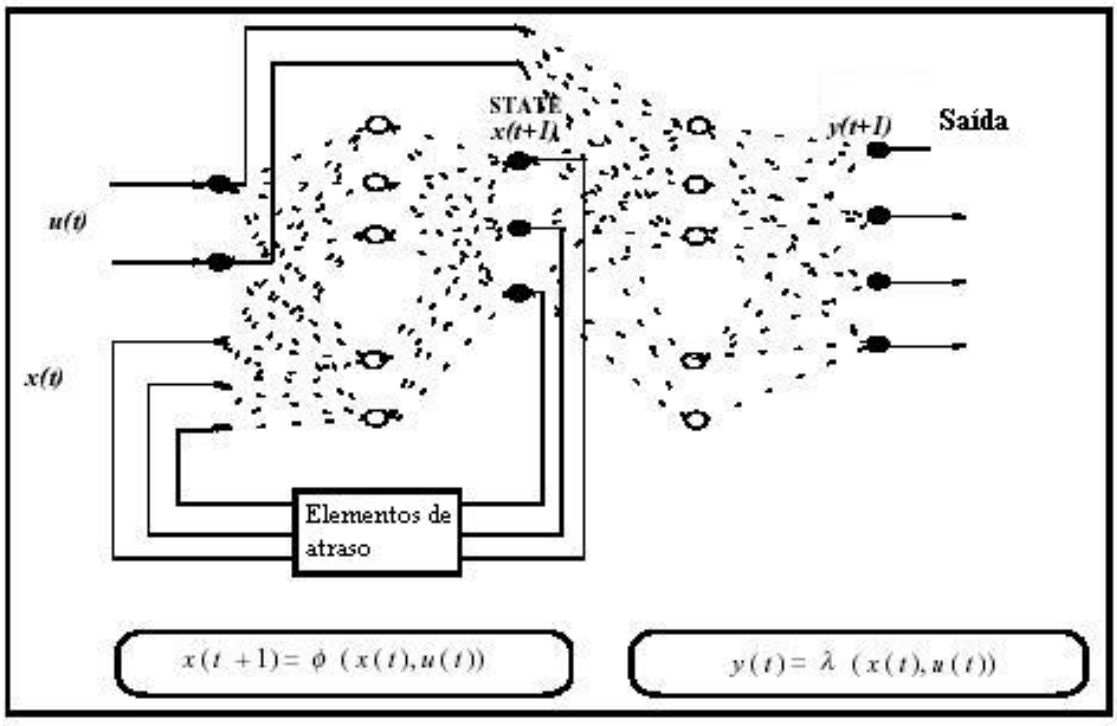

Figura 6.6: Modelo genérico de rede neural recorrente capaz de implementar AFDs

6.3.1 Redes de Primeira Ordem

O conhecimento que uma RNA deve possuir para implementar autômatos finitos centraliza-se nas suas regras de transições, o que pode caracterizar a rede como tendo um conhecimento do tipo K-L (*priori knowledge and learning*)[38].

Desta forma no trabalho de Paolo Frasconi[39] é dada ênfase em um treinamento caracterizado a priori, onde o aprendizado da rede neural não se caracteriza exclusivamente em pares de entrada e saídas, mas sim na arquitetura da rede e em questões relativas a como os pesos são interligados.

A equação genérica de uma rede neural recorrente, a qual define suas características dinâmica e temporal, se dá por uma soma ponderada de estados e/ou entradas, através de uma função discriminante, a qual geralmente tem a seguinte forma:

$$S^{t+1} = F(S^{(t)}, I^{(t)}; W, \beta) \tag{6.3}$$

Onde $S^{(t)}$ e $I^{(t)}$ representam os valores de todos os neurônios de estado e de entrada, respectivamente, no tempo *t*; F é uma função vetorial de mapeamento, geralmente não linear; W são as matrizes de peso que definem a ponderação entre as conexões; β é o conjunto de *biases* dos neurônios de estados.

É importante a percepção de que a equação acima faz um mapeamento do tipo $\delta(q_j, a_k) = q_i$, tal qual um autômato de estado finito [102][60].

Um conjunto de neurônios auto-recorrentes codificam os estados do autômato, entretanto um pré-processamento no autômato original é realizado, onde uma sutil modificação nos estados é feita, afim de deixar cada par de estados consecutivos separado por uma distância de Hamming unitária, de modo a garantir que a rede seja alimentada por vetores regidos por uma base ortonormal, o que evita uma possível desestabilização dos neurônios de estados.

Quando dois estados adjacentes possuem distância de Hamming maior do que 1, o pré-processamento é efetivado, criando novos estados intermediários. Estes estados são temporários e nada mais são que a operação booleana OU entre os dois estados originais. A figura 6.7 ilustra a derivação do autômato original para o intermediário.

Os neurônios da rede de primeira ordem são regidos pela equação 6.4

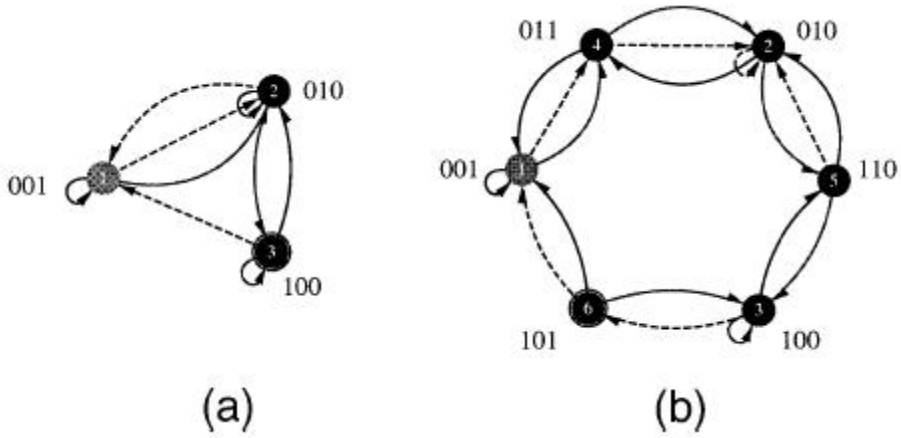

Figura 6.7: Autômato finito original(a) e intermediário(b)

$$S_i^{(t+1)} = \sigma(\alpha_i(t)) = tanh(\frac{\alpha_i(t)}{2}), \;\; onde \;\; \alpha_i(t) = \sum_j V_{ij} S_j^{(t)} + \sum_k W_{ik} I_k^{(t)} \qquad (6.4)$$

Sendo que $S_j^{(t)}$ e $I_k^{(t)}$ representam a saída dos neurônios de estados e de entrada, respectivamente, assim como V_{ij} e W_{ik} são seus pesos correspondentes.

Além dos neurônios recorrentes de estados, a rede possui três camadas diretas que implementam funções booleanas, tendo o intuito de construir as transições do AFD.

Desta forma, quando uma transição de estados do tipo $\delta(q_j, a_k) = q_i$ é executada, o neurônio correspondente ao estado q_j muda de um sinal altamente positivo para um sinal de saída intensamente negativo, e o neurônio correspondente ao estado q_i realiza operação contrária, mudando seu sinal de um intenso negativo para um valor altamente positivo.

A rede recorrente que implementa um AFD é ilustrada na figura 6.8, sendo que os pesos das auto-recorrências dos neurônios de estados permitem uma duração variável da troca destes neurônios, ou seja, permitindo uma estabilização confiável na passagem de um estado para outro.

Os autores provam em seu artigo que esta rede pode implementar qualquer autômato finito com *n* estados e *m* símbolos de entrada usando $2mn - m + 3n$ neurônios e no máximo $m(n^2 + n + m + 4) + 6n$ pesos.

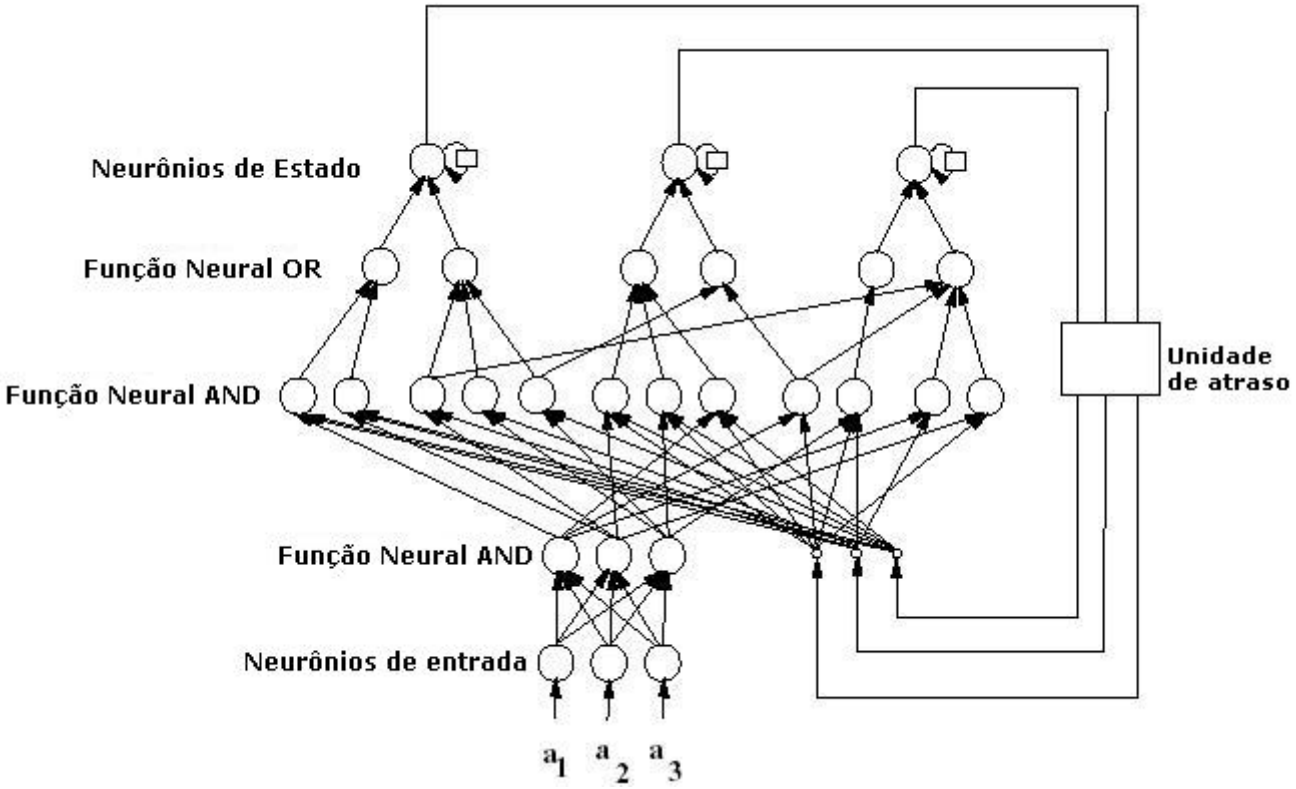

Figura 6.8: Rede Neural Recorrente adaptada de [39] que implementa AFD

6.3.2 Redes de Segunda Ordem

Uma implementação mais eficiente de um AFD utilizando redes neurais recorrentes pode ser vista em [77], onde se leva em consideração a ordem da RNA. Nesta implementação, a codificação dos estados não necessita passar pelo rigoroso pré-processamento do modelo proposto por [39], além do que, utiliza uma quantidade menor de pesos, sem perder potencialidade na indução gramatical. Além disso, trabalhos como o de [43] apontam limitações para cadeias maiores em redes ditas de primeira ordem.

A ordem de uma rede neural imputa a dimensionalidade dos componentes da soma ponderada efetivada pelos neurônios no cálculo do seu net, sendo que esta soma reflete a conectividade da rede, pois está relacionada com os pesos das ligações neurais.

Diferente do modelo anterior, a representação matemática que modela a dinâmica da rede recorrente proposta por [77] é dada pela equação 6.5:

$$S_i^{t+1} = \tau(\alpha_i(t)) = \frac{1}{1 + exp(-\alpha_i(t))}, onde \;\; \alpha_i(t) = b_i + \sum_{jk} W_{ijk} S_j^{(t)} I_k^{(t)} \qquad (6.5)$$

Sendo que b_i é o termo de bias associado com os neurônios recorrentes de estado S_i; I_k denota o neurônio de entrada para o símbolo a_k; e w_{ijk} é o peso correspondente.

Estes pesos de segunda ordem provêem uma representação direta da tripla {estado atual, entrada, próximo estado}, o que é fundamental no aprendizado das regras de transição do AFD. Assim, os neurônios da camada de saída S_i^{t+1} possuem, além dos seus próprios conjuntos de pesos, uma participação direta da informação oriunda dos neurônios que processam a cadeia de entrada e do estado anterior. A rede proposta por [77] é ilustrada na figura 6.9.

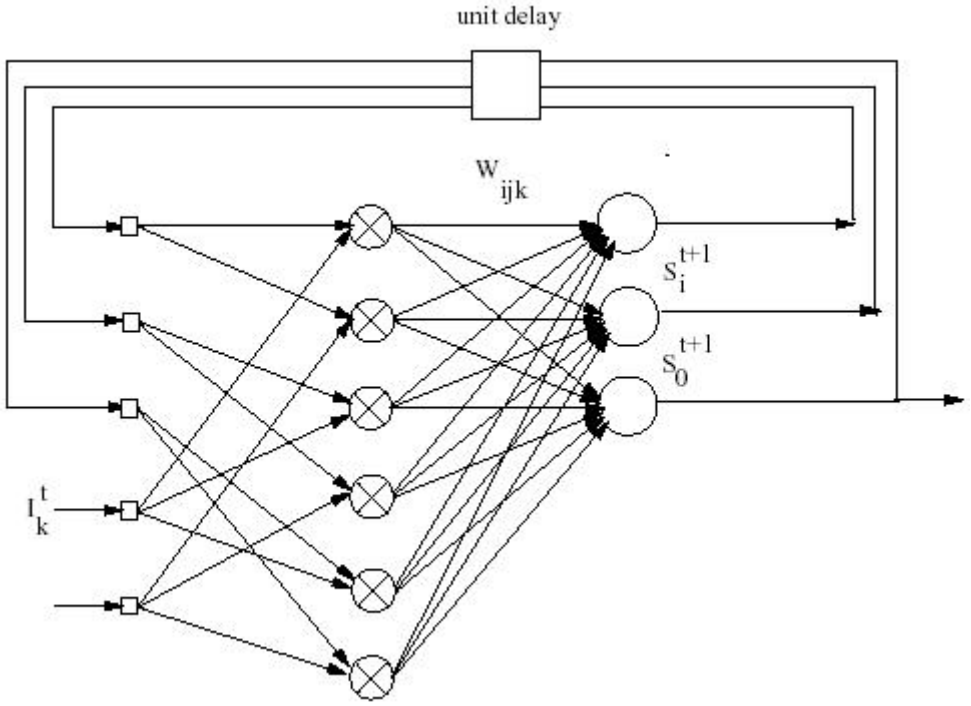

Figura 6.9: Rede Neural Recorrente proposta em [77] que implementa AFD

Para construir esta rede dois passos são seguidos: Programar os pesos da rede, de forma a causar o aprendizado das transições $\delta(q_j, a_k) = q_i$ do AFD e programar a saída de um neurônio S_0 que representa o aceite ou rejeite da rede após o processamento de uma cadeia. Além disso, os neurônios S_j e S_i correspondem aos estados q_j e q_i

Uma constante H é utilizada para atribuir os valores dos pesos, de modo que os neurônios funcionem de maneira similar à rede proposta por Paolo Frasconi [39]. Nesta rede os valores de saída dos neurônios que devem responder por determinado estados tem seus valores setados para alto positivo, e quando não estiverem sido “solicitados”, tem

seus valores setados para baixo negativo.

Neste sentido, os pesos W_{jjk}, W_{ijk} e W_{0jk} e os biases b_i são implementados com a constante H da seguinte forma, onde F é o conjunto de estados finais que o AFD pode assumir:

$$W_{ijk} = \begin{cases} +H & \text{se } \delta(q_j, a_k) = q_i \\ 0 & \text{caso contrário} \end{cases}$$

$$W_{jjk} = \begin{cases} +H & \text{se } \delta(q_j, a_k) = q_j \\ -H & \text{caso contrário} \end{cases}$$

$$W_{0jk} = \begin{cases} +H & \text{se } \delta(q_j, a_k) \in F \\ -H & \text{caso contrário} \end{cases}$$

$b_i = -H/2$ para todos os neurônios de estado S_i

A rede aceita a cadeia caso o valor de resposta do neurônio de saída $S_0(t)$ no final do processamento da cadeia seja maior ou igual a 0,5, caso contrário a cadeia é rejeitada.

Os autores realizaram um experimento com um AFD gerado aleatoriamente de 100 estados, com 1000 strings de comprimento também igual a 1000, utilizando $\sum = \{0, 1\}$. A rede conseguiu ter um acerto de 100%, utilizando $H > 6,3$. Uma criteriosa análise matemática é feita em [77] no que diz respeito a escolha do valor de H.

6.4 Considerações sobre a memória

Na seção anterior foram descritos modelos de redes neurais que não permitem um armazenamento duradouro das ações descritas pelos agentes autônomos. Ou seja, a entidade cognitiva representada pelas redes neurais que implementam autômatos finitos se baseia em ações que dependem apenas do estado atual e da recepção sensória vigente. Desta forma, a única memória disponível é aquela representada pelo estado do autômato e que está codificada na saída dos neurônios de estado da rede.

Nas próximas duas seções os agentes serão implementados com RNAs mais poderosas no que tange a memorização que suas entidades cognitivas são capazes de realizar. Para

tanto, é interessante comentar brevemente sobre algumas características biológicas da memória.

Em trabalhos que datam ao longo do século XIX a memória começou a ser vista como uma entidade que poderia ser subdividida. O filósofo francês Maine de Biran escreveu uma tese chamada *"The Influence of Habit on the Faculty of Thinking"*[34], onde suas conjecturas inferiram que poderiam existir diferentes tipos de memória. De Biram baseou-se em observações nas variações individuais internas das habilidades da memória. Em caráter evolutivo o trabalho descrito em [40] aventou que cada faculdade mental possui uma memória separada. No clássico *"Principles of Psychology"* de William James [48] foi feita a distinção sobre a existência de uma memória primária que possuía curta duração e uma memória secundária que James estatizava como *"the knowledge of a former state of mind after it has already once dropped from the consciousness"*. Entretanto, apenas em meados do século XX [19][45] que a separação das memórias de curta e longa duração se tornou um modelo aceito sobre como a arquitetura de armazenamento do cérebro funciona.

Donald Hebb [45], levantou a hipótese que se um neurônio excita continuamente outro neurônio, algum processo de crescimento ou mudança metabólica ocorre em uma ou em ambas as células, de modo modo que a força da conexão sináptica entre elas aumenta. Então, segundo Hebb, este estímulo contínuo do neurônio seria necessário para que ocorresse um aprendizado de longa duração, enquanto que a memorização de curta duração seria em função das repetidas excitações anteriores à formação da memória de longa duração.

Os trabalhos de [22] e [80] no final da década de 50 também coletaram evidências da existência de uma memória de curta duração na qual o esquecimento mesmo de pequenas quantidades de informação aconteciam em função de um mecanismo intrínseco das próprias conexões que serviam de substrato para a memória de curta duração.

Posteriormente, em [14], através de estímulos elétricos na formação hipocampal de coelhos, conseguiram perceber que um aumento na freqüência da estimulação gera um limiar de potenciação, denominado LTP (*Long Term Potentiation*) que quando atingido se relaciona intimamente com o aprendizado de longa duração. O LTP ainda é motivo de controvérsias na literatura, pois não se sabe ainda se ele é uma espécie de "liberador

de neurotransmissores", ou se aumenta a sensibilidade pós-sináptica, facilitando a transmissão, ou ainda segundo [13], ele atua como mecanismos pré-sinápticos aumentando a probabilidade da liberação do sinal a ser transmitido. Para mais informações sobre o LTP e sobre a revolução causada por ele na comunidade de neurociências, uma leitura em [49] é recomendada.

Com o intuito de tentar curar ataques epilépticos o médico William Scoville executou uma incisão bilateral no lóbulo temporal médio de um paciente conhecido pelo pseudônimo H. M. Entretanto, o paciente apresentou uma grave deterioração crônica em sua memória. [28][71]

O ocorrido com H. M. foi um fato definitivamente isolado, sua deficiência ocasionou o efeito de não conseguir registrar novos fatos na sua memória de longa duração. Além disso, embora, sua operação tenha sido feita quando ele tinha 27 anos, ele não não consegue lembrar de nada desde quando ele tinha 16 anos de idade. Seu desembaraço com linguagem e compreensão é geralmente normal, mantendo fluência na produção verbal e semântica.

É interessante observar que H. M. consegue resolver problemas de raciocínio que envolvam a memória de curta duração (ele foi capaz de resolver o problema das torres de Hanoi [27]), tendo mantido esta intacta depois da operação, aliado ao fato que ele possui memórias da sua infância, mas não consegue lembrar nada recente, leva a crer que as memórias de longa e curta duração possuem um "aparato físico" diferente.

Estudos mais recentes [10] dão à memória de curta duração uma conotação de memória de trabalho[1], a qual refere-se a hipótese que alguma forma de armazenamento temporário das informações é necessária para auxiliar nas atividades cognitivas, as quais incluem percepção, raciocínio e aprendizado.

Desta forma, retomando os **autômatos de pilha**, pode-se considerar que a pilha seria um modelo de memória de trabalho, onde as limitações no que tange o acesso a ela, trariam à tona uma curta duração dos dados na pilha, pois a obrigação do acesso apenas ao topo da pilha geralmente faz com que os dados sejam sobrepostos e por vezes apagados. Justifica-se este idéia pelo fato de que a entidade cognitiva de um AA poderia acessá-la temporariamente, de modo a buscar informações acerca do processamento que ela deseja

[1] Do inglês *Working Memory*

efetivar [107].

Como já foi visto no capítulo 4, os autômatos de pilha são modelos suficientes para representar uma gama interessante de comportamentos robóticos. Seria conveniente buscar inspirações plausíveis biologicamente afim de que se possa simular um autômato de pilha, e conseqüentemente a memória de trabalho (pilha) inerente a ele, utilizando para tanto mecanismos baseados nas redes neurais artificiais que dotariam a entidade cognitiva do robô.

Seguindo o mesmo raciocínio, poderia se pensar novamente em uma **máquina de Turing** como um dispositivo que permite uma memorização de forma contínua e duradoura, pois seu mecanismo irrestrito de acesso à fita de entrada permite que os dados possam ser acessados de modo a manter a memorização permanente das informações obtidas pelo AA, configurando assim, uma memória de longa duração. Desta forma, comportamentos que necessitassem de uma memorização de longa duração poderiam ser mais facilmente representados por uma máquina de Turing.

Neste semblante, surgem então as questões: Como implementar autômatos de pilha, autômatos linearmente limitados e máquinas de Turing em RNAs? qual topologia utilizar? Seriam as estruturas de memórias entidades externas ou internas em uma RNA? No âmago da computabilidade, seriam os modelos equivalentes? As próximas seções versarão sobre estes assuntos.

6.5 Implementação de Autômato de Pilha utilizando Redes Neurais Artificiais

6.5.1 Implementação de Autômato de Pilha Utilizando RNA com Pilha Interna

As informações que recebemos sensorialmente em um nível mais alto de abstração são memorizadas de maneira simbólica pelo nosso cérebro, seja de modo permamente ou não. Associamos signos abstratos às entidades captadas pelos nossos sensores externos. Entretanto, também podemos interagir com nossa memória apenas com o nosso pensamento, não tendo necessariamente um estímulo externo, ou seja, utilizamos o mesmo aparato

físico para representar entidades sejam elas externas ou internas, e isto pode evidenciar que o cérebro humano tem um mecanismo interno de células que conseguem memorizar diferentes padrões de maneira simbólica.

Desta forma, seria conveniente se inspirar biologicamente para construir redes neurais artificiais que modelem autômatos de pilha, onde a entidade que representa a memória de curta duração seja interna às próprias conexões neurais inscritas na topologia da rede.

Para demonstrar a viabilidade da implementação de autômatos de pilha utilizando redes neurais artificiais, será dada uma descrição do funcionamento da arquitetura de RNA chamada de Neural Network Stack (NNS), proposta por [24].

Esta arquitetura tem como apanágio o fato de uma rede de perceptrons conseguir agir como uma função de mapeamento binário[23]. Na definição 6.1 a função de mapeamento binário é definida.

Definição 6.1. Dado um conjunto U de k vetores de entrada binários $u_1, ..., u_k$ de dimensão *m* e um conjunto V de k vetores de saída binários $v_1, ..., v_k$ de dimensão *n*. A **função de mapeamento binário** é dada por $g : U \rightarrow V$, de tal sorte que $g(u_i) = v_i$, para $1 \leq i \leq k$.

6.5.1.1 Binary Mapping Perceptron Module (BMP)

Seja um conjunto A de k vetores de entrada binários $a_1, ..., a_k$ de dimensão *m*, onde $a_h =< a_{h1}, ..., a_{hm} >$ e $a_{hi} \in \{0, 1\}$ para $1 \leq h \leq k \ \& \ 1 \leq i \leq m$, e um conjunto D de k vetores binários de saídas desejadas $d_1, ..., d_k$ de dimensão *n*, onde $d_h =< d_{h1}, ..., d_{hn} >$ e $d_{hj} \in \{0, 1\}$ para $1 \leq h \leq k \ \& \ 1 \leq j \leq n$.

Um BMP pode agir como uma função de mapeamento binário, vista na definição 6.1. Um módulo BMP tem *m* neurônios de entrada, *k* neurônios na camada escondida e *n* neurônios de saída. Para fazer o mapeamento binário de cada par ordenado (a_h, d_h), onde $1 \neq h \neq k$, um neurônio escondido *h* é criado com limiar de ativação $|a_h|^2 - 1$. Os pesos W_{ih} e W_{hj} são $2a_{hi} - 1$[2] e d_{hj}, respectivamente.

A figura 6.10 apresenta o funcionamento de um BMP, a função de ativação f_0 dos neurônios de saída é a função identidade, ou seja, $f_0(x) = I(x) = x$ e para os neurônios da camada escondida, a função de ativação f_h é dada por:

[2]Algebricamente $|a_h|^2 - 1$ é equivalente a $a_{h1} - \overline{a}_{h1}$, conforme visto na figura 6.10

$$f_h(x) = \begin{cases} 1 & \text{se } x > 0 \\ 0 & \text{caso contrário} \end{cases}$$

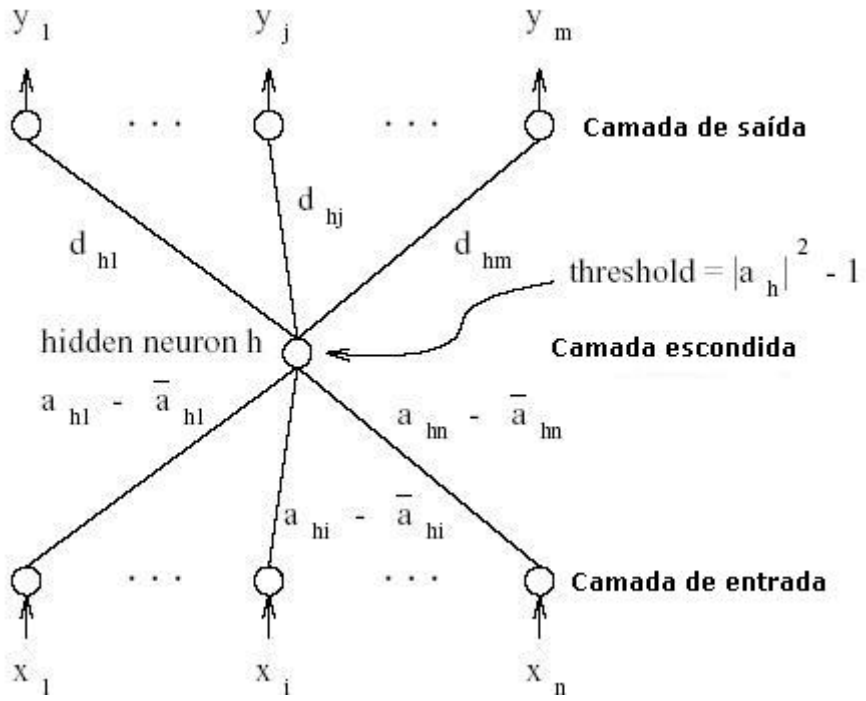

Figura 6.10: Implementação do perceptron para o mapeamento binário (a_h, d_h), adaptada de [24]

Para dado vetor de entrada a_h, apenas o neurônio escondido *h* produz uma saída 1, sendo a saída de todos os outros neurônios igual a zero. Assim, o valor computado no neurônio de saída *j* é d_{hj}, e conseqüentemente o vetor binário de saída será dado por $< d_{h1}, ..., d_{hn} >= d_h$. Além disso, devido a apenas um neurônio da camada escondida ter valor de saída 1 (tendo os outros setados a zero), a computação da camada de saída é habilitada exatamente por um neurônio da camada escondida.

6.5.1.2 Características e Funcionamento da NNS

A Neural Network Stack é organizada em módulos, sendo composta principalmente de dois módulos de BMP, um que controla a cabeça de leitura do topo da pilha (*pointer control module*) e outro que armazena os dados que se encontram na pilha (*stack memory module*). Na figura 6.11 a arquitetura global é mostrada.

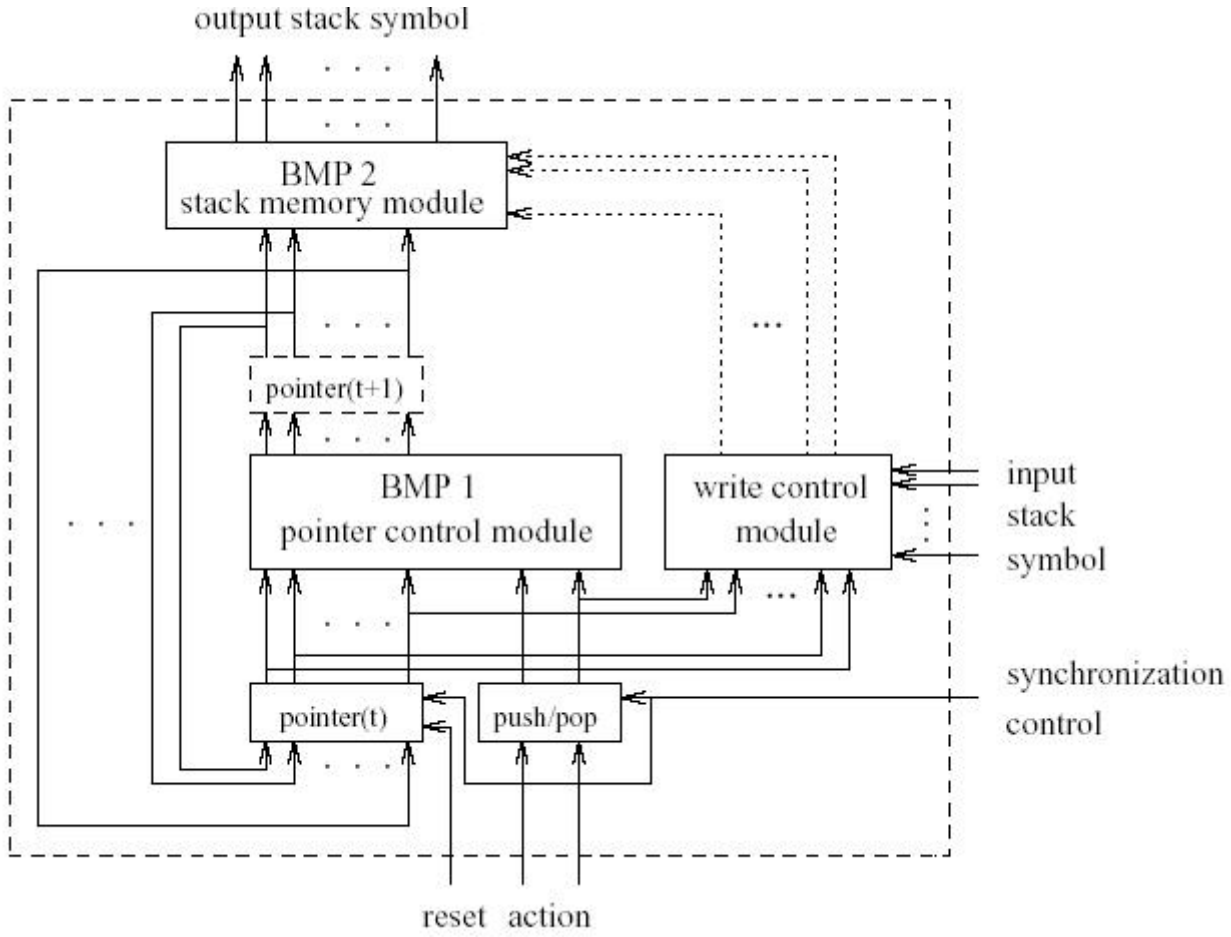

Figura 6.11: Neural Network Stack, extraída de [24]

O módulo BMP1 (*pointer control module*) controla o movimento do cabeçote de leitura da pilha que é incrementado quando a ação for push e decrementado para um pop. Este módulo é composto por uma rede multi-layer perceptron[3] de duas camadas. Esta entidade possui m + 2 neurônios de entrada, onde *m* codifica os 2^m possíveis valores que podem estar no topo da pilha e os outros dois neurônios codificam a ação sobre a pilha (push=01, pop=10 e no-op=00). A camada oculta possui 3×2^m, e a camada de saída possui m neurônios que representam o novo valor no topo da pilha ((*pointer(t+1)*) após uma das operações ser efetivada (push, pop ou no-op).

Em cada incremento ou decremento no topo da pilha é feito um mapeamento binário que utiliza um neurônio na camada escondida para corresponder ao padrão dado.

Já o módulo BMP2 (*stack memory module*) utiliza *m* neurônios de entrada, *n* neurônios

[3]Outra denominação para redes diretas

de saída e 2^m neurônios na camada escondida, os quais permitem armazenar 2^m símbolos da pilha em em 2^m posições ((*pointer(t), pointer(t-1), pointer(t-2),..., pointer(t-m)*). Os símbolos da pilha são armazenados no BMP2 através dos neurônios da camada oculta, fazendo assim, um mapeamento funcional binário.

A estrutura da rede também possui um módulo que é responsável pela escrita de dados na pilha, o chamado *Write Control Module*. Ele possui duas entradas binárias, a primeira diz respeito a indicação de qual o símbolo que está no *buffer pointer(t)* e uma das entradas do buffer push/pop que indicará (caso tenha o valor 1) se é o momento de se escrever na pilha. Já o outro conjunto de entradas deste módulo é a codificação binária do símbolo da pilha a ser inserido.

O símbolo é inserido diretamente no neurônio correspondente ao valor atual de *pointer(t)*, de modo a armazenar este símbolo na pilha em sua respectiva ordem se entrada.

É necessário um controle de sincronização que se preocupa com que a gravação do símbolo pelo módulo de escrita seja executada antes dos sinais de BMP1 serem passados ao BMP2. Mais detalhes sobre a implementação deste conjunto de retardos para contornar este possível problema podem ser encontrados em [24]

6.5.2 Implementação de Autômato de Pilha Utilizando RNA com Pilha Externa

Retomando o estudo sobre a memória de trabalho [10] do ser humano, algumas experiências foram feitas feitas em [84], onde diferentes indivíduos foram testados com o objetivo de recuperar palavras não relacionadas de uma lista previamente apresentada. Mesmo sem importar a ordem de recuperação, os indivíduos testados conseguiam recuperar as palavras recentes com relativa facilidade, entretanto quando algum fator de distração era acrescido, eles perdiam esta capacidade. Além disso, diversos trabalhos [10][30][53] levam a crer que a memória de trabalho possui limitação na sua capacidade de armazenamento.

Acrescentando o fator ordenação, ou seja, tentar recuperar as palavras na mesma ordem em que foram apresentadas, tornaria os resultados bastante limitados, pois temos uma capacidade de ordenação reversa bastante restrita. No entanto, a utilização de uma

memória de trabalho externa superaria esta limitação.

Aliado a isto, o homem conseguiu desenvolver-se intelectualmente graças ao advento da escrita, pois com ela o conhecimento pode ser repassado entre gerações, permitindo uma memorização “eterna” do conhecimento, sem contar que, para fins de cálculos ela se torna um mecanismo essencial, pois provavelmente, devido nossas limitações na nossa memória de curta duração, não conseguimos realizar facilmente uma divisão ou multiplicação para algarismos acima de dois digitos no sistema decimal.

Ademais, ao se pensar em realizar operações elementares (mesmo de soma) utilizando o sistema binário, não dispondo de um mecanismo de armazenamento exterior (como a escrita) onde se possa guardar os estados intermediários das operações, acabaríamos provavelmente tendo um resultado drástico das operações.

Assim, da mesma forma que os seres humanos possuem utensílios como papel e caneta para auxiliar nos cálculos, torna-se viável a utilização de uma arquitetura de rede neural que possua esta ferramenta estereotipada externamente. Neste sentido, da mesma maneira que se conjecturou que a pilha de um AP opera como uma memória de trabalho, pode-se pensar em arquiteturas de redes neurais que façam o acesso a uma estrutura de pilha externa, a fim de ser capaz de implementar o funcionamento de um autômato de pilha.

Em [103] foi desenvolvido um modelo híbrido de rede neural chamado de *Neural Network Pushdown Automaton* (NNPDA), o qual consiste da junção de uma rede recorrente com uma estrutura de pilha externa. Segundo os autores, este modelo é capaz de aprender e reconhecer algumas classes de linguagens livres de contexto.

Uma tentativa de se aumentar a potencialidade do modelo de rede recorrente de segunda ordem visto na seção 6.3 seria aumentar em tamanho a arquitetura geral da rede, ou fortificar a precisão dos neurônios, na medida em que apresenta-se a rede uma linguagem livre de contexto. No entanto em [112], implementações inferem que este tipo de rede consegue reconhecer linguagens com cadeias de comprimentos limitados.

Segundo os autores, um NNPDA depois de treinado é capaz de reconhecer todas as cadeias não apresentadas a rede de uma dada linguagem gerada por uma GLC desconhecida. Para isto, estas cadeias devem alimentar a rede um caractere por vez, de modo que uma função erro no final de cada cadeia decidirá sobre o aceite ou rejeite de determinada cadeia.

A rede recorrente consiste de neurônios de entrada, estado, ação sobre a pilha (push, pop, no-op) e de leitura sobre a pilha. A rede recorrente faz um mapeamento do estado interno atual S^t, do símbolo de entrada I^t e do símbolo lido da pilha R^t para a saída contendo o próximo estado S^{t+1} e a ação sobre a pilha A^{t+1}. As operações sobre a pilha irão atualizar a próxima leitura do neurônio que representa o símbolo lido da pilha R^{t+1}. Ao final da cadeia o conteúdo do estado interno e a composição da pilha irão determinar o reconhecimento ou não da cadeia. A figura 6.12 ilustra a estrutura geral desta rede

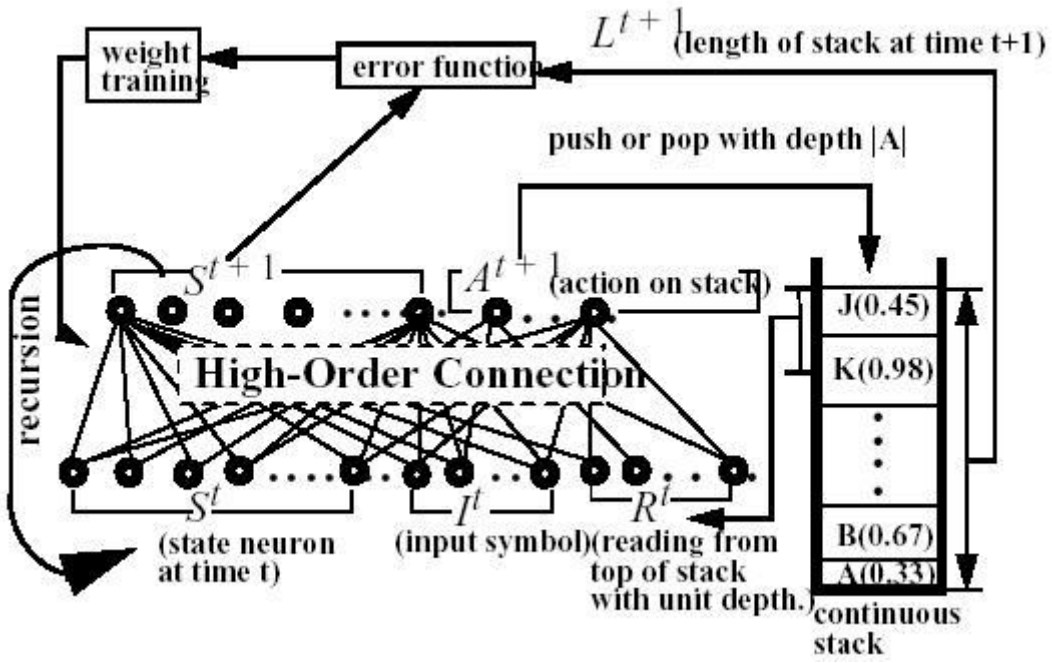

Figura 6.12: Neural Network Pushdown Automaton, extraída de [103]

As regras de transição de um autômato de pilha são representadas por conexões de terceira ordem, de forma a fazer um mapeamento de $\{S^t \times R^t \times I^t\} \rightarrow (S^{t+1}, A^{t+1})$.

De fato, considerando I^t=(1,0,0), (0,1,0) e (0,0,0) que representam os símbolos a, b, c, e dois estados S^t=(1,0) e (0,1), a transição $\{S_j^t, R_k^t, I_l^t\} \rightarrow S_i^{t+1} ou\ A_i^{t+1}$ pode ser codificada em duas matrizes de quatro dimensões W_{ijkl}^s e W_{ijkl}^a.

Desta forma os pesos podem ser ajustados para representar a regra $\{S_j^t, R_k^t, I_l^t\} \rightarrow S_i^{t+1}$, fazendo com que $W_{ijkl}^s = 1$ e $W_{mjkl}^s = 0$, sendo $m \neq i$. De maneira similar $W_{ijkl}^a = [-1, 0, 1]$ representa um mapeamento para as ações sobre a pilha: *push*, *pop*, *no-op*.

Finalmente as saídas da rede recorrente que controla o NNPDA podem ser dadas pelas

seguintes equações:

$$S_i^{t+1} = g(\sum_{j,k,l} W_{ijkl}^s S_j^t R_k^t I_l^t + \theta_i^s) \tag{6.6}$$

$$A_i^{t+1} = f(\sum_{j,k,l} W_{ijkl}^a S_j^t R_k^t I_l^t + \theta_a^s) \tag{6.7}$$

De modo a se adaptar ao treinamento utilizando algoritmos baseados em gradiente descendente (ex: *backpropagation*) é necessário que as variáveis que operam sobre a pilha estejam em um domínio contínuo.

Neste contexto, os símbolos da pilha e as ações sobre ela (*push, pop, no-op*) possuem valores contínuos, com comprimentos L associados a eles, como mostra a figura 6.13. Assim, cada símbolo é interpretado como tendo comprimento $1 \geq L \geq 0$. Esses símbolos contínuos são gerados justamente pelas operações contínuas sobre a pilha, que tem valores provenientes do neurônio de ação A^t.

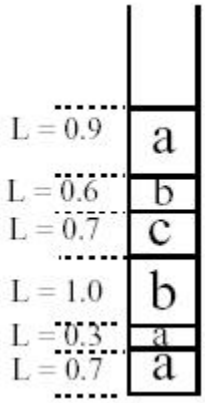

Figura 6.13: Pilha utilizada no NNPDA

As operações sobre a pilha ao serem passadas ao domínio contínuo no neurônio de ação, inserem uma incerteza sobre sobre a ação a ser tomada (*push, pop, no-op*), de modo que esta incerteza é justamente representada pelo comprimento dos símbolos discretos a serem retirados ou colocados na pilha. Isto implica que em cada iteração apenas uma parte de um símbolo discreto (uma letra α qualquer, por exemplo) com um comprimento $|A_i^t|$ é retirado ou colocado na pilha. As operações sobre a pilha, então, se comportam da seguinte forma:

- Se $A_i^t > \varepsilon$ então realizar-se-á um push.

- Se $A_i^t < -\varepsilon$ então a realizar-se-á um pop.

- Se $-\varepsilon \leq A_i^t \leq \varepsilon$ então não se realizará nenhuma operação (no-op).

Onde ε um número próximo de zero.

No que tange à leitura da pilha, que será o valor alimentado em R^t, a cada passo é lido da pilha um símbolo contínuo de comprimento igual a 1, isto evita possíveis perturbações e descontinuidades. Para mais detalhes o leitor interessado é convidado a ler [103].

Para exemplificar o funcionamento pode-se retomar à figura 6.13. Considerando a situação em que $A^t = -0,9$, o símbolo "a" no topo da pilha é retirado. Conseqüentemente, a próxima leitura de R^t conterá o símbolo "b", com comprimento 0,6 e "c"com comprimento 0,4.

Por outro lado, se devido a uma pequena pertubação, o valor de A^t passar a ser -0,899, a próxima leitura R^t seria a (L=0,001), b (L=0,6) e c (L=0,399). Estes valores podem ser interpretados como a probabilidade de qual símbolo R^t obteve da pilha. Desta forma, a probabilidade de ter lido um símbolo "a"(tomando o modo discreto) repetidamente é muito baixa, o que caracteriza uma tolerância a falhas desta abordagem.

Em suma, o funcionamento do NNPDA pode ser resumido através de uma cadeia *aaabbbf* (o símbolo "f" representa o final da cadeia) alimentando um símbolo de cada vez os neurônios de entrada I^t. Este símbolo pode ser colocado na pilha, ou algum outro símbolo pode ser retirado da pilha com um comprimento $|A^t|$. Ao atingir o símbolo f, o NNPDA irá gerar a saída adequada que indicará se a cadeia aaabbb é reconhecida ou não.

Consegue-se obter resultados satisfatórios desta arquitetura para linguagens como a dos parênteses balanceados, $a^n b^n$ e wcw^r, com um conjunto de treinamento de no máximo 512 exemplos. Assim, este modelo obteve uma validade eficiente no que tange sua capacidade de generalização para corretamente classificar grandes conjuntos de cadeias não vistas. Novamente o leitor interessado é convidado a consultar [77] e [31] para mais detalhes sobre as simulações.

6.6 Linguagens Sensíveis ao Contexto e RNAs

Embora parcos no que se refere aos resultados alcançados, existem estudos de redes neurais para implementar linguagens sensíveis ao contexto [42], [16], [101], [15]. Entretanto, em sua vasta maioria os experimentos se baseiam na linguagem $a^n b^n c^n$

Em caráter ilustrativo será comentado o trabalho proposto em [15] que utiliza a arquitetura de RNA chamada Rede Sequencial em Cascata (RSC).

Um exemplo da RSC é visto na figura 6.14. Esta rede possui três neurônios de entrada, dois de estados e três neurônios de saída. Cada ativação do estado anterior é multiplicada pela saída dos neurônios de entrada X^t. Os produtos alimentam a camada de saída através dos conjuntos de pesos W e V.

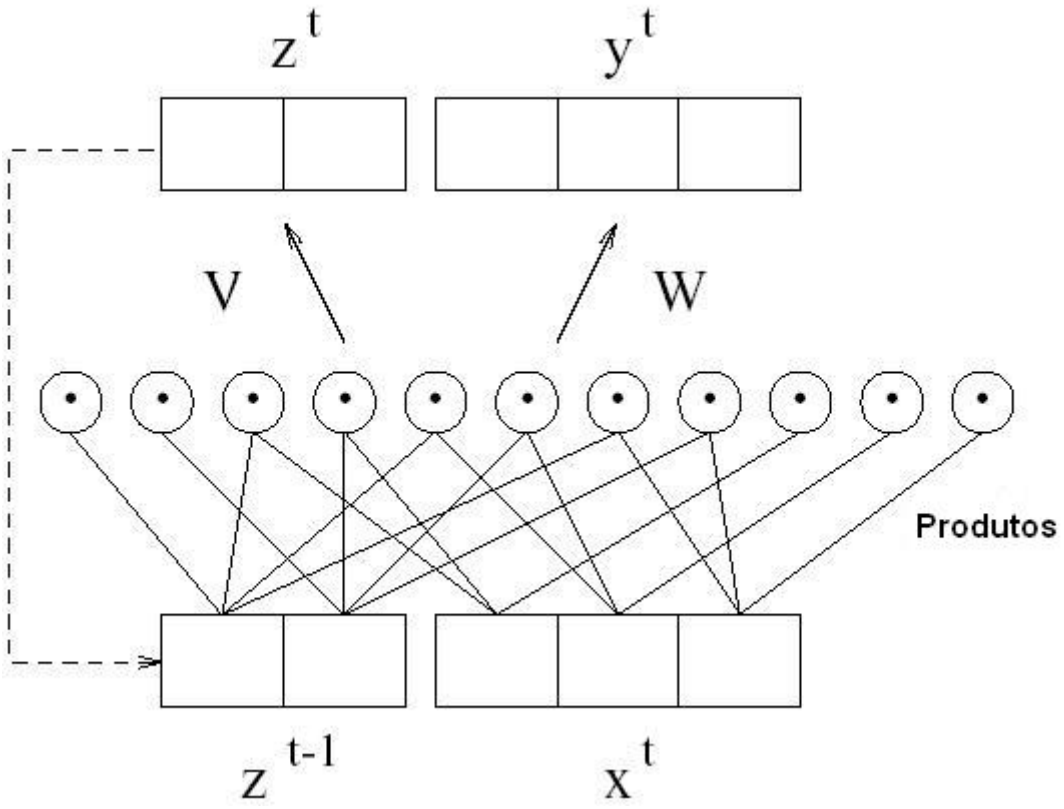

Figura 6.14: Rede Sequencial em Cascata, extraída de [15]

A saída de um neurônio *i*, no instante *t* pode ser definida pelas equações 6.8 e 6.9

$$y_i^t = f(\sum_{j,k} W_{ijk} z_j^{t-1} x_k^t + \sum_j W_{ij\theta} z_j^{t-1} + \sum_k W_{i\theta k} x_k^t + W_{i\theta\theta}) \tag{6.8}$$

$$z_i^t = f(\sum_{j,k} V_{ijk} z_j^{t-1} x_k^t + \sum_j V_{ij\theta} z_j^{t-1} + \sum_k V_{i\theta k} x_k^t + V_{i\theta\theta}) \tag{6.9}$$

Os indices *i, j, k* representam a conexão com a saída dos neurônios Z^t e Y^t, os estados anteriores e a entrada corrente, respectivamente.

Os produtos entre os neurônios de entrada (Z^{t-1} e X^t) e o próximo estado (Z^t) são conectados através de pesos de segunda ordem W_{ijk}, $W_{ik\theta}$ e $W_{i\theta k}$ que alimentam a *ith* saída com os *jth* neurônios de estado e os *kth* neurônios de entrada.

Em [15], diversos experimentos foram feitos com intuito de fazer uma RSC aprender a linguagem sensível ao contexto $a^n b^n c^n$. Os melhores resultados foram apresentando um conjunto de centenas de strings com taxa de aprendizado bastante baixa (com objetivo de atravessar a superfície de erro cautelosamente), tendo a rede conseguido generalizar para todas as cadeias até $n = 18$.

6.7 Algumas Considerações

Como se pode perceber, os resultados de RNAs que aprendam ou implementem o funcionamento de um autômato linearmente limitado ainda estão bastante aquém do desejado, sendo ainda um campo promissor para novos investimentos da comunidade científica.

Ademais, mesmo no que tange as linguagens livres de contexto, ainda não se tem um modelo global, que seja tanto de pilha externa ou interna, que garanta o aprendizado de qualquer linguagem desta categoria.

Também é importante ressaltar que um maior entendimento de cunho matemático de como deve se comportar a dinâmica das redes neurais para utilização em inferência e reconhecimento gramatical se faz necessário, a fim de se encontrar modelos de rede que consigam ter um controle estável e adequado do funcionamento holístico da rede.

6.8 Computabilidade Neural

Tendo em vista que a principal contribuição do presente trabalho é a possibilidade de se representar comportamentos robóticos através da hierarquia de Chomsky, dotando a entidade cognitiva dos AAs com redes neurais artificiais que consigam implementar os

autômatos reconhecedores destas linguagens, é relevante algum comentário sobre até onde as RNAs podem chegar, no que tange sua computabilidade.

Sendo a máquina de Turing o dispositivo computacional capaz de reconhecer qualquer linguagem da hierarquia de Chomsky, se existirem modelos de redes neurais que consigam apresentar equivalência com a máquina de Turing, corroborar-se-á a viabilidade da implementação das linguagens de Chomsky utilizando RNAs.

Nos trabalhos de Hava Sielgelmann e Eduardo Sontag [98], foi utilizada uma rede neural recorrente de primeira ordem, com pesos contidos no domínio dos racionais, por não requererem muita precisão, e pela conveniência que o Conjunto de Cantor traz ao ser utilizado na prova da equivalência entre a MT e às RNAs. Eles conseguem chegar, através de provas matemáticas, a conclusão que uma rede contendo no máximo 886 neurônios é capaz de computar qualquer função parcial recursiva.

Jordan Pollack em sua tese de doutoramento [83] desenvolveu uma rede recorrente, que denominou de *"Neural Machine"*, a qual era composta por um número finito de neurônios conectados através de conexões de alta ordem. Em seus estudos ele chegou à universalidade das RNAs, tal qual a máquina de Turing.

Também se encontra na bibliografia trabalhos que utilizam um número infinito de neurônios, como observado em [114], onde o rede com neurônios lineares e em quantidade ilimitada tem o intuito de representar todas as possíveis configurações de uma máquina de Turing de múltiplas fitas.

No artigo [97] um tipo de rede recorrente chamada NARX com um número limitado de neurônios, sendo baseada em modelos autoregressivos não lineares, também teve suas potencialidades provadas como equivalentes às da máquina de Turing. Nestas redes, a recorrência se dá diretamente dos neurônios da camada de saída para os da entrada. Além disso, em [61] uma análise sobre o questionamento da necessidade das redes recorrentes precisarem de realimentação dos neurônios da camada escondida afim de representar os estados internos da MT também são levados em pauta. Ainda em [47], se demonstra que o aprendizado baseado em métodos que utilizam gradiente descendente são mais eficientes em redes do tipo NARX, pois estas não necessitam de estados intermediários.

De grande relevância também pode ser considerado o trabalho proposto em [12], onde prova-se que as redes neurais artificiais podem implementar funções booleanas, usadas

nos computadores baseados em instruções, e sendo estes modelos capazes de simular a máquina de Turing, por dedução também pode-se inferir que as RNAs são modelos equivalentes à MT.

Finalmente em [2], um modelo de rede neural que utiliza fita externa é proposto. Neste modelo, utiliza-se uma rede neural lógica [56] que implementa o controle finito da MT, e que acessa uma fita externa, de modo a conseguir implementar uma máquina de Turing.

6.9 Incremento na Arquitetura PiramidNet

Na seção 2.5.4 foi comentada a arquitetura PiramidNet que tem como característica principal a utilização de redes neurais hierárquicas para a implementação de comportamentos robóticos.

Esta arquitetura inspira-se no modelo hierárquico e modular em que o cérebro opera. Dentre outras evidências, foi dado o exemplo do tratamento dado ao cérebro no que se refere aos estímulos visuais como forma, cor, movimento e posição que são processados por mecanismos neurais anatomicamente separados, organizados no magno celular e parvo celular. [17].

Todavia, até o presente momento a arquitetura PiramidNet permitia em seu nível mais alto o controle através de redes recorrentes que eram capazes de implementar apenas autômatos finitos.

No decorrer deste trabalho, foi visto que diferentes autômatos se tornam necessários para tarefas mais complexas, como o comportamento de ir e voltar pelo mesmo caminho (subseção 4.5.2), e conseqüentemente diferentes estruturas de redes neurais que foram vistas no decorrer deste capítulo se tornam essenciais afim de conseguir implementar outros autômatos, enquadrados em camadas inferiores da hierarquia de Chomsky.

Neste contexto, propõe-se um acréscimo de camadas na arquitetura PiramidNet, onde redes neurais que consigam se comportar como autômatos de pilha e autômatos linearmente limitados serão acrescentadas acima da camada de redes recorrentes que conseguem implementar apenas autômatos finitos. Desta maneira, a nova formulação da arquitetura PiramidNet pode ser vista na figura 6.15.

Por fim, este acréscimo de novas camadas permite que a arquitetura PiramidNet seja

um modelo capaz de controlar uma gama maior de diferentes comportamentos robóticos, baseado em um modelo plausível da maneira pela qual o cérebro opera.

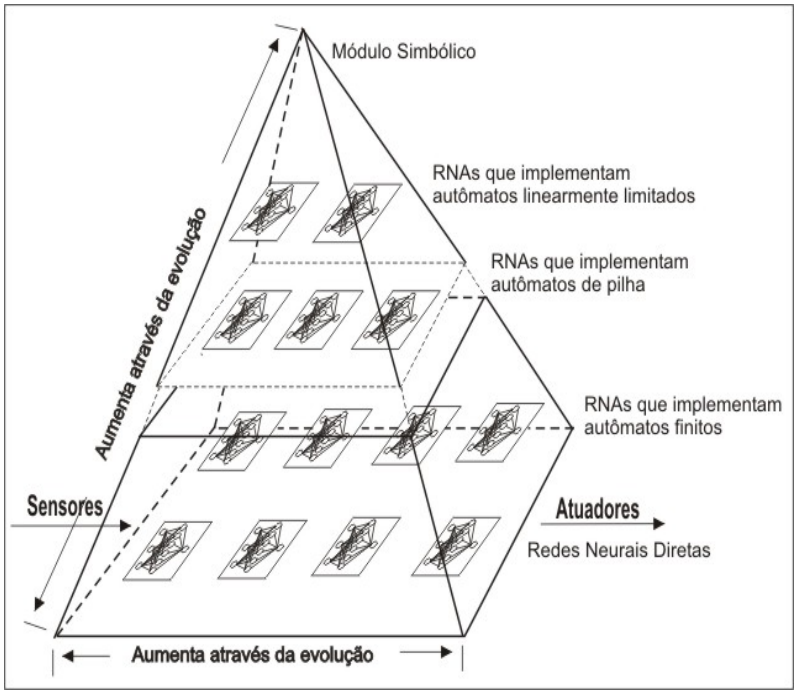

Figura 6.15: Atualização da Arquitetura PiramidNet

Capítulo 7

Considerações Finais

7.1 Conclusões

Este trabalho teve seu sustentáculo englobando diversas áreas da computação, onde se procurou reunir idéias da teoria da computação, linguagens formais e autômatos, redes neurais artificiais, e, realizando, por vezes, paralelos com as ciências biológicas, em especial características do sistema nervoso dos seres vivos.

Esta miscelânea de componentes, teve como objetivo primordial chegar a modelos formais da representação de comportamentos robóticos desempenhados por um agente autônomo sob a ótica da robótica baseada em comportamentos.

Pela necessidade de se encontrar um modelo que fosse capaz de formalizar os comportamentos robóticos, preservando as características de complexidade de diferentes comportamentos, tal qual os estudos de etologia fazem nos comportamentos dos seres vivos, suscitou-se neste trabalho a utilização das linguagens da hierarquia de Chomsky. Assim, foram vistas diversas aplicações das linguagens de Chomsky como modelos capazes de representar comportamentos robóticos.

As linguagens de Chomsky se comportaram de fato como um poderoso meio de se obter a formalização na representação de comportamentos robóticos, onde as gramáticas correlatas a cada nível da hierarquia de Chomsky foram utilizadas como um mecanismo de geração de comportamentos, assim como, os respectivos autômatos reconhecedores se portaram como um eficiente dispositivo de visualização gráfica e funcional dos comportamentos desempenhados por um agente autônomo.

Observou-se na literatura que grande parte das linguagens para programação de robôs, mantém sua representação coarctada a elementos como arquiteturas de hardware [8], linguagens de programação específicas [51][70], ou mesmo através de linguagem natural e aprendizado por imitação [76][20], além disso algumas destas linguagens ainda representam seus comportamentos robóticos em um nível mais alto por autômatos finitos. Entretanto, esta representação, além de não permitir uma generalização para um número infinito ou desconhecido de marcos, também apresenta uma notação que pode se tornar muito grande e de difícil entendimento quando se tenta enfatizar todas as situações em que um robô pode se encontrar.

Neste âmbito da complexidade comportamental, se percebeu neste trabalho que os autômatos de pilha e os autômatos linearmente limitados conseguem efetivar uma notação mais robusta e compacta para a representação dos comportamentos robóticos.

Isto advém, obviamente da premissa cerne deste trabalho de que as linguagens de Chomsky atuam como modelos de representação de comportamentos robóticos, haja vista que os autômatos reconhecedores destas linguagens nada mais são do que mecanismos de verificação de que uma dada cadeia (conjunto de ações), está de acordo com as regras de formação de uma dada linguagem (comportamento).

Também se chegou a uma nova definição formal de agentes autônomos. Nesta definição os AAs foram enfocados como uma máquina de Turing, o que forneceu uma definição dos AAs como mecanismos de computação, capazes de receber dados oriundos dos sensores através da fita de entrada da MT, efetivar um mapeamento cognitivo via produções contidas nas regras de transição de estados da MT, e emergir um comportamento para o ambiente (fita) através do resultado computado pela MT. É importante observar que com esta definição garante-se que os AAs, enquanto mecanismos de computação, possuem a limitação de resolver apenas problemas computáveis sob o prisma da tese de Church-Turing.

Uma importante conjectura postulada neste trabalho, é o comportamento da pilha dos autômatos de pilha como metáfora da memória de trabalho existente em alguns seres vivos [22][80], onde esta estrutura (pilha) interage como um dispositivo de auxílio à computação. Entretanto, seu método de acesso torna-se um limitador à memorização dos marcos de forma contínua, infinita e de fácil recuperação, independente da posição.

Sob a mesma ótica, comportamentos robóticos, aprendidos em tempo de execução, que necessitem de uma memorização permanente devem ser modelados através de uma máquina de Turing, pois o mecanismo de acesso a sua fita infinita permite que os dados sejam armazenados permanentemente na fita, podendo ser acessados a qualquer momento, não tendo necessariamente o risco de símbolos serem sobrescritos ou apagados.

Após a representação dos comportamentos robóticos através das linguagens da hierarquia de Chomsky, tornou-se necessário um mecanismo para implementar estes comportamentos. Para isto, foram utilizadas as redes neurais artificiais.

Desta forma, foi feita uma intensa revisão bibliográfica com intuito de encontrar modelos de RNAs capazes de agir como os autômatos propostos na hierarquia de Chomsky.

Assim, esta pesquisa indicou que as redes neurais que implementam autômatos finitos têm obtidos resultados suficientemente precisos, e redes neurais recorrentes simples conseguem implementar vasta maioria destes autômatos.

Para autômatos de pilha, os modelos seguem duas vertentes: com pilha interna e externa. Ambos os modelos, obtém relativo sucesso em classificar cadeias não vistas no treinamento para determinadas classes de linguagens, entretanto um maior aprofundamento sobre as características das dinâmicas envolvidas em reconhecer linguagens livres de contexto, ainda se faz necessário.

Já redes neurais que implementam autômatos linearmente limitados, ou seja, autômatos reconhecedores de linguagens sensíveis ao contexto, ainda se encontram com resultados incipientes, entretanto, se mostrou que existem trabalhos que conseguem classificar com sucesso linguagens clássicas como $a^n b^n c^n$.

Finalmente, foi desenvolvida uma atualização da arquitetura de controle robótico PiramidNet. Este modelo, que utiliza RNAs dispostas hierarquicamente, se baseava em redes neurais que eram capazes de modelar apenas comportamentos que podiam ser implementados com autômatos finitos. Assim, sua estrutura foi alterada, o que permitiu a junção de mais duas camadas de redes neurais capazes de implementar autômatos de pilha e autômatos linearmente limitados, o que fornece a esta arquitetura um maior poderio na implementação dos comportamentos robóticos.

7.2 Trabalhos Futuros

- Embora tenha sido demonstrada a viabilidade da representação dos comportamentos robóticos utilizando as linguagens de Chomsky, este trabalho não teve o cunho de criar nova teoria, com devidas provas. Assim, um trabalho que se proponha a provar matematicamente e formalmente quais comportamentos podem ser representados por quais níveis da hierarquia de Chomsky seria de grande valia.

- Seria interessante também pensar em encontrar uma linguagem da hierarquia de Chomsky que conseguisse representar os comportamentos robóticos em um labirinto genérico. Isto foi tentado neste trabalho, começando inicialmente com linguagens livres de contexto, entretanto a dependência tanto dos marcos como da direção que o AA percorria nos fazem inferir que possa surgir a necessidade de se modelar isto como uma linguagem sensível ao contexto, onde necessitaria haver dois símbolos não terminais (direção e marco) do lado esquerdo de uma transição da gramática geradora.

- Embora não tanto dentro do escopo deste trabalho, seria interessante uma reflexão sobre a classificação dos comportamentos sob o prisma da etologia, conforme foi visto no capítulo 2. Esta classificação permanece aceita por mais de 30 anos, entretanto, experiências como as descritas em [95], [46] e [96], induzem a uma reflexão sobre conceitos de racionalidade animal.

- Trabalhos que visem o desenvolvimento de modelos de RNAs mais eficientes que consigam implementar o funcionamento de autômatos linearmente limitados também se tornam necessários.

- Finalmente, implementar e aplicar as redes neurais artificiais que são capazes de funcionar como os autômatos da hierarquia de Chomsky em aplicações práticas na robótica também é um objetivo a ser alcançado futuramente.

Referências Bibliográficas

[1] Mars exploration: Home. http://mars.jpl.nasa.gov. Acessado em 15/07/2003, 2003.

[2] Simulação de máquina de turing através de redes neurais. http://www.cin.ufpe.br/ if114/Monografias/Redes%20Neurais/Sem%20Pesos. Acessado em 05/02/2004, 2004.

[3] AGRE, P. E. & CHAPMAN, D. What are plans for? *Robotics and Autonomous Systems 6* (1990), 17–34.

[4] ALCOCK, J. *Animal Behavior: An Evolutinary Approach*. Sinauer Associates, Massachusetts, 1998.

[5] ALON, N.; DEWDNEY, A. & OTT, T. Efficient simulation of finite automata by neural nets. *Journal of the Association for Computing Machinery* (1991), 495–514.

[6] ARKIN, R. C. Path planning for a vision-based autonomous robot. In *Proceedings of the SPIE Conference on Mobile Robots*, 1986.

[7] ARKIN, R. C. Towards cosmopolitan robots: Intelligent navigation in extended man-made environments. Ph.d. dissertation, University of Massachusetts. Department of Computer and Information Science, 1987.

[8] ARKIN, R. C. *Behavior-Based Robotics*. The MIT Press, Cambridge, Massachussetts, 1998.

[9] AZEVEDO, F. M. *Contribution to the Study of Neural Networks In Dynamical Expert Systems*. Ph.d. thesis, Institut dÍnformatique, FUNDP, Belgium, 1993.

[10] BADDELEY, A. D. *Working Memory*. Oxford University Press, 1998.

[11] BALDASSARRE, G.; NOLFI, S. & PARISI, D. Evolving mobile robots able to display collective behaviours. In *Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour*, Monte Verità, Ascona, Switzerland, September 2002, C. Hemelrijk & E. Bonabeau, Eds., University of Zurich, p. 11–22.

[12] BARRETO, J. M. *Inteligência Artificial, no limiar do século XXI: Abordagem Híbrida - Simbólica, conexionista e evolutiva*, 2 ed. ppp Edições, Florianópolis, SC, Brasil, 2000.

[13] BEKKERS, J. M. & STEVENS, C. F. Presynaptic mechanism for long-term potentiation in the hippocampus. *Nature*, 346 (1990), 724–729.

[14] BLISS, T. V. P. & LOMO, T. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology*, 232 (1973).

[15] BODÉN, M. & WILES, J. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science 12* (2000), 197–210.

[16] BODEN, M. & WILES, J. On learning context-free and context-sensitive languages. *IEEE Transactions on Neural Networks 13*, 2 (March 2001), 491–493.

[17] BOERS, E. J. W. & KUIPER, H. Biological metaphors and the design of modular artificial neural networks, 2001.

[18] BONFIM, V. D. Apostila de aula da disciplina de teoria de autômatos e da computabilidade, março 2003.

[19] BROADBENT, D. E. *Perception and Communication*. Pergamon, 1958.

[20] BROOKS, R.; BREAZEAL, C.; MARJANOVIC, M. & SCASSELLATI, B. The cog project: Building a humanoid robot. *Lecture Notes in Computer Science 1562* (1999), 52–87.

[21] BROOKS, R. A. Elephants don't play chess. *Robotics and Autonomous Systems 6* (jun 1990), 3–15.

[22] BROWN, J. Some tests of the decay theory of immediate memory. *Q.J. Exp. Psychol*, 10 (1958), 12–21.

[23] CHEN, C.-H. & HONAVAR, V. Neural network automata. In *Proceedings of World Congress on Neural Networks*, 1994.

[24] CHEN, C. H. & HONAVAR, V. A neural network architecture for syntax analysis. *IEEE Transactions on Neural Networks 10*, 1 (1999), 94–114.

[25] CHOMSKY, N. On certain formal properties of grammars. *Information and Control 2* (1959), 137–167.

[26] CLOUSE, D. S.; GILES, C. L.; HORNE, B. G. & COTTRELL, G. W. Time-delay neural networks: Representation and induction of finite state machines. *IEEE Transactions on Neural Networks* (1997).

[27] COHEN, N. J. & CORKIN, S. The amnesic patient H. M.: Learning and retention of a cognitive skill. *Neuroscience Abstracts*, 7 (1981), 235.

[28] CORKIN, S. Lasting consequences of bilateral medial temporal lobectomy: Clinical course and experimental findings in h.m. In *Seminars in Neurology*, 1984, no. 4, p. 249–259.

[29] DA MOTA ALVES, J. B. Ficção, realidade e expectativa de robôs inteligentes baseados em comportamento. In *Anais do 1º Simpósio Brasileiro de Automação Inteligente*, Rio Claro, SP, Brasil, 1993, p. 145–154.

[30] DANEMAN, M. & CARPENTER, P. A. Individual diferences in working memory and reading. *J. Verb. Learn. Verb. Be*, 19 (1980), 450–466.

[31] DAS, S.; GILES, C. L. & SUN, G. Z. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, 1992.

[32] DAWKINS, M. S. *Explicando o Comportamento Animal*. Manole, São Paulo, Brasil, 1989.

[33] DE ALMEIDA E SILVA, F. Redes neurais hierárquicas para implementação de comportamentos robóticos em agentes autônomos. Mestrado em ciência da computação, Universidade Federal de Santa Catarina, 2001.

[34] DE BIRAN, M. *The Influence of Habit on the Faculty of Thinking*. Williams and Wilkings, 1804.

[35] DETHIER, V. G. & STELLAR, E. *Comportamento Animal*. ed. da Universidade de São Paulo, 1973.

[36] FIRBY, R. J. *Adaptive Execution in Complex Dynamic Worlds*. Ph.d. dissertation, Faculty of the Graduate School of Yale University, 1989.

[37] FOGEL, D. B. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press, New York, 1995.

[38] FRASCONI, P.; GORI, M.; MAGGINI, M. & SODA, F. Unified integration of explicit rules and learning by example in recurrent networks. In *IEEE Transactions on Knowledge and Data Engineering, 1993*, 1993.

[39] FRASCONI, P.; GORI, M. & SODA, G. Injecting nondeterministic finite state automata into recurrent neural networks. Relatório Técnico. DSI-RT15/92, Dipartimento di Sistemi e Informatica, Firenze, Italy, 1992.

[40] GALL, F. J. *The Influence of the Brain on the Form of the Head*. Marsh, Capen e Lion, 1835.

[41] GARDNER, H. E. *The Mind's New Science: A History of the Cognitive Revolution*. Basic Books, New York, 1985.

[42] GERS, F. A. & SCHIMIDHUBER, J. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks 12*, 6 (November 2001).

[43] GOUDREAL, M.; GILES, C. L.; CHAKRADHAR, S. T. & CHEN, D. First-order vs. second-order single layer recurrent neural networks. *IEEE Transactions on Neural Networks* (1994), 524–532.

[44] GOWDY, J. Sausages: Between planning and action. Relatório técnico., Robotics Institute. Carnegie Mellon University, 1994.

[45] HEBB, D. *The organization of behavior. A neuropsychological theory*. Wiley, New York, 1949.

[46] HILGARD, E. R. & ATKINSON, R. C. *Introdução à Psicologia*. Companhia Editora Nacional, 1979.

[47] HORNE, B. & GILES, C. An experimental comparison of recurrent neural networks. In *Advances in Neural Information Processing Systems*, 1995, D. T. G. Tesauro & T. Leen, Eds., MIT Press.

[48] JAMES, W. *The principles of psychology*. Henry Holt, 1980.

[49] JOHNSON, G. *Nos palácios da memória*. Siciliano, São Paulo, 1994.

[50] KAELBLING, L. P. *Rex programmer's manual*. SRI Intl, 1986. Technical Note 381.

[51] KAELBLING, L. P. Goals as parallel program specifications. *In Proc. AAAi Conf. 1* (1988), 60–65.

[52] KAELBLING, L. P. & ROSENSCHEIN, S. Action and planning in embedded agents. *Robotics and Autonomous Systems 6* (1990), 35–48.

[53] KEMPER, S. Adults' sentence fragments - who, what, when, where, and why. *Communication Research*, 19 (1992), 444–458.

[54] KOHLER, W. *Gestalt Psychology: An Introduction to New Concepts in Modern Psychology*. Liveright Publishing Co, 1947.

[55] KOLB, B. & WISHAW, I. Q. *Neurociência do Comportamento*. Manole, 2002.

[56] KONDO, Y. & SAWADA, Y. Functional abilities of a stochastic logic neural network. *IEEE Transactions on Neural Networks* (1992).

[57] LACERDA, L. O. Mapas auto-organizáveis de kohonen aplicados ao mapeamento de ambientes de robótica móvel. Mestrado em ciência da computação, Universidade Federal de Santa Catarina, 2001.

[58] LAVINE, R. A. *Neurophysiology: The Fundamentals*. The Collamore Press, 1983.

[59] LEE, P.; CASSIDY, W.; APOSTOLOPOULOS, D.; BASSI, D.; BRAVO, L.; CIFUENTES, H.; DEANS, M.; FOESSEL, A.; MOOREHEAD, S.; PARRIS, M.; PUEBLA, C.; PEDERSEN, L.; SIBENAC, M.; VALDÉS, F.; VANDAPEL, N. & WHITTAKER, W. Search for meteorites at martin hills and pirrit hills, Antarctica. In *30th Lunar and Planetary Science Conference*, Houston, Texas, 1990.

[60] LEWIS, H. R. & PAPADIMITRIOU, C. H. *Elements of the Theory of Computation*. Prentice Hall, 1981.

[61] LIN, L. T.; HORNE, B. G.; TINO, P. & GILES, C. L. Learning long-term dependencies is not as difficult with NARX. Relatório Técnico. CS-TR-3500, University of Maryland, 1995.

[62] LYONS, D. M. & ARBIB, M. A. A formal model of computation for sensory-based robotics. *IEEE Journal of Robotics and Automation* (June 1989), 280–293.

[63] MACKENZIE, D. *A Design Methodology for the Configuration of Behavior-based Mobile Robots*. Ph.d. dissertation, College of Computing, Georgia Tech, Atlanta, GA, 1996.

[64] MAES, P. The dynamics of action selection. In *AAAI Spring Symposium on AI Limited Rationality. IJCAI*, Detroit, MI, 1997, p. 991–997.

[65] MARCHI, J. Navegação de robôs móveis autônomos: Estudo e implementação de abordagens. Mestrado em engenharia elétrica, Universidade Federal de Santa Catarina, 2001.

[66] MASKARA, A. & NOETZEL, A. Forced simple recurrent neural network and grammatical inference. In *Proceedings of the 14th Conference of the Cognitive Science Society*, Bloomington, IN, 1992, Lawrence Erlbaum, p. 420–425.

[67] MATARIC, M. Navigating with a rat brain: a neurobiologically-inspired model for robot spatial representation. In *In From Animals to Animats*, S, Cambridge, MA., 1991, J.-A. Meyer & Wilson, Eds., MIT Press.

[68] MATARIC, M. J. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science 2*, 3 (March 1998), 82–87.

[69] MESSENGER, J. B. *Nervos Cérebro e Comportamento*, vol. 22. Ed. da Universidade de São Paulo, São Paulo, Brasil, 1980.

[70] MILLER, D. J. & CHARLEENE, R. An object-oriented environment for robot system architectures. In *In Proc. IEEE Intl. Conf. on Robotics and Automation*, Cincinnati, OH, 1990, vol. 1, p. 352–361.

[71] MILNER, B. Disorder of memory after brain lesions in man. *Neuropsychologia*, 6 (1968), 175–179.

[72] MONDADA, F.; GUIGNARD, A.; BONANI, M.; BÄR, D.; LAURIA, M. & FLOREANO, D. Swarm-bot: From concept to implementation. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS 2003)*, Las Vegas, Nevada, US, October 27 - 31, 2003 2003, IEEE Press, p. 1626–1631.

[73] MURPHY, R. R. *Introduction to AI Robotics*. MIT Press, November 2000.

[74] NEISSER, U. *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W.H. Freeman, 1976.

[75] NICHOLLS, J. G.; MARTIN, A. R.; WALLACE, B. G. & FUCHS, P. A. *From Neuron to Brain*. Sinauer Associates, Inc., 2000.

[76] NICOLESCU, M. N. & MATARIC, M. J. Learning and interacting in human-robot domains. *In White, C. and Dautenhahn, K., (Eds.), Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 31* (2001).

[77] OMLIN, C. W. & GILES, C. L. Constructing deterministic finite-state automata in sparse recurrent neural networks. In *IEEE International Conference on Neural Networks (ICNN'94)*, Piscataway, NJ, 1994, IEEE Press, p. 1732–1737.

[78] PANI, J. Personal communication, may 1996.

[79] PAPALIA, B.; PRENDIN, W. & VERUGGIO, G. SARA, An Autonomous Underwater Vehicle for Researches in Antarctica. *Oceans Conference* (1994).

[80] PETERSON, L. R. & J.PETERSON, M. Short-term retention of individual verbal items. *J. Exp. Psychol*, 58 (1959), 193–198.

[81] PETTINARO, G.; KWEE, I.; GAMBARDELLA, L.; MONDADA, F.; FLOREANO, D.; NOLFI, S.; DENEUBOURG, J. L. & DORIGO, M. Swarm robotics: A different approach to service robotics. In *Proceedings of the 33rd International Symposium on Robotics*, Stockholm, Sweden, October 2002, International Federation of Robotics.

[82] PIAGET, J. & INHELDER, B. *The Child's Conception of Space*. Norton, New York, 1967.

[83] POLLACK, J. B. *On Connectionist Models of Natural Language Processing*. Ph.d thesis, Computer Science Dept, University. of Illinois, 1987.

[84] POSTMAN, L. & PHILLIPS, L. W. Short-term temporal changes in free recall. *Q. J. Exp. Psychol*, 17 (1965), 132–138.

[85] QUESNEL, A. *A grécia. Mitos e lendas*, 6 ed. Editora Ática, São Paulo, 1985.

[86] RAY, T. S. An approach to the synthesis of life. In *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity*, C. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen, Eds., vol. 11.

[87] RAY, T. S. Synthetic life: Evolution and optimization of digital organisms. In *Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers*, Athens, GA, December 1991, H. B. I. e. Billingsley K. R., E. Derohanes, Ed., The Baldwin Press, p. 489–531.

[88] RAY, T. S. An evolutionary approach to synthetic biology: Zen and the art of creating life. In *Artificial Life 1(1/2):*, 1994, p. 195–226.

[89] ROISENBERG, M. Emergência de inteligência em agentes autônomos através de modelos inspirados na natureza. Doutorado em engenharia elétrica, Universidade Federal de Santa Catarina, 1999.

[90] ROISENBERG, M. PiramidNet - Redes Neurais Modulares e Hierárquicas: Fundamentos e Aplicações em Robótica. Plano de Trabalho para Projeto de Pesquisa. Relatório técnico., Universidade Federal de Santa Catarina, 2000.

[91] ROISENBERG, M.; BARRETO, J. M.; DE ALMEIDA E SILVA, F.; VIEIRA, R. C. & COELHO, D. K. Pyramidnet: A modular and hierachical neural network architecture for behaviour based robotics. In *ISRA - International Symposium on Robotics and Automation*, Queretaro, Mexico, 2004.

[92] ROISENBERG, M.; BARRETO, J. M. & DE AZEVEDO, F. M. Neural network complexity classification based on the problem. In *IJCNN - IEEE International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998.

[93] ROWLAND, L. P. & SHNEIDER, N. A. Medical progress: Amyotrophic lateral sclerosis. *New England Journal of Medicine* (2001), 1688–1700.

[94] RUSSELL, S. & NORVIG, P. *Artificial Intelligence: a modern approach*. Prentice Hall, 1995.

[95] SERPELL, J. A. *In the Company of Animals: A study of Human-Animal Relationships*. Cambridge University Press, 1996.

[96] SHELDRAKE, R. *Dogs That Know When Their Owners Are Coming Here and other unexplained powers of animals*. Crown Publishing Group, 1999.

[97] SIEGELMANN, H. T.; HORNE, B. G. & GILES, C. L. Computational capabilities of recurrent narx neural networks. Relatório Técnico. CS-TR-3408, University of Maryland, 1995.

[98] SIEGELMANN, H. T. & SONTAG, E. D. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, Pittsburgh, Pennsylvania, United States, 1992, ACM Press, p. 440–449.

[99] SINGH, S.; SIMMONS, R.; SMITH, T.; STENTZ, A.; VERMA, V.; YAHJA, A. & SCHWEHR, K. Recent progress in local and global traversability for planetary rovers. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[100] SOBEL, H. I. *Os Porquês do Judaísmo?* Congregação Israelita Paulista.

[101] STEIJVERS, M. & GRÜNWALD, P. A recurrent network that performs a context-sensitive prediction task. Relatório técnico., NeuroCOLT Technical Report NC-TR-96-035, CWI, Dept. of Algorithmics. Netherlands, 1996.

[102] SUDKAMP, T. A. *Languages and Machines - An Introduction to the Theory of Computer Science*. Addison-Wesley Publishing Company, Inc., New York, 1988.

[103] SUN, G. Z.; GILES, C. L. & CHEN, H. H. The neural network pushdown automaton: Architecture, dynamics and training. *Lecture Notes in Computer Science 1387* (1998), 296–345.

[104] TURING, A. M. Computing machinery and intelligence. *Mind 59* (1950), 433–460.

[105] VIEIRA, R. C.; DE ALMEIDA E SILVA, F. & ROISENBERG, M. Redes neurais hierárquicas para controle robótico. In *IV Congresso Brasileiro de Computação - CBComp*, Itajai-SC, 2004.

[106] VIEIRA, R. C. & ROISENBERG, M. Redes neurais artificiais: Um breve tutorial. Relatório Técnico. UFSC/INE/L3C-01/2003, Laboratório de Conexionismo e

Ciências Cognitivas - L3C, Universidade Federal de Santa Catarina - Florianópolis-SC, 2003.

[107] VIEIRA, R. C.; ROISENBERG, M. & FURTADO, O. J. V. Formal languages aspects as a tool for representation and implementation of behavior-based robotics. In *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Singapore, 2004.

[108] VIEIRA, R. C.; TENÓRIO, M. B.; ROISENBERG, M. & BORGES, P. S. S. Comparação entre redes neurais artificiais e rough sets para classificação de dados. In *VI Brazilian Conference on Neural Networks*, São Paulo-SP, 2003.

[109] WASHINGTON, R.; GOLDEN, K.; BRESINA, J.; SMITH, D.; ANDERSON, C. & SMITH, T. Autonomous rovers for mars exploration. In *Proceedings of the 1999 IEEE Aerospace Conference*, 1999.

[110] WATSON, J. B. *Behaviorism*. People's Institute Publishing Co, 1925.

[111] WETTERGREEN, D.; THORPE, C. & WHITTAKER, W. R. L. Exploring mount erebus by walking robot. In *In International Conference of Intelligent Autonomous Systems*, 1993, p. 72–81.

[112] WILES, J. & ELMAN, J. Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, 1995, M. Press, Ed.

[113] WIND RIVER SYSTEMS, I. *VxWorks Reference Manual Ver. 4.0*. Emeryville, CA.

[114] WOLPERT, D. A computationally universal field computer wich is purely linear. Relatório Técnico. LA-UR-91-2937, Los Alamos National Laboratory, 1991.