UNIVERSIDADE PAULISTA

Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para a área de Automação em Engenharia de Produção

Fernando Marco Perez Campos

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Paulista, como exigência para o exame de qualificação.

SÃO PAULO
2007

UNIVERSIDADE PAULISTA

Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para Automação em Engenharia de Produção

Fernando Marco Perez Campos

Orientador: Prof. Dr. Jair Minoro Abe.
Co-orienteador: Prof. Dr. José Pacheco de Almeida Prado

Área de Concentração: Sistemas de Informação.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Paulista, como exigência para o exame de qualificação.

SÃO PAULO
2007

FERNANDO MARCO PEREZ CAMPOS

Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para Automação em Engenharia de Produção

Esta dissertação foi julgada e aprovada para a obtenção do grau de **Mestre em Engenharia de Produção** no **Programa de Pós-Graduação em Engenharia de Produção** da Universidade Paulista - UNIP.

São Paulo, 28 de fevereiro de 2007.

_____
Prof. Dr. Oduvaldo Vendrameto
Coordenador do Curso

_____
Prof. Dr.Jair Minoro Abe
Orientador

_____
Prof. Dr.José Pacheco de Almeida Prado
Examinador Interno

_____
Prof. Dr. Lafayette de Moraes
Examinador Externo

_____
Prof. Dr.Israel Brunstein
Suplente Interno

_____
Prof. Dr. João Inácio da Silva Filho
Suplente Externo

São Paulo
2007

CAMPOS, Fernando Marco Perez
Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para automação em Engenharia de Produção / Fernando Marco Perez Campos.
– São Paulo, 2007.
108 f.

Dissertação (Mestrado) – Apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Paulista, São Paulo, 2007.

Área de Concentração: Gestão de Informação
Orientação: Prof. Dr. Jair Minoro Abe

1. Lógica Paraconsistente Anotada 2.Robô móvel 3. Hardware Reconfigurável 4. Comportamento robótico

Dedico este trabalho à minha esposa Juliane, à nossa filha Luiza, que no dia do exame de defesa estará completando um ano e meio, e à nossa filha Mariana, que está chegando ainda este mês e que já é muito amada por nós. À todas elas o meu amor!

Dedico-o também aos meus pais Alfredo e Zildete, que sempre me apoiaram de forma incondicional, nas horas em que precisei.

Dedico-o ainda ao meu sobrinho e afilhado Gabriel Marco, que está trilhando este mesmo caminho meu, pelo qual, tenho certeza, terá muito sucesso.

Agradecimentos:

A Deus por me dar saúde, uma família maravilhosa e muita força para lutar sempre.

Á minha esposa Juliane, o meu amor e agradecimentos pelo apoio, carinho e dedicação sem os quais não teria conseguido chegar até aqui. Muito obrigado por estar ao meu lado, e me fazer feliz.

Ás minhas filhas Luiza e Mariana, que estão me ensinando muitas coisas sobre a vida e às quais devo minha vontade de viver e de lutar para melhorar sempre. Com vocês eu aprendi o que é o verdadeiro amor incondicional.

Aos meus pais, Alfredo e Zildete, pela doação, por me proporcionarem uma boa educação e pelo respeito às minhas escolhas. Vocês são um verdadeiro exemplo de amor e vida; obrigado também por cuidarem da Luiza com tanto carinho.

Aos meus irmãos Hamilton e Iara, à minha cunhada Conceição, ao meu cunhado Jorge e aos meus sobrinhos Gabriel, Thyago, e Priscila pelo companheirismo, apoio, e principalmente pelo acolhimento durante os dias em que precisei passar em São Paulo para o desenvolvimento deste trabalho.

Ao meu amigo professor Dr. José Pacheco de Almeida Prado, pelos conselhos e oportunidades oferecidos ao longo dos últimos oito anos. O meu sucesso profissional devo a você. Muito obrigado por confiar em mim.

Especialmente, meus sinceros agradecimentos ao professor Dr. Jair Minoro Abe, pela paciência com minhas falhas, por respeitar minhas dificuldades, pelas orientações e constante disposição de ajudar a qualquer hora e sempre com muita disposição. Para mim foi uma grande honra poder caminhar a seu lado.

Ao professor Dr. Lafayette de Moraes (PUC-SP) pelo aceite do convite para participar da banca de defesa desta dissertação, pelas sugestões dadas no exame de qualificação e pela compreensão dos atrasos ocorridos, por motivos de força maior, na entrega deste material.

Ao amigo professor Sylvio Albernaz, pelo companheirismo nas viagens a São Paulo, pelo revezamento para buscar café durante as aulas de segunda-feira de manhã e por proporcionar bom humor mesmo durante as dificuldades encontradas no caminho até aqui.

Aos amigos professores Saulo Rosa e Silva, Rodrigo Plotze, Marcelo Duarte, Marcelo Alves e professor Belini; e ao meu primo e amigo João Roberto do Carmo, pelas orientações e apoio nas horas em que precisei.

Ao meu sogro Jader, à minha sogra Maria das Graças, e à minha cunhada Júlia por ajudar a cuidar da Luiza durante o tempo em que, além do trabalho, precisei me dedicar ao mestrado.

Ao meu cunhado e colega de trabalho Jailson Alexandre e ao meu amigo e também colega de trabalho Benê pela paciência durante minha ausência.

Ao meu amigo e irmão Rodrigo Melges pelos conselhos nas horas difíceis e companheirismo durante os momentos bons e ruins da minha vida. Você é um amigo de verdade!

Ao colega Edvan Roberto, funcionário da Unip que apesar do pouco contato, bem sei que é uma pessoa verdadeiramente humana. Obrigado pela ajuda nas horas de desespero.

E a todos aqueles que direta ou indiretamente me ajudaram a fazer com que este dia fosse possível. Muito obrigado.

O Ensino

Então um Professor disse:

- Fala-nos do Ensino.

E ele respondeu:

- Nenhum homem vos pode revelar nada que não repouse já meio adormecido na manhã do vosso conhecimento. O mestre que caminha à sombra do templo, entre os discípulos, não reparte a sua sabedoria mas antes da sua fé e do seu amor.

Se for verdadeiramente sábio, não vos convidará a entrar na casa da sabedoria, mas levar-vos-á aos umbrais do vosso próprio espírito.

O astrônomo pode falar-vos da sua compreensão do espaço, mas não pode dar-vos a sua compreensão.

O músico pode cantar para vós a melodia que enche todo o espaço, mas não pode dar-vos o ouvido que aprende o ritmo nem a voz que lhe devolve o eco.

E o que é versado na ciência dos números, pode falar nas relações dos pesos e medidas, mas não pode levar-vos até lá.

Porque a visão de um homem não pode emprestar as suas asas a outro homem.

E assim como cada um de vós se agüenta sozinho no conhecimento de Deus, assim deve estar sozinho no seu conhecimento de Deus e na compreensão da terra.

(*O PROFETA*, GIBRAN KHALIL)

Sumário

Resumo IX
Abstract X
Lista de Abreviaturas, Siglas e Símbolos XI
Lista de Quadros, Tabelas e Figuras XIII

1. Introdução
1.1 Considerações iniciais 15
1.2 Objetivos 19
1.3 Justificativa 19
2. Robótica comportamental
2.1 Definição de robô 21
2.2 Computadores e robôs 22
2.3 Robôs inteligentes 23
2.3.1 Precursores da Cibernética 23
2.3.1.1 Tartarugas de Walter 24
2.3.1.2 Veículos de Braitenberg 26
2.3.2 Precursores da Inteligência Artificial 27
2.3.3 Precursores da robótica 29
2.4 Espectro do controle robótico 31
2.4.1 Sistemas deliberativos 31
2.4.2 Sistemas reativos 32
2.5 Comportamentos 35
2.5.1 Comportamento emergente 35
2.5.2 Comportamento primitivo 36
2.5.3 Comportamento servo e balístico 36
2.5.4 Arbitragem 43
2.6 Métodos de projeto de comportamentos 49
3. Lógica Paraconsistente Anotada
3.1 Lógicas clássicas e não-clássicas 54
3.2 Breve história da lógica 55
3.3 O surgimento da lógica paraconsistente 57
3.4 Lógica paraconsistente 58
3.5 Lógica paraconsistente anotada 59
3.5.1 Reticulado “quatro” 60
3.5.2 Reticulado “seis” 61

Sumário

3.6 Lógica paraconsistente anotada evidencial Eτ 63
3.7 O reticulado no plano cartesiano 65
4. Hardware Reconfigurável
4.1 Hardware Reconfigurável 73
4.2 Uso atual 76
4.3 Hardware 78
4.4 Reconfigurabilidade de FPGAs 82
4.5 Programação de FPGAs 85
5. Implementação
5.1 Considerações iniciais 89
5.2 Metodologia 92
5.3 Construção do robô 97
5.3.1 Fonte de alimentação 97
5.3.2 ATMEGA8 97
5.3.3 Sonares 99
5.3.4 LDRs 99
5.3.5 Hardware Reconfigurável 100
5.3.5.1 Kit didático Altera 100
5.3.5.2 Circuito de análise paraconsistente e definição do comportamento 100
5.3.5.3 Circuito de ativação dos comportamentos 105
5.3.6 Placa *driver* dos motores 107
5.3.7 Testes da arquitetura 107
6 Conclusão
6.1 Considerações finais 109
6.2 Contribuições deste trabalho 110
6.3 Trabalhos futuros 110
Referências Bibliográficas 112

Anexo A Características do microcontrolador ATMEGA8 116
Anexo B Diagrama completo do circuito de análise paraconsistente e comportamento 117
Anexo C Diagrama completo de ativação dos Comportamentos 119

Resumo

CAMPOS, F. M. P. **Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para a área de Automação em Engenharia de Produção.** Dissertação (Mestrado em Engenharia de Produção) – Instituto de Ciências Exatas e Tecnológicas, Universidade Paulista, 2007.

Palavras-chave: Lógica Paraconsistente Anotada, robô móvel, Hardware Reconfigurável, Comportamento robótico.

Este trabalho apresenta a utilização da Lógica Paraconsistente Anotada, juntamente com as técnicas de comportamentos robóticos (*Behavior-Based Robotics*), aplicada sobre uma arquitetura de *Hardware* Reconfigurável.

O tratamento das informações inconsistentes ou paracompletas é implementado dentro do Hardware Reconfigurável de uma forma simples, porém funcional, juntamente com o módulo que define o seu comportamento para que o seu destino seja alcançado com sucesso.

Abstract

CAMPOS, F. M. P. **Arquitetura de Hardware Reconfigurável Paraconsistente em Navegação de Robôs Móveis: uma contribuição para a área de Automação em Engenharia de Produção.** Dissertation (Master of Science in Production Engineering) – Instituto de Ciências Exatas e Tecnológicas, Universidade Paulista, 2007.

Keywords: Paraconsistente Annotated Logic, Mobile Robot, Reconfigurable Hardware, Behavior-Based Robotics.

This work presents the use of the, Paraconsistente Annotated Logic with the techniques of robotic behaviors (Behavior-Based on Robotics), applied on an architecture of the Reconfigurable Hardware. The treatment of the uncertain information or paracompliteness is implemented inside of the Reconfigurable Hardware of a simple, however functional form, with the module that defines its behavior so that its destination is reached successfully.

Abreviaturas, Siglas e Símbolos.

ASIC (Application Specific Integrated Circuit) ≡ Circuito integrado dedicado construído para executar uma tarefa específica, ou seja, um processador dedicado.

CLPs (Controladores lógicos programáveis) ≡ É um computador especializado, baseado num microprocessador que desempenha funções de controle de diversos tipos e níveis de complexidade

CPLDs (*Complex Programmable Logic Devices*) ≡ Dispositivo lógico programável complexo, introduzido no mercado internacional pela empresa Altera Corp. em 1984. Permite a implementação de grande variedade de circuitos lógicos.

EDA (*Eletronic Design Automation*) ≡ Metodologia de projeto de circuitos digitais, apoiada em poderosas ferramentas de software, que envolve várias etapas de projeto, geralmente automatizadas. O ambiente de programação consiste em vários programas de síntese, mapeamento, posicionamento, roteamento, simulação e testes.

EEPROM (*Electrically Erasable Programmable Read Only Memory*) ≡ Memória de leitura no qual pode-se apagar os dados por sinais elétricos, sem a utilização de luz ultravioleta.

FPGAs (*Field Programmable Gate Array*) ≡ Dispositivo semiconductor largamente utilizado para o processamento de informações digitais. Foi criado pela Xilinx Inc. e teve seu lançamento no ano de 1985 como um dispositivo que poderia ser programado de acordo com as aplicações do usuário (programador).

HDL (*Hardware Description Language*)≡ Linguagem de descrição de *hardware* desenvolvida para auxiliar os projetistas a documentar projetos e simular grandes sistemas, principalmente em projetos de dispositivos ASICs.

HR ≡ *hardware* Reconfigurável

LUT (*Look-Up Table*) ≡ Um tipo de bloco lógico que contém células de armazenamento, utilizadas para implementar pequenas funções lógicas.

MOS (*Metal Oxide Semiconductor*) ≡ Semicondutor de óxido de metal.

MPGA (*Mask Programmable Gate Arrays*) ≡ Dispositivo lógico programável, que utiliza no seu processo de fabricação máscaras genéricas de módulos pré-projetados.

PLD (*Programmable Logic Devices)* ≡ Dispositivos lógicos programáveis simples (circuitos integrados) que podem ser configurados pelo próprio usuário.

SPLD (*Simple Programmable Logic Devices*) ≡ Dispositivos lógicos programáveis simples de baixa capacidade.

SRAM (*Static Random Access Memory*) ≡ Memória de leitura e escrita estática que utiliza semicondutores que funcionam como interruptores elétricos chamados biestáveis ou *flip-flops*.

T ≡ inconsistente

V ≡ verdadeiro

F ≡ falso

$\perp$ ≡ indeterminado

Qv ≡ quase verdadeiro

Qf ≡ quase falso

μ ≡ constante de anotação

$\in$ ≡ pertence

τ ≡ reticulado

$\vee$ ≡ disjunção

$\wedge$ ≡ conjunção

$\rightarrow$ ≡ implicação

$\neg$ ≡ negação

Lista de Quadros, Tabelas e Figuras

Quadro 5-1 – Camadas do robô Luiza 92

Tabela 3-1 – Tabela verdade com operador NOT 67
Tabela 5-1 – sinais dos sensores de luz. 101
Tabela 5-2 – sinais dos sonares. 101
Tabela 5-3 - comportamentos do robô móvel Luiza. 102
Tabela 5-4 – Tabela-verdade da análise paraconsistente do robô móvel Luiza. . 103
Tabela 5-5 – Tabela-verdade do codificador dos comportamentos. 103

Figura 1-1 – Disposição dos sonares do robô Amanda 17
Figura 1-2 – Robô Amanda desviando de um obstáculo 18
Figura 2-1 - Espectro do sistema de controle de um robô 31
Figura 2-2 - MEF do Escape 41
Figura 2-3 - MEF simplificada do *escape*, usando tempo decorrido na implementação 42
Figura 2-4 - O objetivo do Seguir-linha é seguir uma faixa no chão, mantendo-a entre dois sensores 43
Figura 2-5 - Arbitragem por prioridade fixa. Comportamentos localizados mais acima no esquema terão maior prioridade 44
Figura 2-6. Comportamento avaliador do sensor 48
Figura 2-7. Uso de um sensor virtual 48
Figura 2-8. MEF de avaliação contra falsos positivos 49
Figura 2-9. Projeto etologicamente guiado/restringido 51
Figura 2-10. Projeto baseado na atividade localizada 52
Figura 2-11. Metodologia guiada por experimentos 53
Figura 3-1. Reticulado “quatro” 60
Figura 3-2. Reticulado “seis” 62
Figura 3-3. Reticulado associado à $E\tau$ 64
Figura 3-4. Reticulado da $E\tau$ no QUPC 66
Figura 3-5. Valores ternários e independentes dos graus de evidência favorável e evidência contrária geram cinco estados lógicos não extremos 69
Figura 3-6. Reticulado com cinco graus de evidências 70
Figura 3-7. Reticulado com nove graus de evidências 70
Figura 3-8. Reticulado otimizado 71
Figura 4-1 - Reconfigurabilidade dinâmica e remota de fpgas 75

Figura 4-2 – Estrutura simplificada de um FPGA 78
Figura 4-3 – Antifuse 81
Figura 4-4 – Tecnologia de programação Gate flutuante 81
Figura 4-5– Tecnologia de programação SRAM 82
Figura 4-6 – Exemplos de reconfigurabilidade de FPGAs 83
Figura 4-7 – Etapas do EDA 85
Figura 4-8 – Editor gráfico do Quartus II 86
Figura 5-1 – Diagrama das camadas do robô Luiza 94
Figura 5-2 - Fluxograma do programa do robô móvel Luiza 96
Figura 5-3 - Fluxograma do programa do robô móvel Luiza (ajuste de direção) ... 97
Figura 5-4 - Fluxograma do programa implementado no ATMEGA8. 98
Figura 5-5 – Bloco lógico da análise paraconsistente implementado no ATMEGA8 105
Figura 5-6 - Bloco lógico do circuito de ativação dos comportamentos 106
Figura 5-7 - Diagrama do *chip* L293E 107

1 INTRODUÇÃO

1.1 Considerações Iniciais

Vários estudos têm sido feitos na tentativa de se achar uma solução eficiente e eficaz para a construção de uma máquina capaz de transformar informações em conhecimento manipulável mecanicamente.

Computadores com estas habilidades podem ser utilizados para as mais diversas aplicações, das quais a mais relevante para este trabalho é a capacidade de tomar uma decisão baseada em informações que nem sempre são completas e consistentes.

De acordo com (Costa e Abe, 2000), as imprecisões e inconsistências surgem naturalmente na descrição do mundo real. Isso ocorre em vários contextos; no entanto, os seres humanos são capazes de raciocinar adequadamente. A automatização de tais raciocínios requer o desenvolvimento de teorias formais apropriadas.

Para o desenvolvimento deste trabalho baseou-se fortemente na Lógica Paraconsistente Anotada (LPA). Para mais informações e detalhes da LPA, consultar o capítulo 3 deste trabalho (Lógica Paraconsistente Anotada Evidencial Eτ) ou, ainda, (Abe, 1992) e (Costa, Abe *et al.*, 1991).

A implementação da LPA em aplicações pode ser feita utilizando-se diversas tecnologias disponíveis como, por exemplo, *software* para ser executado em um computador, processador ou microcontrolador; diretamente no *hardware* de forma a ser desenvolvido com base em um sistema digital e eletrônica discreta; utilizando-se a recente tecnologia

denominada *Hardware R*econfigurável, em que um *chip* pode, por meio de uma programação, transformar-se fisicamente em um circuito integrado personalizado.

Em (Abe e Silva, 1998), (Da-Silva-Filho, Abe *et al.*, 1997), (Da-Silva-Filho, 1997) são descritos circuitos elétricos digitais (portas lógicas complemento, conjunção e disjunção) inspirados em uma classe de lógicas paraconsistentes anotadas; sendo a inconsistência tratada diretamente no *hardware*.

Um exemplo de implementação desta teoria refere-se ao robô móvel Emmy. O projeto robô móvel Emmy foi desenvolvido por (Da-Silva-Filho, 1999) e colaboradores, construído sobre uma base circular de alumínio com um diâmetro de 30 cm e 60 cm de altura.

A detecção de obstáculos é feita por dois sensores de ultra-som denominados para-sônicos, que são sincronizados por um microprocessador e fornecem ao sistema de controle informações já equacionadas sobre o grau de evidência favorável e evidência contrária; o sistema de controle é constituído por um *hardware* discreto Paracontrol (Controlador Lógico Paraconsistente); o acionamento mecânico utiliza dois motores de corrente contínua de 12Vcc acoplado a caixas de redução; a sua alimentação é fornecida por duas baterias de 12Vcc interligadas a fim de se obter uma fonte simétrica.

Um outro projeto com a mesma proposta de estudo é o robô Amanda, desenvolvido por (Silva, 2005).

O robô Amanda foi construído para validar um sistema de navegação para robôs móveis utilizando-se a Lógica Paraconsistente Anotada aplicada na teoria dos campos potenciais artificiais (Khatib, 1985), como método de reconhecimento e desvio de obstáculos.

Construído sobre uma base retangular de plástico, o robô Amanda, assim como o Emmy, pode ser dividido em três camadas fundamentais: percepção, controle e atuadores que funcionam com a alimentação de uma bateria recarregável de 12Vcc.

A camada de percepção é dotada de dois sonares denominados “Sonares Paraconsistentes”, que fornecem sinais não tratados, além de quatro sensores de luz do tipo LDR, ambos conectados ao módulo de controle do robô.

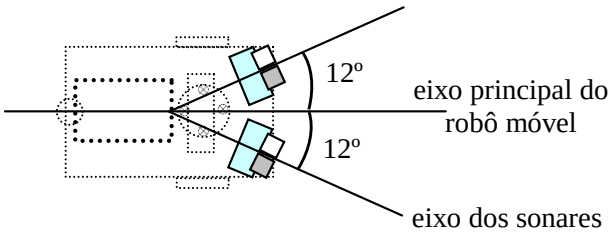


Figura 1-1 – Disposição dos sonares do robô Amanda (fonte: Silva 2005)

O módulo de controle possui como elemento principal um microcontrolador Risc Atmega8 da Atmel, que é o responsável por tratar os dados enviados pelos dois sonares utilizando a LPA, fazer a leitura dos sensores de luz e aplicar a técnica dos Campos Potenciais, para poder enviar sinais de controle aos atuadores de forma que ele consiga seguir uma fonte de luz e ao mesmo tempo desviar de obstáculos.

Os atuadores são dois motores de corrente contínua de 12Vcc conectados ao módulo de controle por intermédio de um driver de potência e circuitos de apoio, como por exemplo, inversor de corrente para que os movimentos sejam possíveis nos dois sentidos (frente / trás).

Toda a programação do robô Amanda, incluindo o tratamento dos sinais dos sonares, a recepção de sinais dos sensores de luz, a LPA, a tomada de decisão etc. é feita em *software* gravado no microcontrolador.

Figura 1-2 – robô Amanda desviando de um obstáculo (fonte: Silva 2005).

Os dois robôs apresentados tiveram como objetivo implementar a LPA em um projeto prático de Robótica, proporcionando ao autômato trafegar por ambientes desconhecidos, desviando de obstáculos de forma eficiente em ambientes com características diferentes e ao mesmo tempo dinâmicas.

Pela exposição feita, pode-se verificar que a implantação da LPA foi desenvolvida num primeiro momento por meio de eletrônica discreta, no caso do Emmy; e, posteriormente, utilizando-se um microcontrolador, no caso da Amanda.

1.2 - Objetivos

Considerando-se a viabilidade do *Hardware* Reconfigurável para o tratamento de informações paraconsistentes, bem como sua tecnologia emergente, nesta pesquisa adota-se a Lógica Paraconsistente Anotada em um *Hardware R*econfigurável.

A partir deste estudo, verifica-se se a LPA desenvolvida em um *Hardware* Reconfigurável consegue manter as vantagens da flexibilidade e, ao mesmo tempo, obter uma resposta satisfatória com relação ao desempenho.

1.3 Justificativa

Em vista da aplicabilidade do HR que pode manipular informações imprecisas, inconsistentes e/ou paracompletas, este trabalho se justifica por poder contribuir em aplicações industriais, especialmente nos processos de produção como Sistemas Flexíveis de Manufatura (Nickerson e Jasiobedzki, 2000), (Vaughan, Stoy *et al.*, 2000), (Ozbayrak e Al., 1997), (Wang, 1986) formados por um conjunto de máquinas versáteis capazes de processar simultaneamente médios e baixos volumes de uma variedade de tipos de partes, controlados por um Sistema Computacional e operadores. Segundo (Gómez, Rodrigues *et al.*, 1996), uma das expectativas de um Sistema de Manufatura Flexível é dotar a manufatura com algum grau de automação através do progresso obtido no desenvolvimento de seus componentes tecnológicos (máquinas, robôs, etc.).

Assim, esta pesquisa poderá servir de base para o desenvolvimento de futuros projetos funcionais como, por exemplo, seguir uma faixa desenhada em um “chão de fábrica” interligando células de manufatura.

Quando necessário, o robô deverá desviar de obstáculos que poderão surgir em posições aleatórias, bloqueando o seu caminho e conseqüentemente a execução da tarefa determinada, voltando em seguida para a faixa, desempenhando dessa forma a função de uma esteira, com a vantagem da flexibilidade de rota.

O trabalho também contribui nos avanços das aplicações dos sistemas lógicos paraconsistentes, ilustrando a lógica paraconsistente e seu significado prático.

2. Robótica comportamental

2.1. Definição de robô:

Segundo a *Robotics Industry Association* (RIA), um robô é um manipulador multifuncional e reprogramável, projetado para mover peças, ferramentas ou dispositivos especializados, com os movimentos programados para o desempenho de uma variedade de tarefas (Jablonsky e Posey, 1985).

Para Arkin, tal definição é um tanto restrita, excluindo robôs móveis, entre outras coisas, e propõe a seguinte definição:

Um robô inteligente é uma máquina capaz de extrair a informação de seu ambiente e de usar o conhecimento sobre seu mundo e mover-se com segurança de maneira significativa e personalizada. (Arkin, 1998)

Fisicamente, os robôs podem assumir tanto formas de seres vivos como de veículos não-tripulados. Podem ser diferenciados conforme o tamanho, os materiais, tipo de montagem, os atuadores (motores e transmissões), os sistemas sensoriais, de locomoção e de computação embutidos.

Os robôs devem ter um sistema de controle para prover mobilidade de maneira coordenada.

2.2 Computadores e robôs:

Assim como computadores, os sistemas robóticos possuem algum tipo de estrutura ou arquitetura. Podem ser programados com linguagens que vão desde o *solder*, onde os sensores são conectados mais ou menos diretamente aos motores, até linguagens de alto nível, como LISP e Java.

Programas de robô também devem seguir alguma metodologia eficiente de programação, como a escrita do *software* em módulos, para facilitar tanto a manutenção como o crescimento do sistema.

Apesar de muitas semelhanças, os robôs são diferentes dos computadores. As diferenças fundamentais estão relacionadas aos objetivos e restrições (Jones, 2004)

Como exemplo de algumas particularidades entre os robôs e os computadores, podemos citar:

- Serial x paralelo: em geral, um programa de computador calcula uma resposta e retorna um resultado. O processo serial consiste numa série de passos, onde a saída de um é à entrada do outro. Um robô móvel e autônomo não calcula uma resposta como os seres vivos, mas procura atingir um objetivo ou manter-se num estado, ao mesmo tempo em que evita situações de risco e armadilhas;
- Planos x oportunidades: um programa típico de computador executa um plano até o fim, enquanto que um robô móvel autônomo atua no mundo real aproveitando as oportunidades que surgem. Se esse robô

seguir um plano cegamente, sem aproveitar as oportunidades que surgem, exibirá comportamentos ineficientes e, no mínimo, estranhos;

- Queda gradual do sistema (*graceful degradation*): O resultado retornado por um programa de um computador, ou de um robô, tem uma forte dependência dos dados de entrada. Os dados não apropriados inseridos pelo ser humano ou mesmo sensores eletrônicos, podem, de alguma forma, causar uma queda abrupta na performance do sistema. Os robôs devem possuir a habilidade de persistir em funcionamento mesmo durante a ocorrência de erros e com um nível reduzido de desempenho, quando os fatores do mundo real dificultar o cumprimento das tarefas.

A abordagem comportamental é excepcionalmente boa pela execução em paralelo, a realização de objetivos de maneira oportunista e capacidade de gerenciar a queda gradual do sistema.

2.3 Robôs Inteligentes:

Para entendermos o estado atual da Robótica Comportamental (*behaviour-based robotics*), veremos desenvolvimentos históricos relevantes em três áreas relacionadas: Cibernética, Inteligência Artificial e Robótica (Arkin, 1998).

2.3.1 Precursores da Cibernética

No final da década de 1940, Norbert Wiener liderou o desenvolvimento da Cibernética: uma combinação de Teoria do Controle,

Ciência da Informação e Biologia visando explicar os princípios comuns do controle e da comunicação tanto nas máquinas como nos animais. A matemática desenvolvida para sistemas de controle de *feedback* contribuiu com a visão de um organismo como uma máquina (Weiner, 1948).

2.3.1.1 Tartarugas de Walter

Em 1953, W. Grey Walter aplicou princípios da Cibernética na criação de um projeto robótico pioneiro chamado *Machina Speculatrix*, que foi posteriormente passada para *hardware* como Tartaruga de Grey Walter.

A tartaruga era um dispositivo analógico consistindo em uma fotocélula, um sensor de toque, um motor de propulsão, outro de direção e duas válvulas (as "células nervosas"), lembrando um triciclo. O motor propulsor ficava no garfo da roda frontal. O motor de direção ficava no corpo e girava a estrutura de direção em 360 graus, sempre para o mesmo lado, assim fazendo com que o robô descrevesse uma trajetória com laçadas. A fotocélula era fixada à estrutura de direção, apontando sempre para a direção do movimento.

Com uma arquitetura tão simples, a Tartaruga exibia os seguintes comportamentos:

- Buscar-luz: enquanto o motor de direção continuamente movia o robô, o sensor rodava até que uma luz fraca fosse detectada;
- Seguir-luz: uma vez que uma luz fraca fosse detectada, a tartaruga movia-se em sua direção;

- Evitar-luz: um comportamento aversivo repelia a tartaruga da luz forte. Esse comportamento era muito útil quando a tartaruga estava recarregando a sua bateria.

- Evitar-obstáculo: usado para evitar obstáculos, empurrando-os para o lado ou virando ao redor deles;

- Recarregar: quando a bateria estava fraca, a tartaruga percebia a luz forte como fraca. Como a estação de recarga tinha uma forte luz sobre si, o robô movia-se em sua direção e se atracava. Depois da recarga, a fonte de luz era percebida como forte, o que fazia o robô deixar a estação.

Além disso, a Tartaruga de Walker já possuía um esquema de arbitragem de comportamentos: executava sempre o comportamento de maior prioridade em dada situação.

Alguns dos princípios embutidos nesse projeto incluem (Walter, 1953):

- Parcimônia: as variações dos padrões de comportamento, mesmo com uma estrutura tão simples, são complexas e imprevisíveis. Por isso, há uma valorização dos reflexos simples como base dos comportamentos;

- Exploração e especulação: o movimento constante é adequado sob circunstâncias normais para manter o robô longe de armadilhas. O sistema só fica imóvel enquanto estiver recarregando;

- Atração (tropismo positivo): o sistema tende a mover-se na direção de certo objeto. No caso de uma Tartaruga de Walter, é uma luz de intensidade média;

- Aversão (tropismo negativo): o sistema evita certos estímulos negativos, tais como obstáculos pesados e lombadas;

- Discernimento: o sistema tem a habilidade de distinguir entre comportamento produtivo e improdutivo, adaptando-se à situação logo no momento seguinte.

Uma arquitetura recente emprega as idéias de tropismo positivo e negativo de Walker na criação de robôs baseados em comportamento (Agah e Bekey, 1997).

2.3.1.2 Veículos de Braitenberg

Valentino Braitenberg, psicólogo, reviveu essa tradição três décadas depois de Walter. Usou circuitos analógicos para realizar uma série de experimentos do pensamento, projetando um grupo de veículos. Tais sistemas usavam influências inibitórias e excitatórias, ligando diretamente os sensores aos motores. Através dos seus inúmeros tipos de veículos, uma grande gama de comportamentos foram implementados. Assim como a Tartaruga de Walter, esses sistemas eram máquinas personalizadas, não reprogramáveis, e comportamento aparentemente complexo surgia de simples transformações senso-motoras (Braitenberg, 1984).

Posteriormente, cientistas do Media Lab do MIT criaram verdadeiros robôs inspirados nas criaturas de Braitenberg. Com blocos de LEGO

modificados construíram doze criaturas. Outras mais complexas também foram montadas e atribuídas com traços de personalidade, tais como persistência, coerência, desumanidade ou espírito de observação (Hogg, Martin *et al.*, 1991).

2.3.2 Precursores da Inteligência Artificial

O nascimento da Inteligência Artificial (IA) como um campo distinto é geralmente associado ao *Dartmouth Summer Research Conference*, em 1955. Os objetivos dessa conferência envolveram o estudo do uso da linguagem, redes neurais, teoria da complexidade, auto-melhoramento, abstrações, criatividade e outros tópicos. Na proposta original, Marvin Minsky indica que uma máquina inteligente "tenderia a elaborar dentro dela um modelo abstrato do ambiente na qual está localizada. Se um problema fosse dado, ela iria antes explorar soluções dentro do modelo do ambiente e daí tentar experimentos externos" (Mccarthy, Minsky *et al.*, 1955).

Alguns dos exemplos mais conhecidos da tradição de planejamento de IA incluem:

- STRIPS (*Stanford Research Institute Problem Solver*): esse planejador automático foi desenvolvido para a demonstração de teoremas. Um planejador típico recebe três entradas: uma descrição do estado inicial do mundo, uma descrição do objetivo desejado e um conjunto de possíveis ações, todos codificados numa linguagem formal (neste caso, também chamada STRIPS). O planejador produz uma seqüência de ações que levam

do estado inicial a um estado que satisfaça o objetivo (Fikes e Nilsson, 1971);

- ABSTRIPS: refinamento do STRIPS; usa uma hierarquia de espaços de abstração para melhorar a eficiência de planejadores do tipo STRIPS, refinando os detalhes de um plano conforme eles se tornassem importantes (Sacerdoti, 1974);

- HACKER: esse sistema busca, através de uma biblioteca de procedimentos, propor um plano para posteriormente ser analisado. O domínio do mundo dos blocos (blocos de brinquedo colocados em movimento por um braço robótico simulado simplificado) servia como um ponto de demonstração primário (Sussman, 1975);

A influência da IA na Robótica foi baseada na idéia de que o conhecimento e suas representações são fundamentais para a inteligência; contudo, as noções de sentir e agir dentro do ambiente começaram a ganhar preeminência sobre este paradigma.

Avanços importantes na Robótica e em *hardware* de sensores possibilitaram o teste das hipóteses da comunidade da Robótica Comportamental (Arkin, 1998).

O surgimento e crescimento da Inteligência Artificial Distribuída (IAD) ocorreram paralelos a esses avanços. Com o sistema Pandemônio (Selfridge e Neisser, 1960), começou a tomar força a noção de que múltiplos processos competitivos ou cooperativos (agentes) são capazes de gerar comportamento coerente. A Teoria da Sociedade da Mente de Minsky

propunha sistemas multiagentes como a base para toda a inteligência, através da interação coordenada e concatenada entre cada agente simples, uma inteligência altamente complexa poderia emergir (Minsky, 1986). Comportamentos individuais podem assim serem vistos como agentes independentes na Robótica Comportamental, relacionando-a mais próxima da IAD.

2.3.3 Precursores da Robótica

Nas décadas de 1960 e 1970 destacaram-se os seguintes projetos:

- Shakey: construído no final da década de 1960 no *Stanford Research Institute* (SRI), foi um dos primeiros robôs móveis e demandou o desenvolvimento do STRIPS. Em sua base, dois motores controlados independentemente acionavam cada uma das duas rodas traseiras, que guiavam à roda frontal de direção. Mais acima, na periferia do corpo do robô, sensores do tipo "antena" ("*cat whiskers*") serviam para detectar colisões.

Em seu topo, havia uma câmera de vídeo e um sensor de alcance óptico. O robô habitava um mundo artificial em uma área de escritório com objetos especialmente coloridos e moldados para auxiliá-lo no reconhecimento de objetos através da visão. O planejador usava informação armazenada dentro de um modelo de mundo simbólico, atualizado com os dados da percepção., e planejava uma ação antes de executá-la (Nilsson, 1969);

- HILARE: Iniciou-se por volta de 1977 no Laboratoire d'Automatique et d'Analyse des Systèmes (LAAS) em Toulouse, França. Esse robô era equipado com três rodas: duas motoras e uma frontal.

Para sensoriamento, usava uma câmera de vídeo, 14 sensores ultra-sônicos e um sensor de alcance laser. Seu mundo continha o piso plano e liso, típico de um escritório. O planejamento era conduzido conforme um espaço de representação multinível: modelos geométricos representavam distâncias e medidas atuais dos mundos, e um modelo relacional expressava a conexão entre salas e corredores (Giralt, Chatila *et al.*, 1984);

- Stanford Cart: usado para testar a visão estéreo como meio de navegação. Avançava aproximadamente um metro a cada dez ou quinze minutos, com uma rodada completa de aproximadamente cinco horas. O processamento visual era o que mais consumia tempo, mas o carrinho navegava com sucesso pelos obstáculos detectados visualmente e adicionados ao seu mapa interno na forma de esferas. Aplicava um algoritmo de busca utilizando um grafo, para encontrar o caminho mais curto (Moravec, 1977);

- CMU Rover (1980): Era um robô cilíndrico com três pares de rodas guiáveis e alimentadas independentemente, capaz de carregar uma câmera montada em um mecanismo *pan/tilt* e dotado de sensores infravermelhos e ultra-sônicos. Esse robô foi sucedido por vários outros CMUs (Arkin, 1998).

2.4 Espectro do controle robótico

Dentre as estratégias de controle dos robôs atuais, consideraremos um espectro que tem nas suas extremidades, sistemas deliberativos e sistemas reativos (figura 2-1) (Arkin, 1998):

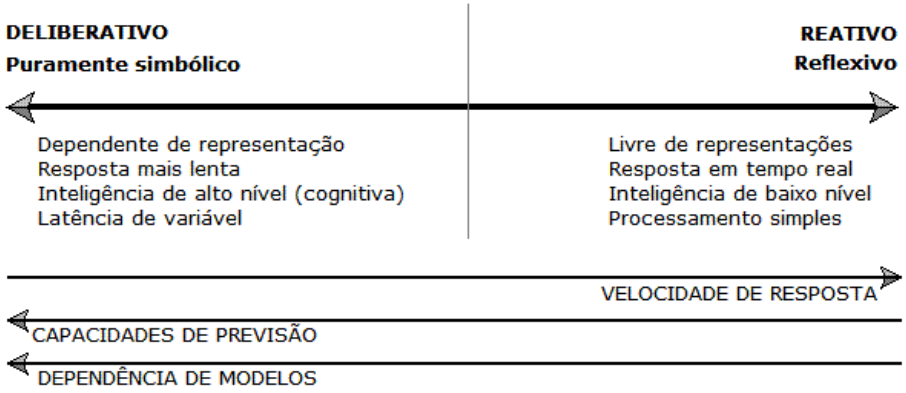


Figura 2-1 - Espectro do sistema de controle de um robô.

2.4.1 Sistemas deliberativos

Um robô empregando pensamento deliberativo requer conhecimento relativamente completo sobre o mundo para predizer o resultado de suas ações. Se a informação usada não for detalhada ou estiver desatualizada, o resultado do processamento poderá ficar seriamente comprometido. Em um mundo dinâmico, onde objetos podem se mover arbitrariamente, é potencialmente perigoso confiar em informações prévias inválidas (por estarem desatualizadas). Os modelos de representação são geralmente alimentados com os dados inseridos pelos sensores.

Sistemas de raciocínio deliberativo freqüentemente têm várias características comuns:

- Possuem uma estrutura hierárquica com uma subdivisão de funcionalidade claramente identificável;

- A comunicação e o controle ocorrem de maneira previsível e pré-determinada, subindo e descendo a hierarquia, com pouco, ou até mesmo sem movimento lateral;

- Os níveis mais altos da hierarquia passam submetas para níveis subordinados;

- O escopo de planejamento, tanto espacial como temporal, muda durante a descida pela hierarquia. Nos níveis inferiores, as necessidades de tempo são menores e considerações de espaço são mais locais;

- Forte dependência de modelos de mundo representados por símbolos.

2.4.2 Sistemas reativos

De maneira simples, controle reativo envolve o casamento estreito entre percepção e ação, o que produz uma pronta resposta em um ambiente dinâmico e não-estruturado.

Sistemas robóticos reativos têm as seguintes características (Arkin, 1998):

- Base das ações nos comportamentos: um comportamento consiste em um simples par senso-motor, com a atividade sensória

provendo a informação necessária para satisfazer a aplicabilidade de uma resposta reflexo-motora de baixo nível;

- O uso de conhecimento representacional é evitado e a reação ao mundo é direta, conforme este seja sentido. Isso é de valor particular em mundos altamente dinâmicos e perigosos, onde as imprevisibilidades e hostilidades potenciais são inerentes. Construir modelos abstratos do mundo é um processo que demanda muito tempo e sujeito a falhas, o que reduz o acerto potencial de uma ação do robô, exceto nos mundos mais previsíveis;

- Inspiração no comportamento animal: A biologia tem nos mostrado que muitas das tarefas que gostaríamos que nosso robô executasse são factíveis. Além disso, as ciências biológicas, tais como a neurociência, a etnologia e a psicologia têm elucidado vários mecanismos e modelos que poderão ser aplicados.

Algumas das características dos insetos que podem servir de modelo para as funcionalidades de um robô móvel autônomo programado, segundo o paradigma comportamental, (Jones, 2004) são:

- Habilidade em procurar o suprimento de necessidades: alimento, abrigo e parceiros;
- Habilidade em evitar situações de risco ou perigo, como predadores;
- Navegar pelo ambiente sem se perder;
- Em alguns casos, cooperar na construção de grandes estruturas e outros grandes feitos;
- Uso de um cérebro minúsculo;

- Uso de um sistema de visão bastante rudimentar.

- *Software* modular: Possibilita a expansão do sistema através da adição de novos comportamentos sem ter que alterar ou descartar os antigos. É muito útil para a construção de sistemas robóticos cada vez mais complexos.

Enfim, podem-se verificar alguns aspectos-chave dessa metodologia comportamental:

- Localização (*situatedness*): O robô é uma entidade situada no mundo real e envolvida por ele. Não opera sobre representações abstratas a realidade, mas sobre a realidade em si (Brooks, 1991). O ambiente é altamente dinâmico, tanto no espaço como no tempo. Assim, para um agente robótico ser projetado apropriadamente, deve-se considerar as oportunidades e perigos do meio. Na natureza, os processos evolucionários moldam os agentes para se encaixarem em seus nichos ecológicos, mas essas escalas de tempo não estão disponíveis para o roboticista. Adaptação, contudo, pode ser crucial;

- Corporificação (*Embodiment*): Um robô tem uma presença física (um corpo). Essa realidade espacial tem conseqüências em suas interações dinâmicas com o mundo que não podem ser simuladas fielmente (Brooks, 1991). De fato, uma crítica recorrente à pesquisa de IA tradicional é em relação a sua base simbólica, isto é, o sistema pensa através de símbolos que não têm nenhuma correlação com a realidade física. O mundo desses sistemas vive em estado de alucinação. Simulações de robô é um

exemplo, com "robôs" parecendo estar sentindo e agindo, quando apenas criam novos símbolos a partir de velhos. A corporificação força um robô a funcionar conforme seu ambiente: sentindo, agindo e sofrendo as conseqüências de suas falsas percepções e concepções (Flynn e Brooks, 1989).

- Emergência: É a inteligência que surge como resultado das interações do agente robótico com seu ambiente. Não depende unicamente do robô nem unicamente do meio;

- Escala da inteligência: Embora muitas tarefas precisem apenas da competência de criaturas como insetos, têm sido discutidos se a inteligência não deveria abranger seu nível humano.

2.5 Comportamentos

Segundo a escola behaviorista de psicologia, um comportamento, simplesmente falando, é a reação a um estímulo. Essa visão pragmática permite que expressemos como um robô deverá interagir com seu ambiente, num paradigma puramente reativo.

2.5.1 Comportamento emergente

É o comportamento resultante da interação de comportamentos primitivos em atividade. Pode não ser implementado pelo programador. Para que o sistema exiba tal comportamento, devem-se definir os comportamentos primitivos considerando o(s) comportamento(s) emergente(s) desejado(s).

2.5.2 Comportamento primitivo

É um conceito genérico que impõe algumas restrições no código da implementação. Da maneira que é entendido em Robótica Comportamental, comportamentos primitivos são compostos de duas partes (Jones, 2004):

1. Um sistema de controle, que transforma cada informação recebida pelos sensores em comandos atuadores;

2. Um gatilho (*trigger*), que determina quando o componente de controle deve agir.

Suponha que robôs sejam programados para recolher algumas bolinhas da arena de teste para uma região iluminada. Quando os sensores do pára-choque acusarem a colisão com alguma bolinha, então o robô empurra a bolinha até a região iluminada. Quando não estiver empurrando nenhuma bolinha, ele procura por uma. Assim, quando o robô encontra uma bolinha para empurrar, o gatilho é disparado, informando que navegar em direção a luz é necessário. Por sua vez, o sistema de controle do comportamento conduz o robô até seu destino, através dos dados colhidos pelas fotocélulas.

2.5.3 Comportamentos servo e balístico

Os Comportamentos Primitivos podem ser classificados em duas categorias (Jones, 2004):

- Comportamento servo: tipicamente implementa o controle de *feedback* em um *loop*, respondendo imediatamente a mudanças nas entradas dos sensores. Um exemplo seria um robô que busque uma fonte

de luz. Sempre que a luz for considerada pelo sistema, o gatilho será disparado, chamando o comportamento;

- Comportamento balístico: uma vez disparado, executa um determinado conjunto de operações até o fim, um ponto final bem demarcado. Comportamentos balísticos têm a desvantagem de deixar o robô cego em relação ao ambiente durante a execução, por isso devem ser implementados com cautela.

Ao sobrepormos comportamentos, estes se tornam muito complexos e começamos a perder os benefícios da abordagem comportamental. A idéia é escrever um conjunto de comportamentos simples, cada um dos quais lidará com uma situação bem específica.

Cruise

Seja um robô provido de duas rodas traseiras independentemente motorizadas e pelo menos uma roda frontal livre. O comportamento Cruise deve manter esse robô andando em linha reta. As velocidades da roda esquerda v_esquerda e da roda direita v_direita têm o mesmo valor v em módulo e em sinal:

```
Behavior Cruise

  v_esquerda = v

  v_direita  = v

end Cruise
```

Adicionando a essa função um segundo parâmetro b de pequeno valor em módulo, podemos escrever o comportamento CruiseB, codificado como:

```
Behavior CruiseB

  v_esquerda = v - b

  v_direita  = v + b

end CruiseB
```

onde:

- Se b > 0, o robô descreverá uma curva para a esquerda;
- Se b = 0, o robô entra no comportamento Cruise típico;
- Se b < 0, o robô irá virar-se para a direita;
- Quanto maior |b|, mais fechada será a curva executada pelo robô.

Home

Esse comportamento é disparado por um gatilho e define a movimentação de um robô em relação a uma determinada fonte de luz: em direção à luz, indiferente à sua presença ou evitando-a. Para suportar esse comportamento, é preciso que o robô tenha duas fotocélulas, uma localizada à esquerda e outra, à direita. O sistema do robô também deverá fornecer outras variáveis, além de v:

Behavior Home

```
if ( (fc_esquerda + fc_direita)/2 > luz_minima) then

v_esquerda = v + reatividade*(fc_direita - fc_esquerda)

v_direita  = v - reatividade*(fc_direita - fc_esquerda)

end if

end Home
```

onde:

- fc_esquerda e fc_direita correspondem à quantidade de luz captada pelas fotocélulas esquerda e direita, respectivamente;
- luz_minima é o valor mínimo da intensidade da luz para que o comportamento seja acionado. Se a média dos valores das duas fotocélulas for maior que esse valor, então o comportamento será ativado;
- reatividade determinará o tipo da reação do robô em relação à luz, onde:
  - Se reatividade > 0, então o robô andará em direção à luz;
  - Se reatividade = 0, a luz será ignorada;
  - Se reatividade > 0, o robô fugirá da luz;

Escape

Para a implementação desse comportamento, é preciso que o robô tenha pelo menos dois sensores de colisão. Uma vez detectada a trombada, Escape busca afastar o robô da possibilidade de uma nova colisão.

A resposta do robô dependerá da sua geometria. Em robôs de simetria cilíndrica, a primeira ação é recuar o suficiente para descomprimir os sensores, permitindo que ele possa girar com facilidade na segunda etapa. A terceira operação resume-se em andar para frente em linha reta por certa distância. Note que a interrupção do sinal de colisão ao recuar dificulta a implementação do Escape como comportamento servo.

Comportamentos servos duram apenas momentaneamente, computando o que fazer apenas naquele momento. Não guardam informações sobre o que fizeram no passado, nem se preparam para futuras ações. Diz-se que tais comportamentos não têm estado, ou seja, não têm memória.

Comportamentos que suportem estados podem ser projetados com a ajuda das Máquinas de Estados Finitos (MEF), com regras definidas quanto à transição do sistema de um estado para o outro. Podemos estocar informação em uma variável em vez de usar um estado explícito, se isso simplificar o sistema: usando uma variável d, vejamos a MEF por partes (Figura 2-2):

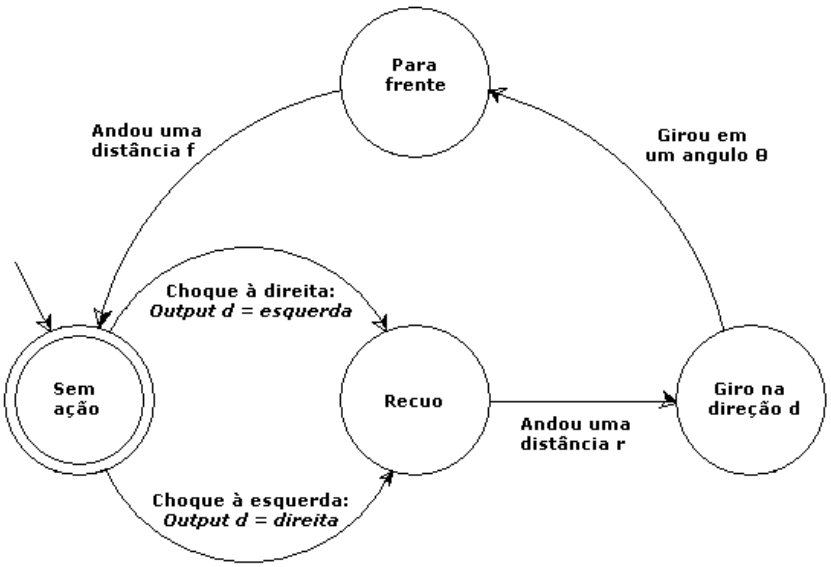


Figura 2-2. MEF do Escape (fonte: JONES 2004)

1. Antes de entrar no Escape, ou logo depois de ter saído dele, o robô encontra-se no estado "Sem ação";

2. Quando da ocorrência de uma colisão, d recebe o lado para o qual o robô deverá girar e o sistema passa para o estado "Recuo", que fará o robô recuar. Se o choque foi à direita, d irá informar que o robô deverá girar para a esquerda (anti-horário) e vice-versa;

3. No estado de "Recuo", assim que tiver recuado uma distância r, o robô passa para o estado "Giro na direção d";

4. Após ter girado um ângulo Θ, o robô deixa o estado "Giro na direção d" e entra no estado "Para frente";

5. No estado "Para frente", assim que tiver avançado uma distância f, ele deixa o comportamento, entrando no estado "Sem ação".

O ângulo Θ de rotação do robô é um parâmetro do sistema, escolhido de forma a balancear restrições (Jones, 2004):

- Para Θ grande (aprox. 90 graus ou mais): o robô irá em geral se desvencilhar facilmente de grandes obstáculos, mas irá falhar para mover-se por pequenas passagens;
- Para Θ pequeno: o robô irá se mover mais facilmente na desordem, mas precisará de muitos movimentos para trás e para frente ao executar um Escape trivial a partir de uma parede.

Na implementação, uma versão mais simples da MEF freqüentemente basta, usando-se tempo decorrido em vez de distância e ângulo. Os tempos t_g e t_r são ajustados experimentalmente (Figura 2-3):

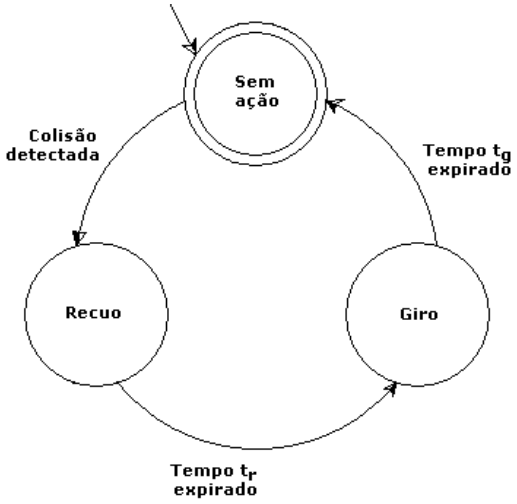


Figura 2-3 - MEF simplificada do Escape, usando tempo decorrido na implementação (fonte: JONES 2004)

Seguir-linha:

Um comportamento de particular interesse é o Seguir-linha, que pode ser implementado como um Home diferencial: sensores diferenciais detectam a presença da linha à direita e à esquerda do robô. A extremidade da linha mais próxima do sensor esquerdo significará uma virada para a esquerda, e vice-versa. Andar centralizando a linha entre os dois sensores fará com que a linha seja seguida. Por causa da pequena região de captação, Seguir-linha poderá precisar de sensores adicionais e comportamentos de apoio para uma performance ótima (Figura 2-4):

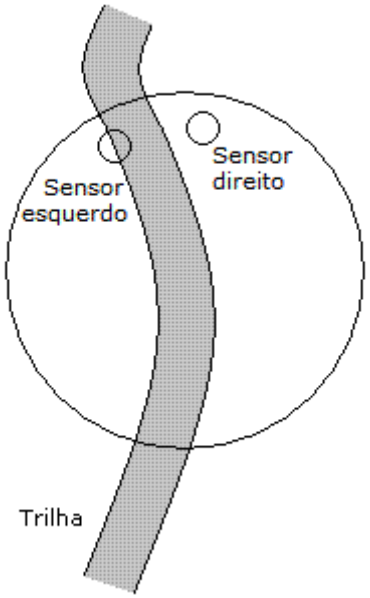


Figura 2-4 - O objetivo do Seguir-linha é seguir uma faixa no chão, mantendo-a entre dois sensores (fonte: JONES 2004).

2.5.4 Arbitragem

Um recurso robótico é escasso se, por algum momento, a demanda por aquele recurso puder exceder seu suprimento. Sempre que dois ou mais comportamentos competirem por um mesmo recurso será necessária a

arbitragem. Como exemplo trivial, temos as duas rodas dos robôs, que podem ser disputadas por vários comportamentos.

Para qualquer método usado para resolver conflitos, devemos nos guiar pelas necessidades da totalidade do sistema. Nesse caso, o robô conseguirá executar operações vitais sempre que necessário. Qualquer estratégia que não permita a execução dos comandos dos comportamentos de maior prioridade quando esses forem invocados, ou que dilua tais comandos, irá reduzir a habilidade do robô em executar sua tarefa.

Na arbitragem por prioridade fixa, o programador decide de antemão qual comportamento deverá ganhar a disputa sempre que um conflito ocorrer, com a atribuição de prioridades mais altas para comportamentos mais importantes. Assim, o comportamento de maior prioridade será atendido, enquanto o(s) outro(s) será, ou serão completamente ignorado(s) (Figura 2-5):

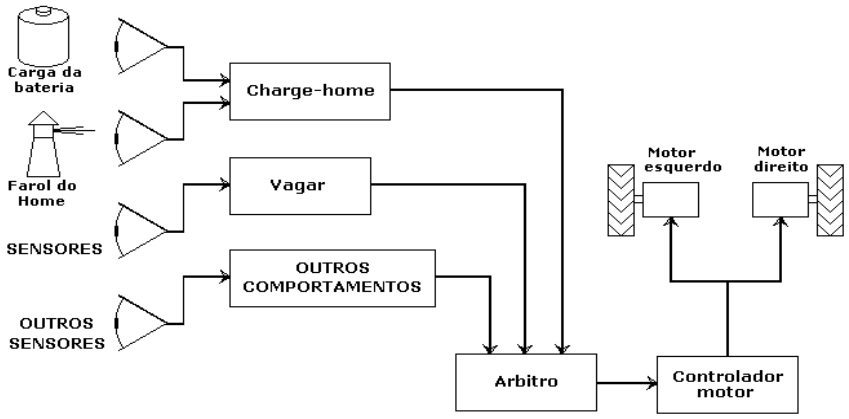


Figura 2-5 - Arbitragem por prioridade fixa. Comportamentos localizados mais acima no esquema terão maior prioridade (fonte: JONES 2004).

Os comportamentos precisam mandar comandos apenas para os árbitros relevantes. A aprovação de um árbitro não necessariamente

significa que o comportamento será aprovado por outros árbitros com os quais está conectado, pois vários árbitros podem estabelecer prioridades diferentes para o mesmo comportamento (Jones, 2004).

Em geral, comportamentos funcionam sem conhecimento do estado de nenhuma outra parte do sistema além dos sensores, mas pode ser necessário a um comportamento saber se o controle foi concedido. Logo, os árbitros são freqüentemente implementados com uma saída indicando o vencedor da disputa. Junto com os comandos que cada comportamento envia para o árbitro, vai um símbolo único que identifica o comportamento. Assim, se um comportamento encontrar seu próprio símbolo na linha de saída do árbitro, então saberá que venceu a disputa e que estará em controle.

Queda gradual do sistema

Como mostrado no exemplo a seguir, a estrutura comportamento/arbitragem leva naturalmente a aplicabilidade da queda gradual do sistema, que simplesmente emerge:

1. Início sob circunstâncias normais: o comportamento Navegar é executado com o apoio do Evitar-por-sonar. O último monitora os sensores de sonar de longo alcance, mantendo o robô navegando a alguma distância dos obstáculos detectados;
2. Primeira queda gradual do sistema: os sonares não detectam uma quina de parede de superfície lisa, acusando uma falsa negativa para o robô, que segue em direção à parede.

Contudo, os sensores infravermelhos poderiam detectar o obstáculo conforme o robô se aproxima, acionando o comportamento Evitar-por-Infravermelho;

3. Segunda queda gradual do sistema: infelizmente, a parede é muito escura no infravermelho. Logo, Evitar-por-Infravermelho não será invocado e o robô seguirá em frente com essa falsa negativa. O robô colide, mas os sensores de choque mecânico poderiam ativar o comportamento Escapar-de-colisão, que faria o robô recuar, rotacionar e mover-se para outra direção;

4. Terceira queda gradual do sistema: os sensores de colisão também reportam uma falsa negativa, fazendo o robô insistir em mover-se contra o obstáculo, aumentando a corrente nos motores. Mantida por certo tempo, essa corrente elevada seria detectada por um sensor que acionaria o comportamento Stall-escapar. O robô apenas recuaria e giraria, uma vez que não há a informação sobre a localização do bloqueio em relação ao robô;

5. Quarta queda gradual do sistema: além de tudo isso, o piso liso está escorregadio: as rodas patinarão, nos motores não haverá alta corrente e Stall-escapar não será acionado. Contudo, o robô tem um sensor de estase, ou seja, um sensor que detectaria a falha nas rodas. Assim, o comportamento Escapar-de-estase executa uma espécie de procedimento de pânico onde o robô poderá rotacionar e recuar, possivelmente de

maneira aleatória, numa tentativa de se livrar dessa falha mais genérica. Nessa situação, o robô atingiu seu nível mais baixo de performance. Mas mesmo assim, o robô mantém a possibilidade de realizar sua tarefa depois de certo tempo.

Mesmo que a causa dos problemas fosse sensores permanentemente inoperantes, teríamos obtido a mesma série de negativos falsos. No caso de falsos positivos constantes, quando o sensor reporta uma condição inexistente, será preciso uma programação mais sofisticada (Jones, 2004).

Note que em Robótica Comportamental não empregamos um único sensor caríssimo do qual devemos obter níveis inatingíveis de precisão e confiabilidade. Em vez disso, conseguimos uma performance robusta usando uma combinação de sistemas relativamente não-confiáveis funcionando juntos.

Poeira, sujeira, vibração e desgaste em geral prejudicam a operação dos sensores. Para maximizar a confiança, um robô não deve depender de nenhum sensor defeituoso. No caso de um sinal negativo falso, é fácil assegurar que o sistema irá cair gradualmente, mas temos que considerar também a possibilidade de um falso positivo.

As duas maneiras pelas qual a avaliação dos sensores pode ser realizada e os seus resultados passados para a estrutura de controle são:

- Na primeira maneira (Figura 2-6), adiciona-se um comportamento avaliador da funcionalidade de um sensor. É feita uma conexão do Avaliador para cada comportamento que poderia ser afetado

adversamente por indicações falsas positivas. Cada um desses comportamentos pode ser mantido em um estado inicial, mandando nenhum comando motor, até que o Avaliador tivesse verificado que o sensor está funcionando adequadamente.

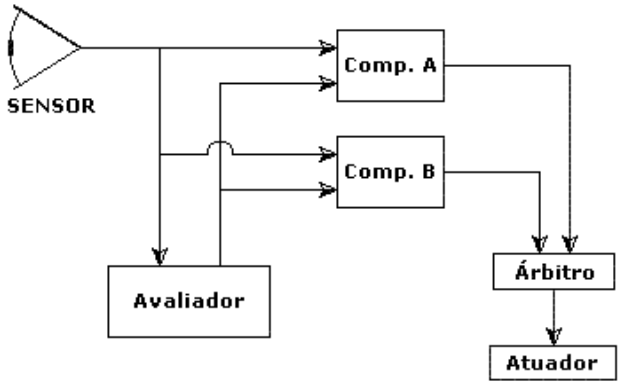


Figura 2-6. Comportamento avaliador do sensor (fonte: JONES 2004).

- Um segundo método é implementar um comportamento sensor virtual (Figura 2-7). O sensor virtual é alimentado pelo sensor real, mas não passa indicações positivas até que a funcionalidade do sensor tenha sido verificada.

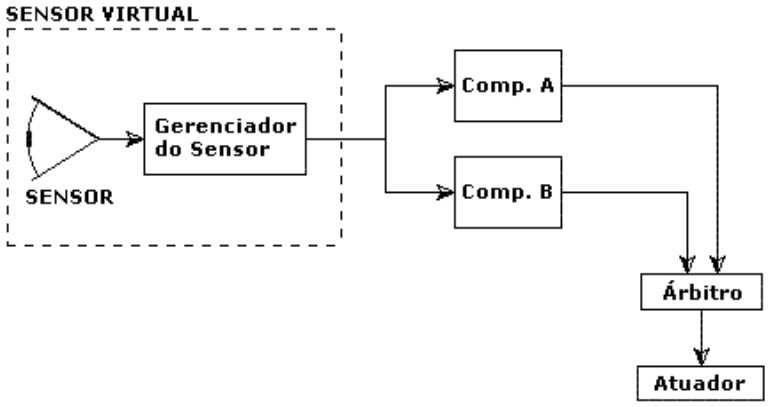


Figura 2-7. Uso de um sensor virtual (fonte: JONES 2004).

Em muitos casos, a verificação pode ser um tanto fácil.

Suponha que quando o robô é ligado, o sensor reporta a presença de um objeto. Pode ser que exista realmente um objeto sendo detectado ou que seja uma indicação positiva falsa. Se, conforme o robô se mover, a indicação desaparecer, então poderemos concluir que o robô simplesmente foi iniciado perto de um obstáculo.

Dentro do Avaliador será observado se o sensor passa pelos dois estados, de detecção e não-detecção, conforme a MEF abaixo (Figura 2-8):

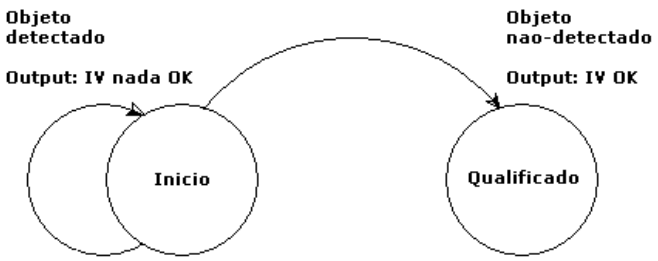


Figura 2-8. MEF de avaliação contra falsos positivos (fonte: JONES 2004).

Adicionar *software* que avalie sensores em geral não custa nada (exceto o tempo de desenvolvimento) e pode melhorar grandemente a robustez do robô (Jones, 2004).

2.6 Métodos de projeto de comportamentos

A seguir, três metodologias utilizadas atualmente para o projeto de comportamentos. O melhor método a ser empregado será aquele que faça o

robô produzir a resposta mais adequada para uma dada tarefa e ambiente (Arkin, 1998):

1. Projeto etologicamente guiado/restringido (*Ethologically guided/constrained design*): Estudos sobre o comportamento animal podem prover grandes *insights*, sendo a ênfase aqui no termo "guiado". Um modelo é feito baseado em um estudo científico, de preferência com um pesquisador da Biologia. O modelo animal é adaptado para ser instalado computacionalmente e embasado conforme as capacidades senso-motor do robô. Os resultados do experimento robótico são comparados aos resultados dos estudos originais biológicos, e tanto o modelo biológico como o robô são modificados e aperfeiçoados numa tentativa de produzir resultados mais coerentes com os dados originais sobre o animal (Figura 2-9);

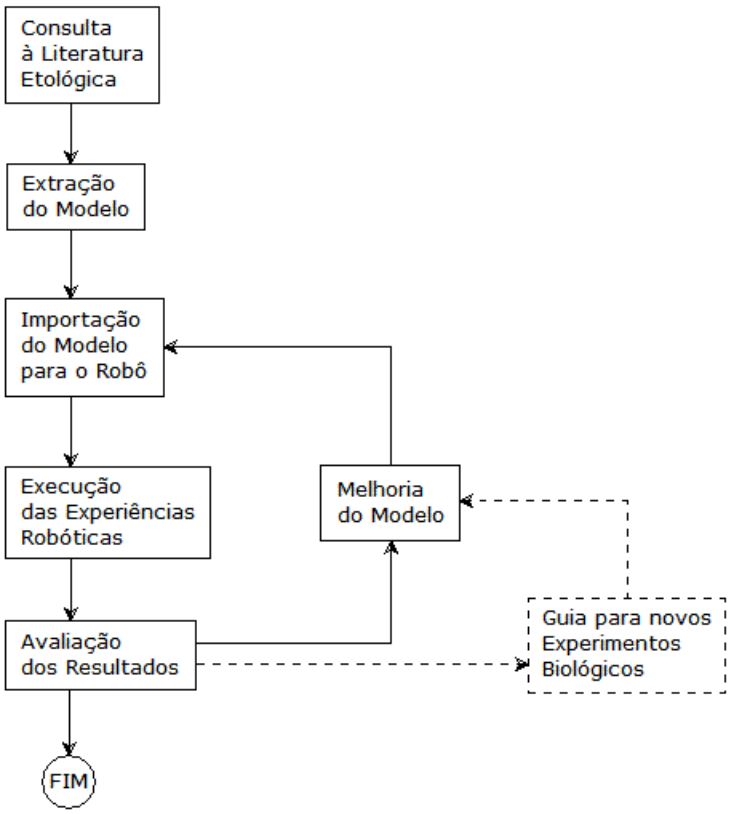


Figura 2-9. Projeto etologicamente guiado/restringido (fonte: ARKIN 1998).

2. Projeto baseado na atividade localizada (*Situated activity-based design*): As ações do robô dependerão das situações enfrentadas. O robô deve reconhecer as situações e escolher uma ação (ou talvez várias) para ser(em) executada(s). Essa gama de situações pode ser entendida como microcomportamentos, ou seja, comportamentos projetados para circunstâncias bem específicas. Nessa metodologia é necessário ter um entendimento sólido sobre a relação robô-ambiente (Figura 2-10).

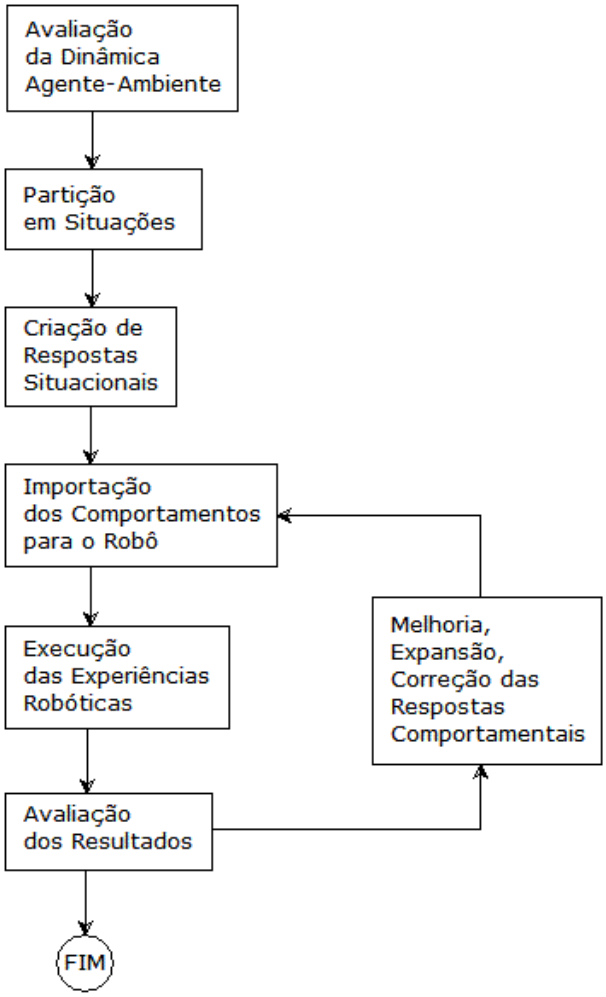


Figura 2-10. Projeto baseado na atividade localizada (fonte: ARKIN 1998).

3. Projeto guiado experimentalmente (*Experimentally driven design*): Os comportamentos são criados de uma maneira *bottom-up*. A idéia é começar dando ao robô um conjunto limitado de capacidades, depois fazer as experiências no mundo real, ver o que funciona e o que não, corrigir comportamentos imperfeitos e então adicionar novos comportamentos iterativamente, até que o sistema atinja uma atuação satisfatória (Figura 2-11).

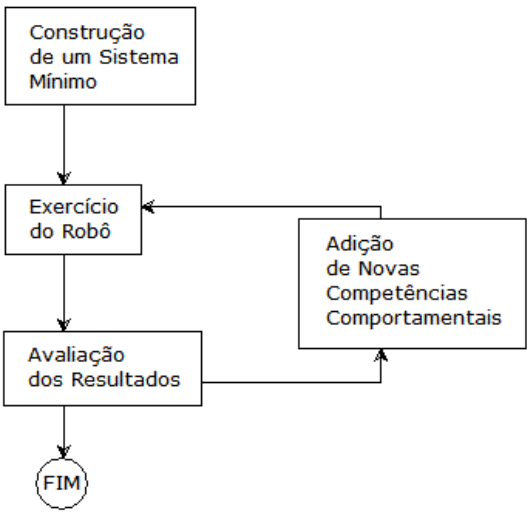


Figura 2-11. Metodologia guiada por experimentos (fonte: ARKIN 1998).

3. Lógica Paraconsistente Anotada

3.1 Lógicas Clássicas e Não-Clássicas

A Lógica Clássica, também chamada de Lógica Tradicional, contém, entre outros, estes quatro princípios:

- Princípio do Terceiro Excluído: de duas proposições contraditórias A e A, uma delas é verdadeira.

- Princípio da Não Contradição: de duas proposições contraditórias, A e A, uma delas deve ser falsa. Assim, se considerarmos as proposições “A Terra é redonda” e “A Terra não é redonda”, uma delas deve ser falsa.

- Princípio da identidade: todo objeto é idêntico a ele mesmo.

No mundo real, ocorrem indefinições e inconsistências, que muitas vezes são situações não tratáveis diretamente pela Lógica Clássica.

Podemos imaginar uma situação onde um determinado sensor de um robô indica que não há obstáculos à frente e ao mesmo tempo um outro sensor indica o contrário. Neste caso, diante de informações contraditórias, o robô não tem condições de tomar uma decisão considerando a Lógica Clássica.

Para resolver esses tipos de problemas, surgiram lógicas alternativas à Lógica Clássica, denominadas de Lógicas Não-Clássicas. Estas permitem que projetos de sistemas de controle considerem os conceitos lógicos

ausentes da Lógica Clássica (Da-Silva-Filho e Abe, 1999), mas onipresentes em situações reais.

De maneira geral, as Lógicas Não-Clássicas podem ser classificadas em dois grupos:

1. As que complementam a Lógica Clássica, com o enriquecimento de sua linguagem e escopo mediante a introdução conveniente de novos operadores e postulados. Exemplos: lógicas modais e lógicas temporais;

2. As que rivalizam a Lógica Clássica, também chamadas de heterodoxas. Estas modificam ou restringem os princípios da Lógica Clássica.

Ao infringir o Princípio do Terceiro Excluído, teremos as Lógicas Paracompletas, como a Lógica Intuicionista de Brouwer-Heyting e várias lógicas polivalentes;

Ao não satisfazer o Princípio da Não Contradição, pode-se trabalhar com informações contraditórias de modo não trivial por meio das Lógicas Paraconsistentes.

3.2 Breve história da lógica

A história da Lógica pode ser dividida em três períodos:

- Período Aristotélico: a partir da primeira sistematização realizada na Grécia antiga por Aristóteles (384-322 a.C.), incluindo a

lógica trivalente do inglês William of Ockham (c. 1285-1348) na Idade Média.

- Período Booleano: autores como G. Boole (1815-1864), A. de Morgan (1806-1871), W. S. Jevons (1835-1882) causaram uma grande evolução ao aplicarem idéias algébricas na Lógica, criando o método algébrico. Por outro lado, além de outras contribuições, G. Frege (1848-1925) propôs um método lingüístico (ou proposicional) da Lógica. Ambos os métodos são os principais, a partir daqueles anos, para o tratamento de um sistema lógico.

- Período Contemporâneo: iniciado a partir de 1900, neste período destacam:

  - A publicação de uma importante obra: Principia Mathematica, em três volumes;

  - A Lógica impõe-se como uma nova ciência graças às contribuições de vários autores;

  - A concepção de uma lógica heterodoxa, a Lógica Intuicionista, pelos holandeses L. E. J. Brouwer (1881-1966) e A. Heyting (1898-1980), que infringe o Princípio do Terceiro Excluído;

  - As contribuições da escola polonesa de lógica, em particular de J. Łukasiewics (1878-1956), que formulou os primeiros sistemas lógicos polivalentes posteriores à Idade Média;

  - A evolução da Lógica em passos gigantes a partir de 1930, com o surgimento de várias Lógicas Não-Clássicas, o grande

desenvolvimento do método algébrico e a conseqüente compreensão da integração da Lógica como um ramo da Matemática.

3.3 O surgimento da lógica paraconsistente

Os precursores da Lógica Paraconsistente foram o polonês J. Łukasiewics e o russo N. Vasilév. Por volta de 1910, eles indicaram a possibilidade de uma lógica paraconsistente ao restringirem o Princípio da Não Contradição. Com a modificação da silogística Aristotélica, Vasilév chegou a conceber uma lógica paraconsistente, por ele denominada imaginária, pensada mais ou menos à maneira aristotélica.

A Lógica Paraconsistente foi descoberta simultânea e independente do polonês S. Jaśkowski e do brasileiro Newton C. A. da Costa. Em 1948, S. Jaśkowski, discípulo de J. Łukasiewics, publicou suas idéias sobre lógica e contradição, mostrando como se poderia construir um cálculo sentencial paraconsistente. Sua lógica paraconsistente foi denominada de discursiva, mas tratou apenas do cálculo proposicional. Da Costa, por sua vez, desenvolveu infinitos cálculos proposicionais paraconsistentes, estendendo-os a cálculos de predicados com ou sem igualdade, teorias de descrições e teoria de conjuntos. Da Costa mostrou como podem ser construídas matemáticas paraconsistentes que generalizam a matemática tradicional.

Posteriormente, diversos autores, incluindo Da Costa, prosseguiram com o trabalho de S. Jaśkowski, fazendo com que a Lógica Paraconsistente progredisse a partir dos anos 1970, sendo valorizada em todo mundo.

3.4 Lógica paraconsistente

A Lógica Clássica pode ser entendida pelo seu núcleo, constituído do cálculo de predicados clássico de primeira ordem, com ou sem igualdade, com uma linguagem bem definida e uma semântica padrão. Esse núcleo estende-se às teorias de conjuntos e às lógicas de ordem superior (teorias de tipos) (Abe, Costa *et al.*, 1999; Costa e Abe, 2000).

Adotando um desenvolvimento lingüístico, uma lógica L é vista como uma classe de técnicas de derivação de um conjunto de proposições em novas proposições. Tais proposições são expressas por meio das sentenças de uma linguagem. Essa lógica L possui uma linguagem correspondente L' que contém operações, tais como disjunção ($\vee$), conjunção ($\wedge$), implicação ($\rightarrow$) e negação ($\neg$). Logo, uma lógica permite a realização de inferências e construção de teorias.

Consideremos ainda que essa lógica L comporte a operação de negação. Uma teoria T edificada mediante suas proposições é um conjunto de sentenças fechado pelas inferências aceitas pela lógica L. Em outras palavras, a teoria T conterá todas as conseqüências de suas sentenças, ou seja, todos os seus “teoremas”.

Vale observar a classificação de uma teoria T de duas formas:

- A teoria T é trivial se qualquer sentença da linguagem L' for um teorema. Caso contrário, existindo pelo menos uma sentença que não seja um teorema, então T é não-trivial.

- T é consistente se dela não puder ser derivada uma fórmula A e sua negação $\neg$A. Caso contrário, T diz-se inconsistente.

Na Lógica Clássica, não há separação entre teorias inconsistentes e triviais: se uma teoria T for inconsistente, então será também trivial, e vice-versa. Mas interessa-nos a separação desses dois conceitos: uma lógica que seja o fundamento de teorias inconsistentes, mas não sejam as teorias triviais, é uma lógica paraconsistente. (Abe, 1992).

3.5 Lógica paraconsistente anotada

A Lógica Paraconsistente Anotada (LPA) guarda uma relação íntima com um reticulado, o que possibilita a sua aplicação em situações de inconsistências de modo não-trivial. Os resultados do uso do reticulado podem ser analisados com a lógica convencional ou com outros tipos de lógicas não-clássicas, fazendo da LPA um forte instrumento de aperfeiçoamento técnico aplicável na engenharia eletrônica digital (Abe, Costa *et al.*, 1999). Os primeiros estudos da sintaxe e semântica sobre a LPA foram realizados no início dos anos 1990 por N. C. A. Da Costa, J. M. Abe, V. S. Subrahmanian e C. Vago. Ainda nesse período, V. S. Subrahmanian realizou o primeiro trabalho citando proposições anotadas em programação lógica e J. M. Abe, um estudo sistemático sobre as lógicas anotadas de primeira ordem, teoria dos modelos e teoria anotada de conjuntos.

3.5.1 Reticulado “quatro”

Inicialmente, seja um reticulado de Hasse $|\tau| = \{T, V, F, \perp\}$ com quatro constantes de anotação, a saber: Inconsistente (T), Verdadeiro (V), Falso (F) e Paracompleto ($\perp$). Suas anotações seguem as regras do diagrama de Hasse (Figura 3-1).

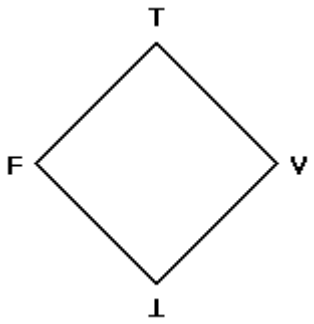


Figura 3-2. Reticulado “quatro”. (fonte: DA COSTA, ABE et al. 1999).

Ainda, consideramos um operador: $|\tau| \rightarrow |\tau|$, que atua sobre o reticulado e possui, intuitivamente, o "significado" da negação da lógica anotada:

- ~1 = 0;
- ~0 = 1;
- ~T = T;
- ~$\perp$ = $\perp$.

Para a proposição p ≡ "Há um obstáculo à frente", temos:

- P_1 = Há um obstáculo à frente (estado verdadeiro);
- P_0 = Não há um obstáculo à frente (estado falso);

- p_T = Há e não há um obstáculo à frente (estado inconsistente);
- $p_\perp$ = Nada se pode afirmar sobre a existência de um obstáculo à frente (estado paracompleto).

Intuitivamente, quando dizemos que "cremos em uma proposição p com grau de evidência μ" (que representaremos por $p\mu$) podemos considerar as constantes anotacionais do reticulado como graus de evidências. É interessante a maneira de pensar vinda com a LPA:

> "Ao abandonar as “verdades lógicas certas” que, quando trazidas à realidade científica, não correspondem aos fatos, a verdade pode ser concebida como algo cumulativo, portanto, sua verdade e sua falsidade podem ser marcadas mediante graus de evidências." (Abe, Costa *et al.*, 1999).

3.5.2 Reticulado “seis”

Considerando mais duas anotações, Anad, Subrahmanian e Flog propuseram um reticulado da LPA, que aumenta a precisão de controle.

Seja um reticulado finito τ = < $|\tau|$, ≤ >, onde τ = {$\perp$, V, F, Qv, Qf, T} e Quase Verdadeiro (Qv) e Quase Falso (Qf) são as duas constantes de anotações adicionais (Prado e Abe, 1998). A relação de ordem ≤ subjacente é representada pelo diagrama Hasse (Figura 3-2).

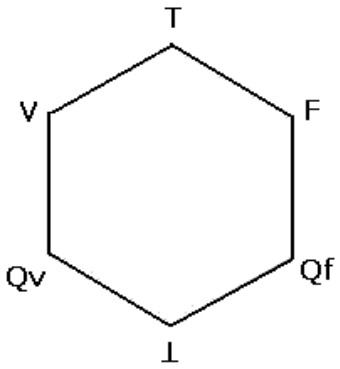


Figura 3-2. Reticulado “seis” (fonte: . DA COSTA, ABE et al. 1999).

Nesse caso, sendo:

- p ≡ "Há um obstáculo à frente" e
- grau de evidência $\mu = Q_v$, obtemos p_{Qv} = É quase verdade que exista um obstáculo à frente.

Usando esse reticulado, um sistema robótico poderia ser enriquecido com mais dois novos comportamentos, que o habilitassem ainda mais para buscar novas evidências em situações adversas. A variação no grau de evidência deixaria o comportamento do robô um pouco mais próximo do comportamento humano. Mas um grande inconveniente é que teriam de ser usados circuitos que aceitassem essa lógica multivalorada de quatro valores, ou seja, com sensores considerando os valores 0, 0,25, 0,75 e 1 (Abe, Costa *et al.*, 1999).

3.6 Lógica paraconsistente anotada Eτ

Um outro exemplo de Lógica Paraconsistente Anotada é a Lógica Paraconsistente Anotada Eτ (Prado, 1997), (Prado e Abe, 1998), (Da-Silva-Filho, 1999).

Passamos a considerar a Lógica Paraconsistente Anotada Evidencial Eτ cujo reticulado τ é o conjunto [0,1] $\times$ [0,1], onde [0,1] indica o intervalo real unitário.

O primeiro valor da anotação representa o grau de evidência favorável expressa pela proposição p; o segundo, o grau de evidência contrária, expressa pela proposição p. Na prática, o reticulado recebe como entrada os graus de evidência favorável e contrária, obtendo por intermédio deles o estado lógico correspondente.

O reticulado τ compõe-se de:

- $|\tau|$ = [0,1] x [0,1] onde [0,1] é o intervalo unitário real;
- $\leq$ = $\{((\mu_1,\rho_1),(\mu_2,\rho_2)) \in ([0,1] \times [0,1])^2 \mid \mu_1 \leq \mu_2 \text{ e } \rho_1 \leq \rho_2\}$ é a relação de ordem dos números reais. Os valores dos graus de evidência favorável μ_1 e de evidência contrária μ_2 são independentes e assumem valores reais no intervalo fechado [0,1].

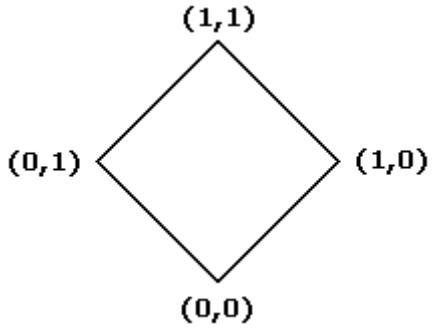


Figura 3-3. Reticulado associado à $E\tau$ (fonte: . DA COSTA, ABE et al. 1999).

Na $E\tau$, o operador ~: $|\tau| \rightarrow |\tau|$ é definido como:

$\sim [(\mu_1,\mu_2)] = (\mu_2,\mu_1)$,

onde $\mu_1,\mu_2 \in \{x \in \Re / \ 0 \leq x \leq 1\}$, considera-se (μ_1,μ_2) como uma anotação de p.

Ao associar uma anotação (μ_1, μ_2) a cada proposição p, a proposta é anotar o grau de evidência favorável μ_1 em p e o grau de evidência contrária μ_2 em p. Assim, nesse reticulado. .a anotação (1.0,0.0) indica "evidência favorável total"; (0.0,1.0), "evidência contrária total"; (1.0,1.0), "evidências totalmente inconsistentes" e (0.0,1.0), "ausência total de evidência favorável".

Seja a proposição p ≡ "A noite está estrelada", podemos variar o significado dessa preposição conforme a anotação a seguir:

Por p(1.0,0.0), " a noite está estrelada com evidência favorável total e evidência contrária nula." Em outras palavras, a noite está estrelada (verdadeiro);

- Por p(0.8,0.0), "a noite está estrelada com evidência favorável até 80% e evidência contrária nula." Em outras palavras, a noite está estrelada, mas não totalmente;

- Por p(1.0,1.0), "a noite está estrelada com evidência favorável total (ou até 100%) e evidência contrária total (ou até 100%)." Em outras palavras, as evidências são conflitantes (inconsistentes);

- Por p(0.0,0.0), "a noite está estrelada com evidência favorável nula e evidência desfavorável nula." Aqui há uma ausência total de evidências (paracompleto);

Seja uma proposição p ≡ “Há um obstáculo à frente” tratada por dois sensores infravermelhos S1 e S2 de um robô móvel autônomo. Em dado momento, o sistema poderá ter que lidar com informação inconsistente: o sensor S1 indica a existência de um obstáculo à frente (verdadeiro), enquanto o sensor S2 nada indica (falso). Segundo a $E\tau$, o valor lógico resultante da anotação seria inconsistente, o que poderia fazer o robô buscar por mais informação para dissolver essa ambigüidade.

3.7 O reticulado no plano cartesiano

O reticulado associado à $E\tau$ pode ser representado através do Quadrado Unitário do Plano Cartesiano (QUPC) real, associando ao eixo x o grau de evidência favorável e o eixo y, ao grau de evidência contrária (Figura 3-4):

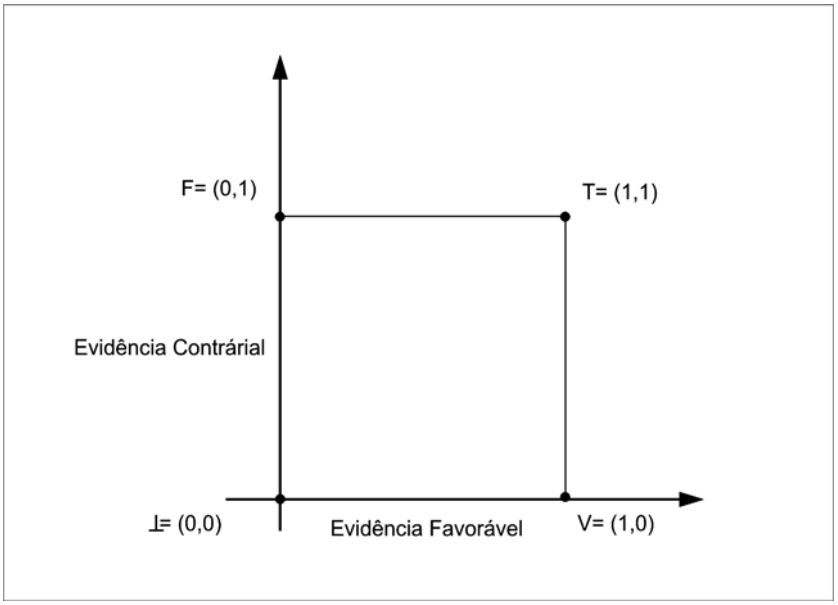


Figura 3-4. Reticulado da $E\tau$ no QUPC (fonte: DA COSTA, ABE et al. 1999).

Se os valores dos graus de evidência favorável e de evidência contrária forem binários e independentes, então os resultados obtidos no QUPC são os denominados estados extremos, a saber:

- T = (1, 1): Inconsistente;
- V = (1, 0): Verdadeiro;
- F = (0, 1): Falso;
- $\perp$ = (0, 0): Paracompleto.

Valendo-se de uma tabela-verdade, verifica-se que, ao aplicar o operador ~, que denominaremos de operador NOT, a inversão entre os valores das anotações μ_1 e μ_2 de estados extremos resulta em uma negação dos estados lógicos resultantes:

(μ_1,μ_2)		$\sim[(\mu_1,\mu_2)] = (\mu_2,\mu_1)$		Estados resultantes
μ_1	μ_2	μ_2	μ_1	
1	1	1	1	T
1	0	0	1	F
0	1	1	0	V
0	0	0	0	$\perp$

Tabela 3-1. Tabela verdade com operador NOT (fonte: DA COSTA, ABE et al.1999).

Na Lógica Paraconsistente Anotada Eτ, a aplicação dos conectivos OR e AND entre dois sinais anotados A e B resultará em uma maximização (OR) e em uma minimização (AND), como na Lógica Clássica. Nesses casos, obteremos graus de evidência favorável e evidência contrária resultante (μ_{1R},μ_{2R}):

- $V \vee T = (1,0) \vee (1,1) = (1 \vee 1, 0 \vee 1) = (1,1) = T$;
- $F \wedge T = (0,1) \wedge (1,1) = (0 \wedge 1, 1 \wedge 1) = (0,1) = F$, e assim por diante.

Sobre esses três operadores, nota-se que:

1. Na aplicação do operador OR em dois sinais anotados contraditórios, ou seja, quando um for verdadeiro e outro falso, o estado lógico resultante é inconsistente: $V \vee F = F \vee V = T$;

2. Na aplicação do operador AND em dois sinais anotados contraditórios, obtém-se paracompleto: $V \wedge F = F \wedge V = \perp$.

Em geral, os sinais dos graus de evidência favorável e evidência contrária podem receber mais de dois valores, enquanto continuam sendo independentes entre si. Para cada valor adicional interpolado no QUPC, tem-se um acréscimo de novos pontos (estados) no reticulado. Os estados extremos já vistos continuarão a ser considerados, mas os novos estados lógicos serão estados resultantes não extremos. Cada estado não extremo receberá uma denominação conforme sua proximidade com os estados extremos no QUPC.

No caso de acrescentar mais um sinal de valor 0,5 a ambos os graus de evidência favorável e evidência contrária, isto é, nos eixos x e y, podemos ter os seguintes estados não extremos (Figura 3-5):

- $V \rightarrow T$ = (1.0,0.5): Verdadeiro, tendendo ao inconsistente;
- $V \rightarrow \perp$ = (0.5,0.0): Verdadeiro, tendendo ao paracompleto;
- $F \rightarrow T$ = (0.5,1.0): Falso, tendendo ao inconsistente;
- $F \rightarrow \perp$ = (0.0,0.5): Falso, tendendo ao paracompleto;
- Qv = (0.5,0.5): Quase verdadeiro.

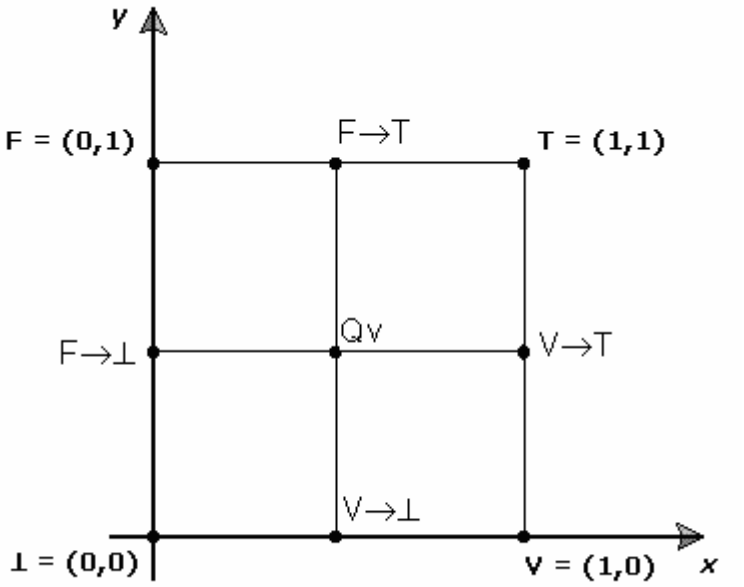


Figura 3-5. Valores ternários e independentes dos graus de evidência favorável e evidência contrária geram cinco estados lógicos não extremos (fonte: DA COSTA, ABE et al.1999).

Com o acréscimo de novos valores, as operações com NOT, OR e AND seguirão os mesmos procedimentos usados para os sinais binários e independentes.

Mas aumentando-se o número de valores dos graus de evidência favorável e evidência contrária, haverá um aumento muito grande de pontos interpolados dentro do QUPC. Como cada ponto interpolado corresponde a um estado lógico de saída, esse aumento de pontos situados dentro do QUPC traz como conseqüência um número muito grande de estados lógicos resultantes (Figuras 3-6 e 3-7).

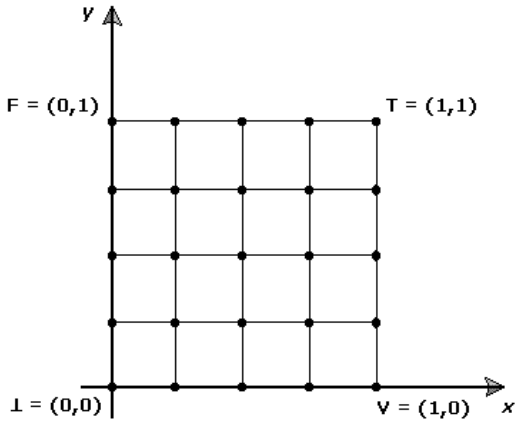


Figura 3-6. Reticulado com cinco graus de evidências (fonte: DA COSTA, ABE et al.1999).

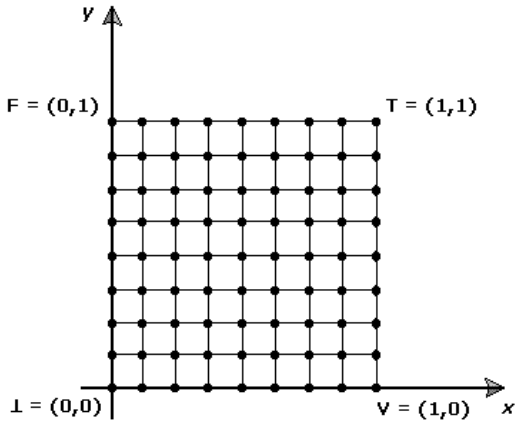


Figura 3-7. Reticulado com nove graus de evidências (fonte: DA COSTA, ABE et al.1999).

Na prática, o número de estados lógicos resultantes de saída desejada é dependente da aplicação e, para estabelecer esse método em sistemas de controle, deve ser feita uma otimização.

Na otimização consideram-se os pontos situados muito próximos como os de um mesmo estado lógico resultante. Dessa forma, são obtidas regiões com certa delimitação, onde todos os pontos considerados são inseridos, formando um conjunto de pontos equivalente a um único estado lógico resultante de saída (Figura 3-8).

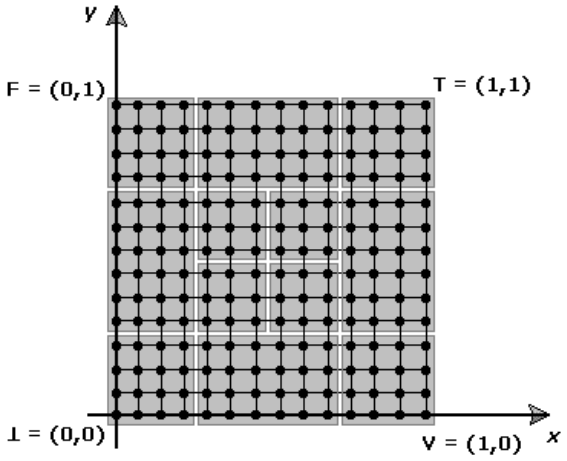


Figura 3-8. Reticulado otimizado (fonte: DA COSTA, ABE et al.1999).

A resolução do processo vai depender do número de regiões delimitadas. A precisão da análise também é dependente da discretização que será efetuada nos sinais de entrada, compostos pelos graus de evidência favorável e evidência contrária que estão sendo aplicados.

A aplicação da Lógica Paraconsistente Anotada $E\tau$ com sinais de valores contínuos e de intensidades variáveis com o tempo ou analógicos permite que esses sinais possam ser, conforme as necessidades dos projetos de aplicação, discretizados ou multivalorados por meio de circuitos eletrônicos.

A transformação dos sinais de continuo em discreto, bem como o aumento ou a diminuição da discretização nos valores analógicos dos graus de evidência favorável e evidência contrária podem ser feitos de forma computacional ou por meio de circuitos eletrônicos.

É conveniente ressaltar que diferentes reticulados podem ser empregados, permitindo que o projetista utilize o mais adequado na sua aplicação.

4 *Hardware* Reconfigurável

4.1 ***Hardware*** **Reconfigurável**

Levando-se em consideração as soluções tecnológicas desenvolvidas nos últimos anos, podemos perceber um avanço significativo tanto na área da computação como na área das engenharias.

Inicialmente, tínhamos basicamente soluções implementadas em circuitos digitais de aplicações específicas (ASICs), também conhecido como soluções de *hardware* fixo, que apesar de resolver muitos problemas, deixam a desejar quando se discute questões do tipo simplicidade de desenvolvimento, flexibilidade da arquitetura e custos, pois, para cada aplicação terá que ser construída uma máscara específica para sua produção.

A partir daí, começaram a surgir os dispositivos programáveis denominados CLPs (Controladores lógicos programáveis), também denominado *Hardware* Programável, onde temos a possibilidade de definir o comportamento do dispositivo por meio de um *software*, tornando-o, assim, mais flexível; porém, com algumas limitações de desempenho.

Podemos dividir os CLPs de acordo com as tecnologias: MPGAs (*Mask Programmable Gate Arrays*), *Standard Cells*, PLDs (*Programmable Logic Devices)*, SPLDs (*Simple Programmable Logic Devices*), CPLDs (*Complex Programmable Logic Devices*) e FPGAs (*Field Programmable Gate Array*).

- MPGA é um tipo de implementação caracterizado por possuir um processo de fabricação semicustomizado por máscaras genéricas de módulos pré-projetados, mas ainda necessita de máscaras específicas para a interconexão dos módulos. Existe também a possibilidade de se utilizar uma biblioteca de células, diminuindo muito o tempo de desenvolvimento e conseqüentemente o custo, se comparado aos dispositivos ASICs.

- *Standard Cells* é uma tecnologia muito semelhante à MPGAs, com a diferença de armazenar a biblioteca de células em banco de dados.

- Nos PLDs, temos como característica principal a possibilidade de configuração do dispositivo após sua fabricação. A configuração (também denominada programação) pode ser feita pelo usuário, proporcionando mudanças de projeto. Pelo fato dos PLDs possuírem um padrão de fabricação, o seu custo é muito baixo, o que o torna uma tecnologia muito interessante comercialmente.

Recentemente, nos deparamos com uma tecnologia em que a flexibilidade do projeto passa a estar presente no *hardware* e não mais no *Software* como era, por exemplo, no *Hardware* Programável; a esta nova tecnologia, deu-se o nome de *Hardware* Reconfigurável e, em termos básicos, podemos dizer que ela combina a velocidade do *hardware* com a flexibilidade do *software* (Ribeiro, 2002), além de simplificar muito a implementação do projeto de *hardware*.

O enquadramento das tecnologias reconfiguráveis aos outros dispositivos existentes está ilustrado no diagrama a seguir:

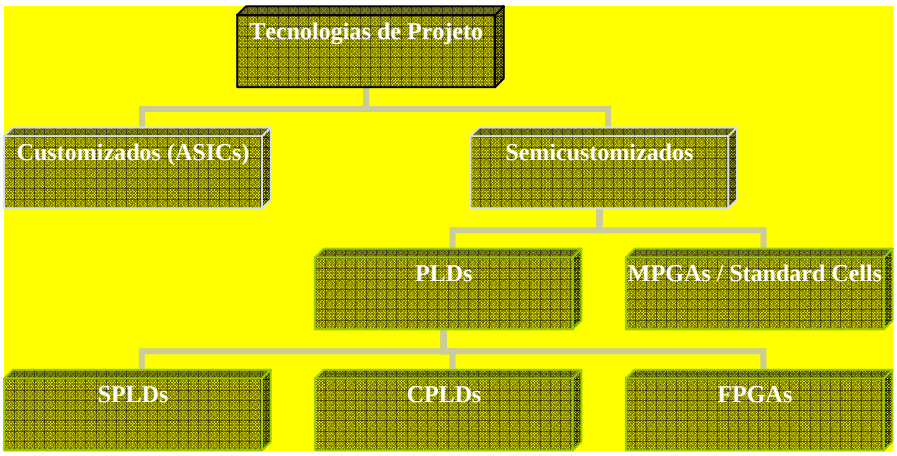


Figura 4-1 - Reconfigurabilidade dinâmica e remota de fpgas (fonte: Ribeiro 2002)

É importante ressaltar que a escolha da tecnologia a ser aplicada em cada projeto deve ser escolhida levando em consideração os recursos necessários a cada necessidade, não existindo, assim, uma única tecnologia adequada para todas as aplicações.

Os sistemas de computação reconfigurável são plataformas onde sua arquitetura pode ser modificada em tempo real para melhor se adequar à aplicação que está sendo executada. Deste modo, o processador reconfigurável passa a trabalhar com uma arquitetura desenvolvida exclusivamente para determinado tipo de aplicação, permitindo uma eficiência muito maior do que a normalmente encontrada em processadores de uso geral.

A computação reconfigurável tem por objetivo suprir a lacuna entre a solução por *software* e a solução por *hardware*, atingindo desempenhos superiores aos obtidos por *software*, mas com flexibilidade maior que a

oferecida por uma solução por *hardware* (Chen et al.,2000; Compton, 2002; Coric et al., 2002; apud Rosário, 2005).

Alguns problemas atuais enfrentados nos sistemas micro processados, incluindo, por exemplo, questões em Robótica, é proveniente da necessidade de processamento paralelo, alta velocidade no processamento das informações dos sensores, necessidade de adequação física e lógica de acordo com diferentes cenários e logicamente manter a arquitetura dentro de um custo viável ao projeto.

Considerando que as principais vantagens do *Hardware* Reconfigurável estão no paralelismo, velocidade (Aporntewan e Chongstitvatana, 2001), capacidade de atualização, padronização de plataformas, amortização do custo de desenvolvimento e alto desempenho, fica evidente os benefícios que esta tecnologia emergente poderá proporcionar aos futuros projetos tecnológicos.

Além disso, com a evolução das metodologias de projeto de *hardware* o ciclo de projeto é reduzido de forma significativa (Aragão, 1998)

4.2 Uso atual

Podemos citar aplicações da computação reconfigurável em diversas áreas, onde em cada uma delas encontram-se aplicações variadas como controle, comunicação, manipulação matemática, criptografia, etc..

A seguir apresentam-se alguns projetos envolvendo HR:

1 - Pesquisa aeroespacial:

- Aplicação de uma FPGA no projeto do veículo Pathfinder, enviado ao planeta Marte pela NASA (Woerner & Lehman, 1995 apud Rosário, 2005).

2 - Acionamento e controle de sistemas mecatrônicos:

- Desenvolvimento de um sistema de controle de inversores de potência que utiliza FPGA (De Pablo et al., 1997 apud Rosário, 2005).

- Aplicação de uma FPGA na implantação do controle de um motor de indução (Empringham et al., 2000 apud Rosário, 2005).

- Adoção de um controlador de motor de indução usando linguagem VHDL (Cirstea et al., 2000, 2001 apud Rosário, 2005).

- Modelagem, simulação e instalação de um controlador usando lógica reconfigurável (Wegrzyn et al., 1998 apud Rosário, 2005).

-Um mini robô móvel seguidor de pistas guiado por visão local (Costa, Gomes *et al.*, 2003).

3 - Automação da manufatura:

- Técnicas para estabelecer controladores em paralelo usando redes de Petri e com aplicação em dispositivos reconfiguráveis (Kozlowski et al., 1995 apud Rosário, 2005).

- Sistema de análise de tolerância a falhas baseado em ferramentas desenvolvidas em VHDL (Baraza, 2002).

4.3 *Hardware*

Uma arquitetura FPGA é caracterizada por um arranjo de células configuráveis denominadas de Blocos Lógicos que estão dentro de um único *chip* e têm a função de estabelecer funções lógicas.

Tais células estão dispostas em linhas e colunas, formando uma matriz bidimensional, e entre as células temos chaves de interconexão que têm a função de interconectar os blocos de acordo com cada projeto que poderá ser instalado. Os blocos que estão na extremidade têm a função de prover recursos de entrada e saída.

A arquitetura de um FPGA é semelhante à arquitetura convencional de um MPGA, com a diferença de que a interconexão do MPGA é feita durante o processo de sua fabricação e as interconexões do FPGA são estabelecidas e restabelecidas pelo usuário de acordo com o projeto que ele está executando (Ribeiro, 2002).

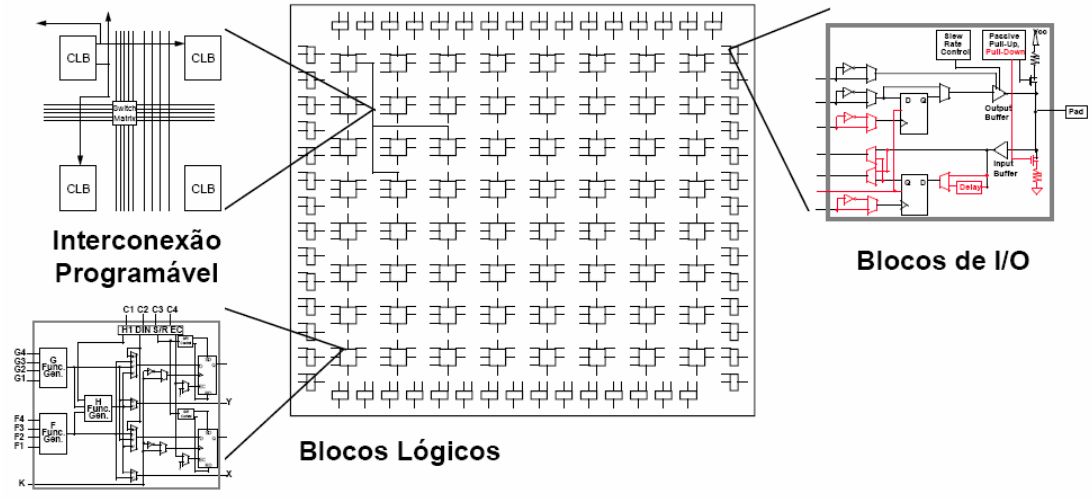


Figura 4-2 – Estrutura simplificada de um FPGA (fonte: Aragão, 1998).

Existem diversas formas para se trabalhar os Blocos Lógicos de uma FPGA, como, por exemplo, pares de transistores, portas básicas do tipo NAND ou XOR de duas entradas, multiplexadores, estruturas AND e OR de múltiplas entradas e *flip-flops* associados com diversos tipos de portas lógicas podem ser utilizados também (Ribeiro, 2002); no caso do fabricante Altera Corp., é utilizado o bloco de memória LUT (*Look-Up Table*) (Costa, 2006).

O tipo de bloco formado com LUT contém células de armazenamento binário que são utilizadas para habilitar funções lógicas. Uma LUT de quatro ou cinco entradas que podem endereçar 16 ou 32 células de armazenamento.

Para a instalação de um circuito lógico em uma FPGA devem-se programar os blocos lógicos para executarem as funções necessárias e também as chaves de roteamento para que os blocos sejam interligados a fim de executar uma determinada tarefa.

O conteúdo armazenado nos LUTs são perdidos sempre que o dispositivo deixar de receber energia. Para resolver este problema de volatilidade, normalmente, utiliza-se uma memória EEPROM (*Electrically Erasable Programmable Read Only Memory*) para carregar os LUTs sempre que necessário.

Existem outras duas características que são de extrema relevância na constituição dos FPGAs; são elas: Granularidade e Arquitetura de Roteamento.

Granularidade:

A menor unidade configurável presente na arquitetura do FPGA é denominada de grão. Existem três tipos de grão que podem ser utilizados nessas estruturas e cada um dos tipos possui as seguintes características:

- Grão pequeno: com um grande número de blocos lógicos simples, eles normalmente habilitam uma função lógica de duas entradas ou um multiplexador 4×1 e um *flip-flop*.

- Grão médio: a maioria das arquiteturas de grão médio estabelecem as funções lógicas em LUTs de quatro entradas. Ele é composto normalmente por duas ou mais LUTs e dois ou mais flip-flops.

- Grão grande: as FPGAs de grão grande possuem, normalmente, como grão ULAs (Unidades lógicas e aritméticas), microprocessadores e memórias.

Arquitetura de Roteamento:

Quem define como as células lógicas serão interligadas é a Arquitetura de Roteamento.

Podemos dividir as tecnologias de roteamento em três: Antifuse, Gate flutuante e SRAM (*Static Random Access Memory*).

A seguir será apresentada uma breve definição de cada uma dessas tecnologias.

- Antifuse: Está presente nesta tecnologia um dispositivo com dois terminais que, quando programado, apresenta um caminho de baixa

impedância entre seus terminais, permitindo assim que o circuito seja fechado. Sem programação, a alta impedância entre os terminais faz com que o circuito fique aberto.

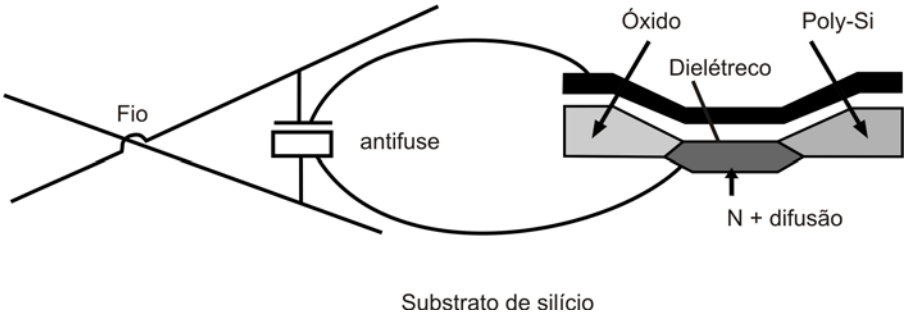


Figura 4-3 – Antifuse (fonte: Aragão 1998)

- Gate flutuante: Construído com a tecnologia MOS (*Metal Oxide Semiconductor*), é a mesma tecnologia presente nas memórias EPROM (*Erasable Programmable Read Only Memory*) e EEPROM; a principal vantagem desta tecnologia está na capacidade de programação e retenção dos dados. Uma outra vantagem é a possibilidade de efetuar a programação ISP (In System Programmability).

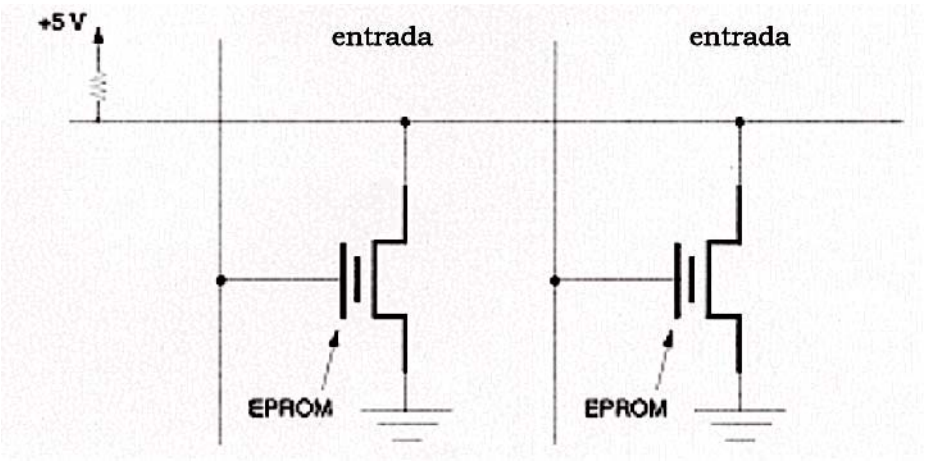


Figura 4-4 – Tecnologia de programação Gate flutuante (fonte: Costa,2006).

- SRAM: Neste caso utilizam-se células de memória SRAM para controlar transistores de passagem ou multiplexadores.

Como a memória SRAM é volátil, ela precisa ser configurada sempre que o *chip* for ligado. Esta configuração pode ficar armazenada num *chip* de memória PROM (*Programmable Read Only Memory*), EPROM ou ainda EEPROM.

Como para cada comutador estão associados pelo menos seis transistores, essa tecnologia tem a desvantagem de ocupar muito espaço, porém, ela pode ser rapidamente reprogramável e sua fabricação requer apenas a tecnologia padrão de circuitos integrados.

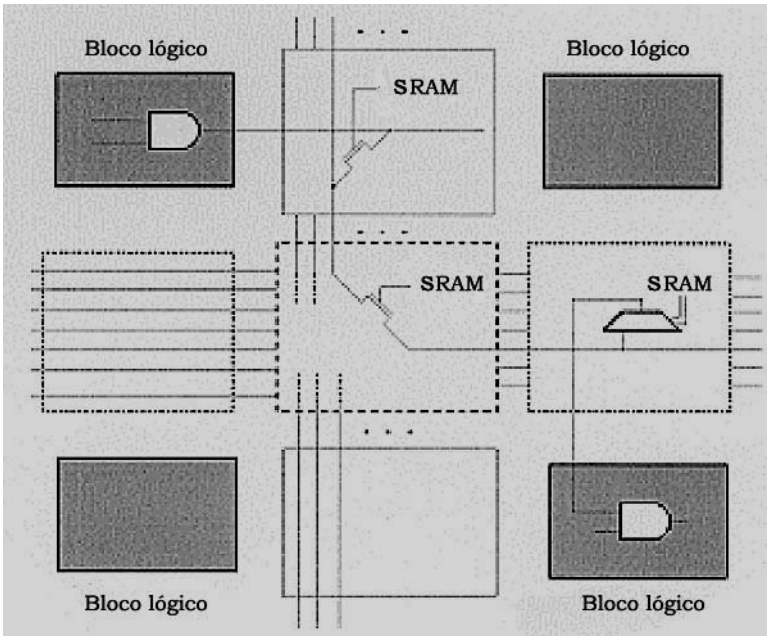


Figura 4-5– Tecnologia de programação SRAM (fonte: Costa,2006).

4.4 Reconfigurabilidade de FPGAs.

A reconfigurabilidade dos FPGAs pode ser exemplificada conforme a figura abaixo:

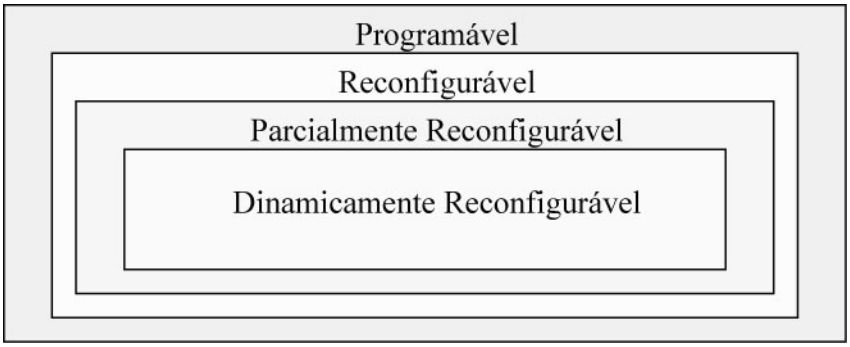


Figura 4-6 – Exemplos de reconfigurabilidade de FPGAs.

Sendo um *hardware* programável e reconfigurável, pode-se ainda classificá-lo como Parcialmente Reconfigurável ou Dinamicamente Reconfigurável.

- Parcialmente Reconfigurável: uma parte do dispositivo é reconfigurado enquanto a outra parte permanece inativa, porém a parte que não será reconfigurada permanece intacta sem nenhuma modificação;

- Dinamicamente Reconfigurável: os circuitos internos de armazenamento podem ser atualizados seletivamente sem prejudicar o funcionamento da lógica do restante que pode estar em execução.

A principal questão nesta abordagem é a implantação de um controlador capaz de manipular a reconfiguração de todas essas tarefas. Este controlador deve oferecer serviços como o carregamento e remoção de tarefas, escalonamento de tarefas e gerenciamento dos recursos (Scatena, 2003).

FPGAs disponíveis no mercado:

A Altera possui as famílias de FPGAs: Stratix, Apex (Apex II, Apex 20k), Excalibur, Flex (Flex 10k, Flex 8000, Flex 6000), Mercury, Acex 1k e as famílias de CPLDs: Max 7000, Max 3000A dentre outras. Já a família Apex é a mais moderna, é composta por dispositivos de grão grosso que são feitos para servir como soluções SoPC (*System on a programmable chip*) complexas.

A Xilinx possui as famílias de FPGAs: Virtex (Virtex, Virtex II, Virtex II Pro), Spartan (Spartan XL, Spartan II, Spartan IIE) e as famílias de CPLDs: CoolRunner, CoolRunner II, XC 9500 como principais, ela é a fabricante mais tradicional no ramo de FPGAs. A família Virtex foi a primeira linha de FPGAs a oferecer um milhão de portas lógicas. Esta família redefiniu a lógica programável expandindo as capacidades tradicionais do FPGA para incluir um poderoso conjunto de características que endereçam problemas no nível da placa para projeto de sistemas de alto desempenho. Hoje a família Virtex possui dispositivos com mais de três milhões de portas lógicas e oferecem um *chip* de memória adicional para aplicações de roteamento de redes (*network switch applications*).

A Atmel possui as famílias de FPGAs de propósito geral, de dispositivos *Embedded Programmable Gate Array* (EPGA) que são dispositivos com núcleo FPGA embutido (EPGA™) e de FPGAs Antifuse. As técnicas ou tecnologias de programação ou configuração podem ser Antifuse, Flash e SRAM (*Static Random Access Memory*).

4.5 Programação de FPGAs

A programação de uma FPGA é composta por várias etapas que, normalmente, são automatizadas.

O uso de ferramentas denominadas EDA (*Eletronic Design Automation*), simplifica e acelera o ciclo de desenvolvimento do projeto.

O EDA é constituído por vários programas interconectados que executam as seguintes etapas: especificação e entrada do projeto, síntese e mapeamento da tecnologia, posicionamento e roteamento, verificação e teste e por fim a programação do FPGA.

Podemos ilustrar as etapas com o seguinte diagrama:

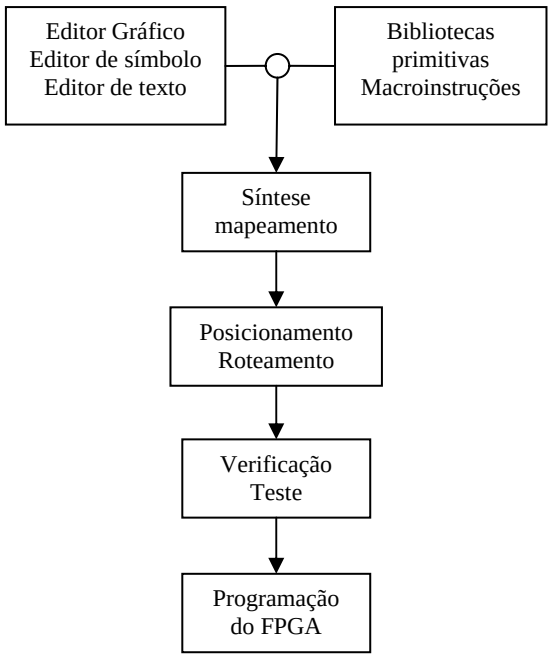


Figura 4-7 – Etapas do EDA (fonte: Costa, 2006)

Especificação e entrada do projeto.

Durante esta fase é que ocorre a programação utilizando técnicas de alto nível como diagramas lógicos ou uma linguagem de programação HDL (*Hardware Description Language*).

Os diagramas são montados a partir de um *software*, como por exemplo, o Quartus II empregado pela empresa Altera Corp. utilizando-se portas lógicas e ou bibliotecas de componentes como microcontroladores, contadores, *flip-flops* etc..

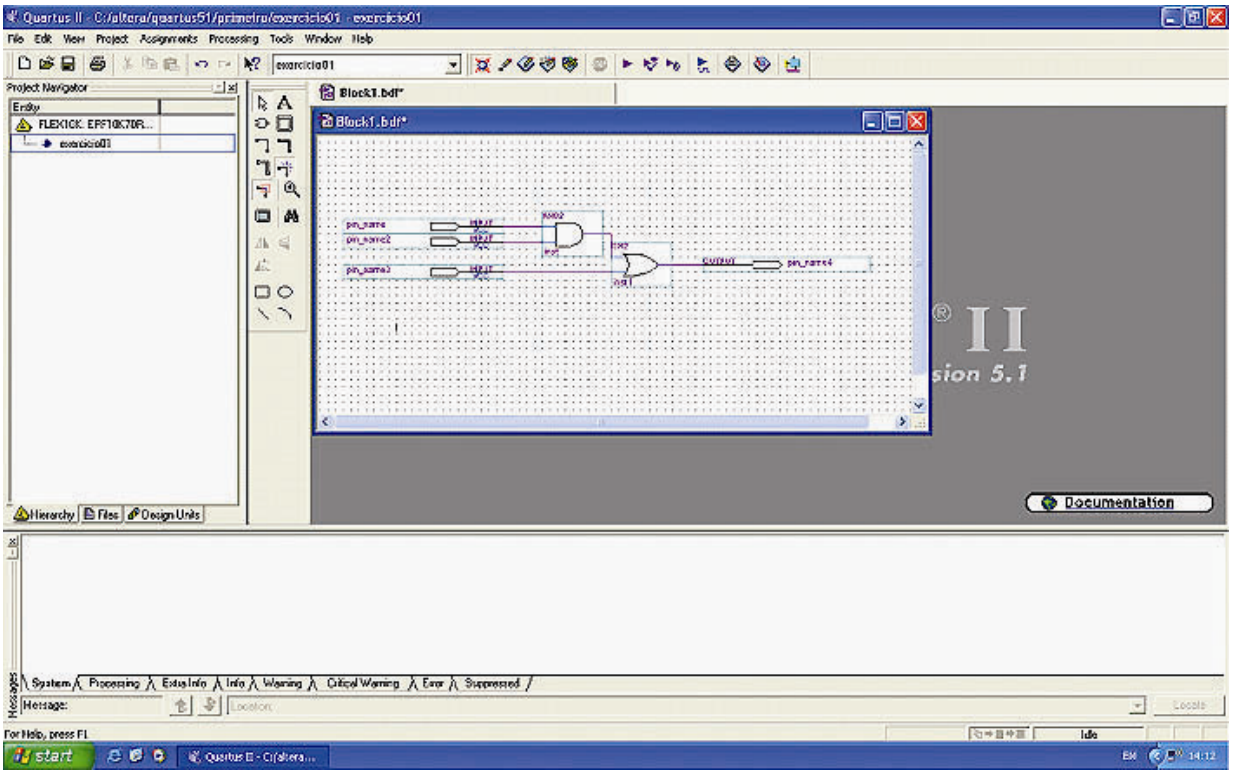


Figura 4-8 – Editor gráfico do Quartus II

Conforme a complexidade do projeto vai aumentando, a descrição por meio de diagramas vai se tornando cada vez mais complexa e inviável; quando isso acontece, a linguagem HDL se torna mais adequada.

Existem diversos compiladores para a linguagem HDL das quais podemos citar: ABEL (*Advanced Boolean Equation Language*), VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) e Verilog.

As programações mais complexas utilizam, normalmente, as linguagens VHDL ou Verilog, que são mais robustas e possuem maiores recursos.

A linguagem HDL pode descrever o comportamento do sistema por meio de equações lógicas, tabela verdade e diagrama de formas de onda (Costa, 2006).

Síntese lógica e mapeamento da tecnologia:

Nesta etapa do desenvolvimento, ocorre a simplificação das equações booleanas descritas na fase da especificação do projeto, para que o mesmo fique otimizado. Ao mesmo tempo, ocorre também o mapeamento dos recursos do *hardware* para a posterior implementação do projeto.

Posicionamento e roteamento:

O posicionamento consiste em relacionar os componentes lógicos aos componentes físicos disponíveis no *hardware*.

Além do posicionamento, é necessário também fazer a alocação das trilhas e dos elementos programáveis, para que os blocos de interconexão possam ligar os componentes de forma correta. A este processo da-se o nome de roteamento.

Verificação e teste:

A verificação e os testes são feitos através de simulações que podem ser definidos a partir do nível funcional, comportamental ou no nível das portas lógicas.

Desta forma podemos detectar restrições de temporização e falhas na definição do projeto antes mesmo da programação do FPGA.

Após todas as etapas anteriores serem concluídas com êxito, um arquivo de configuração é gerado para que o dispositivo possa ser fisicamente programado.

Este arquivo poderá ser transferido para o FPGA de diversas formas, como por exemplo, programação serial passiva (por meio de um cabo conectado a porta paralela) ou pela interface JTAG.

Para finalizar, não é difícil de prever que em um futuro muito próximo, as máquinas irão estar conectadas a grandes redes, como por exemplo, a Internet, e a partir daí se comportar como os programas que utilizamos hoje em dia, onde novas versões poderão ser implementadas automaticamente em sistemas de *Hardware* Reconfigurável.

5. Implementação

5.1 Considerações iniciais

Como foi discutido anteriormente, os robôs são inerentemente imprecisos quanto ao estado de seu ambiente. Essa imprecisão é conseqüência das limitações dos sensores e da impossibilidade de se prever a maioria dos ambientes onde o robô irá atuar. Na hora de obter e tratar dados de seus sensores, robôs paraconsistentes computam crenças sobre o que deve estar acontecendo à sua volta ao invés de gerar apenas um único resultado determinístico. Desta forma, eles podem se recuperar de maneira mais suave de possíveis erros, além de contar com mecanismos que permitem uma melhor integração dos dados obtidos dos vários sensores, podendo lidar com as ambigüidades oriundas dessa integração.

Qual é a vantagem de se usar robôs paraconsistentes? A vantagem é que um robô que carrega uma noção de suas incertezas (de sua ignorância) e age de acordo obterá melhores resultados do que um que não reconhece suas próprias limitações.

Dentro do estudo de mecanismos de navegação para robôs móveis, as duas principais áreas onde a Lógica Paraconsistente Anotada pode e deve ser utilizada são:

- percepção e
- controle.

A percepção sobre o ambiente é considerada como problema fundamental para se obter um robô móvel com capacidades autônomas.

Dentro desse problema, é particularmente complexa a tarefa da auto localização, nesta o robô deve determinar sua localização baseando-se na percepção e no reconhecimento de características específicas do ambiente.

Neste sentido a Lógica Paraconsistente Anotada é uma poderosa ferramenta para modelar a percepção do mundo, permitindo o desenvolvimento de um sistema de localização capaz de estimar a posição do robô mesmo em situações de falta ou inconsistência de dados.

O principal objetivo da robótica é construir robôs que ajam corretamente. Cabe ao mecanismo de controle analisar as informações obtidas pelo processo de percepção e determinar quais ações o robô deve executar. Como dito anteriormente, um robô que tem noção das suas incertezas e as leva em consideração na hora de tomar suas decisões tende a tomar decisões melhores.

É importante considerar que a inconsistência não é característica apenas dos dados, mas uma propriedade do robô, mais precisamente, do estado interno dele. Um robô pode estar incerto sobre a existência de um dado objeto, o valor de um atributo, a verdade de uma proposição ou o efeito da execução de uma ação.

A Lógica Paraconsistente Anotada fornece mecanismos e técnicas para representar e manipular o fenômeno da inconsistência. Permitindo o desenvolvimento de um sistema de navegação onde a inconsistência não é simplesmente eliminada dos dados coletados, por um artifício, mas sim incorporada ao modelo do mundo.

Nas últimas décadas houve o surgimento de diversos paradigmas para sistemas de navegação de robôs. Os primeiros trabalhos assumiam freqüentemente a existência de um modelo correto e completo do ambiente onde o robô iria atuar, as ações deste eram determinadas por sistemas de planejamento. A grande dificuldade, ou até a impossibilidade, de se gerar modelos de representação do mundo corretos e completos dentro das restrições de tempo que a aplicação impõe, levou diversos projetistas a adotarem abordagens reativas onde não são utilizados tais modelos. Um importante sistema que segue essa abordagem é descrito em [Brooks 91], em que o comportamento do robô é determinado por um autômato finito que mapeia as entradas dos sensores em ações, não havendo uma modelagem do mundo. Essa abordagem, contudo, limita as atividades que podem ser desenvolvidas pelos robôs, uma vez que seu conhecimento sobre o mundo está restrito ao alcance de seus sensores.

Um robô construído segundo uma abordagem baseada em Lógica Paraconsistente Anotada está entre esses dois extremos. Nessa abordagem existe um modelo do mundo em que sistemas de planejamento podem atuar, porém tal modelo não necessita ser totalmente correto e completo, pois os mecanismos de anotação e manipulação das inconsistências permitem que sistemas de planejamento atuem em um modelo aproximado do mundo real - podendo haver inconsistência nesta representação - construído dentro dos limites de tempo impostos pela aplicação.

O robô móvel Luiza é implementado neste trabalho com a finalidade de ilustrar o uso da Lógica Paraconsistente Anotada $E\tau$, sobre a arquitetura

emergente denominada Hardware Reconfigurável, para resolver problemas relacionados à navegação do robô.

Sinais, muitas vezes conflitantes, provenientes dos sensores utilizados para a navegação do robô, caracterizam um problema complexo para ser resolvido utilizando a Lógica Clássica, por esse motivo a implementação deverá contemplar o tratamento desses sinais sobre o paradigma da Lógica Paraconsistente Anotada Eτ.

O robô deverá mover-se em direção à fonte de luz mais intensa, ao mesmo tempo em que desvia de obstáculos existentes em seu caminho. Caso a intensidade das luzes ao seu redor seja homogenia, o mesmo deverá permanecer parado.

5.2 Metodologia

O robô está dividido em camadas, a fim de possibilitar alterações entre as camadas da sua constituição para estudos de desempenho.

As camadas estão dispostas dentro e fora do *Hardware* Reconfigurável da seguinte forma:

Quadro 5-1 – Camadas do robô Luiza

CAMADA 4	Sensores e tratamento de sinais.
CAMADA 3	Tratamento das informações paraconsistentes e definição de um comportamento.
CAMADA 2	Implementação do comportamento
CAMADA 1	Atuadores

Camada 4:

Constituem a camada dos sensores, dois sonares paraconsistentes construídos a partir de quatro sonares independentes e oito sensores de luz do tipo LDRs. Ambos estão conectados a um microcontrolador ATMEGA8 que tem a função de transformar os sinais analógicos provenientes dos LDRs em sinais digitais codificados, além de gerar um código binário identificando a presença ou ausência de obstáculos nas proximidades do robô.

Esses sinais são analisados pela camada 3, que está implementada dentro do *Hardware* Reconfigurável.

Camada 3:

A camada 3 está instalada dentro do *Hardware* Reconfigurável por um circuito introduzido pelo *software* Quartus da empresa Altera e tem a função de analisar os sinais provenientes do microcontrolador, aplicar a lógica paraconsistente anotada $E\tau$ e enviar um código de acionamento de um dos comportamentos presentes na Camada 2.

Camada 2:

Também implementado dentro do *Hardware* Reconfigurável, esta camada recebe o sinal proveniente da Camada 3, verifica se o sinal de “habilita” está ativo e gera um código de acionamento dos motores, proporcionando assim que o robô execute um determinado comportamento servo ou balístico.

Camada 1:

O código binário gerado pela Camada 2 é recebido por um circuito que tem a função de *driver* de potência juntamente com uma ponte H que é capaz de inverter o sentido da corrente de alimentação dos motores de forma a configurar o seu sentido de rotação, além ligar ou desligar cada um deles de modo individual.

Os motores utilizados são de corrente contínua alimentada com uma tensão constante de 12 volts.

O diagrama das camadas do robô Luiza pode ser vista na figura 5-1

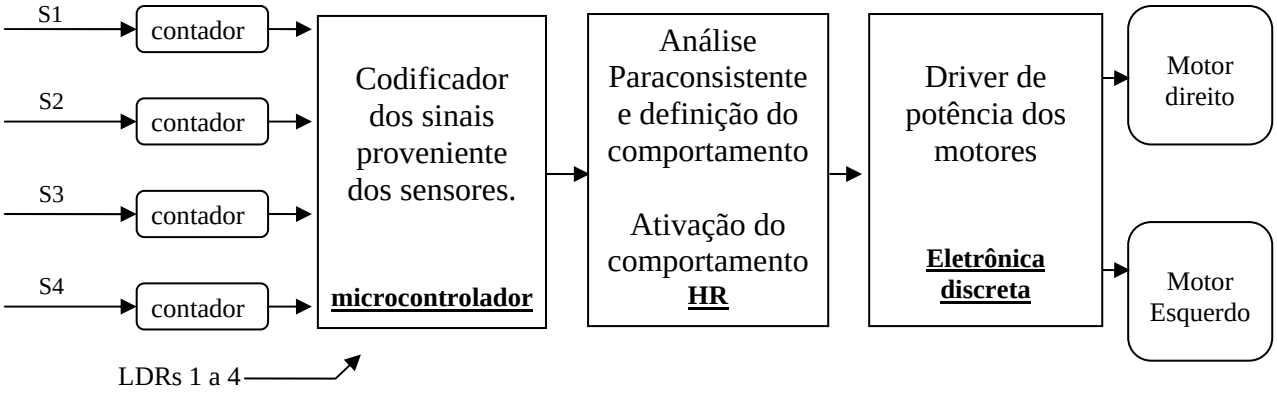


Figura 5-1 – Diagrama das camadas do robô Luiza.

Os comportamentos do robô serão ativados da seguinte forma:

Estado Confiável verdadeiro: neste caso, admite-se que a proposição "Há um obstáculo à frente" é verdadeira e, portanto, o robô deverá ativar o comportamento *scape* para se livrar do obstáculo.

Estado Confiável falso: neste caso, admite-se que a proposição "Há um obstáculo à frente" é falsa e, portanto o robô deverá executar o comportamento *home* para se mover em direção à luz.

Estado não confiável: caso o estado do robô seja “não confiável”, este deverá executar o comportamento *refresh*, em que ele executará um recuo e uma nova medida, com o objetivo de sair desse estado.

Estado Parcialmente confiável: o estado Parcialmente confiável é caracterizado quando um dos sonares paraconsistentes está no estado consistente verdadeiro e o outro no estado consistente falso. Para esta situação, o robô irá ativar o comportamento fuga, em que o desvio será feito para o lado do sonar consistente falso.

O algoritmo de funcionamento do robô segue os fluxogramas das figuras 5-2 e 5-3.

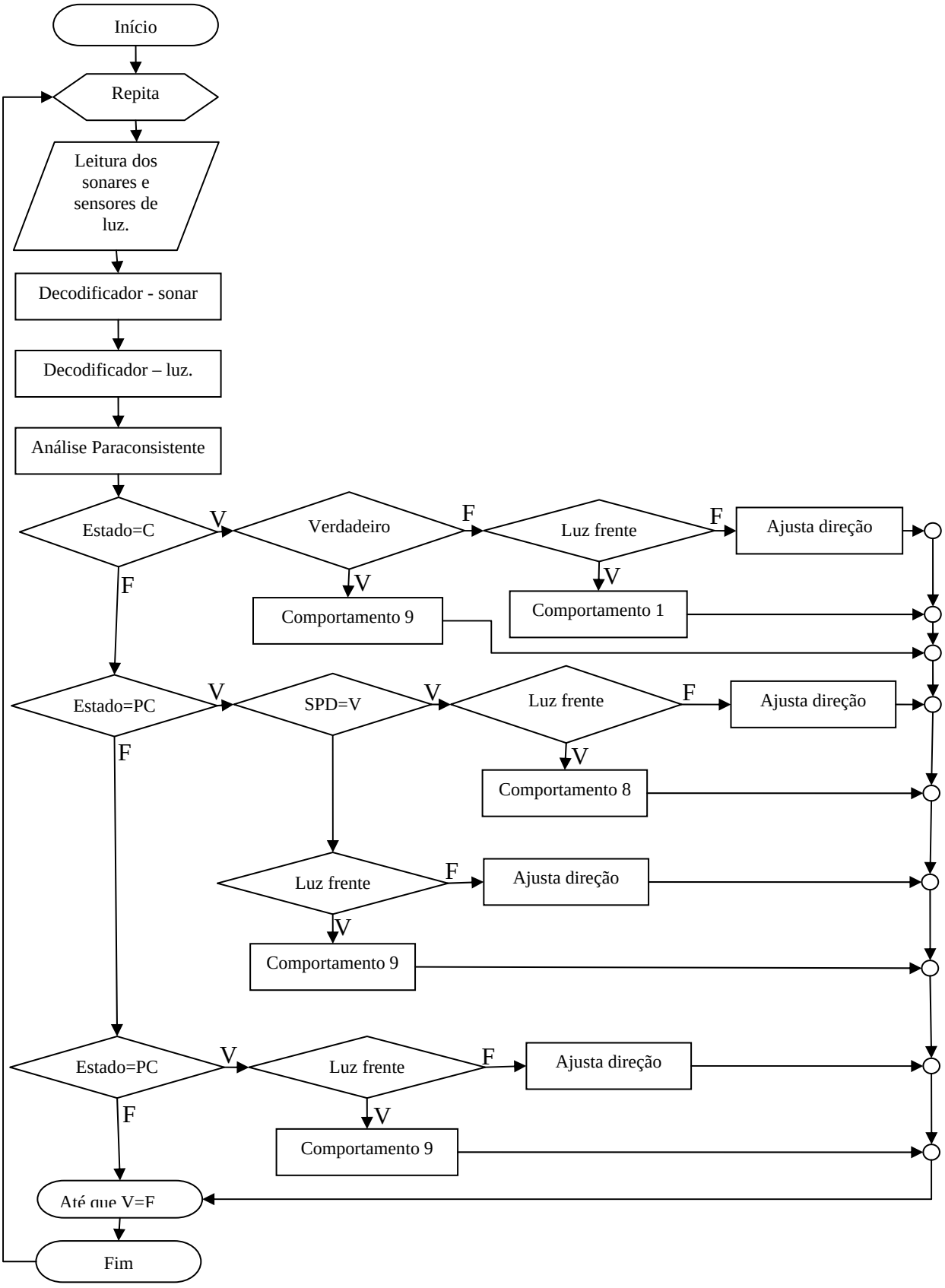


Figura 5-2 - Fluxograma do programa do robô móvel Luiza.

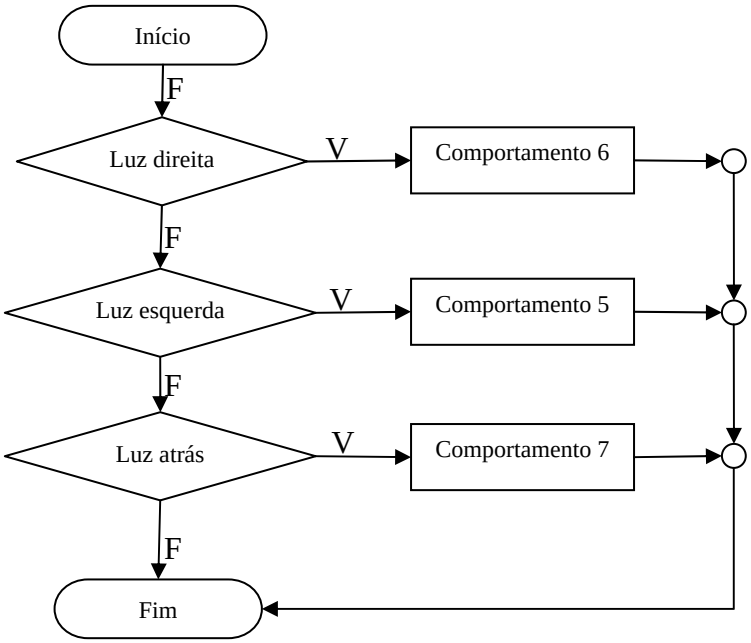


Figura 5-3 - Fluxograma do programa do robô móvel Luiza (ajuste de direção).

5.3 Construção do robô

5.3.1 Fonte de alimentação

O circuito de alimentação do robô recebe de uma fonte externa, ou de uma bateria, a tensão de 12V e fornece para o robô as tensões de 9V (para alimentação do kit UP2) e 5V para alimentação dos sonares, LDRs, microcontrolador ATMEGA8 e motores.

O intercambio entre a bateria e a fonte externa é feito por meio de um cabo com conectores de 2,5mm X 5,55mm.

5.3.3 ATMEGA8

O microcontrolador ATMEGA8 é o dispositivo que desempenha duas funções básicas no projeto do robô Luiza:

- Conversão e codificação dos sinais analógicos provenientes dos sensores de luz (LDRs) em sinais digitais (saídas A e B);

- Codificação dos sinais dos sonares de acordo com a presença ou ausência de um obstáculo próximo, ou seja, distância igual ou menor que 10 centímetros aproximadamente (saídas S1, S2, S3, S4 e S5).

Maiores detalhes sobre o microcontrolador ATMEGA8 poderão ser encontrados no (anexo XXX – Características do microcontrolador ATMEGA8).

O programa desenvolvido para tratar os sinais provenientes dos sensores foi escrito utilizando-se a linguagem de programação Basic, e o compilador Bascom.

O fluxograma está apresentado na figura (5-4).

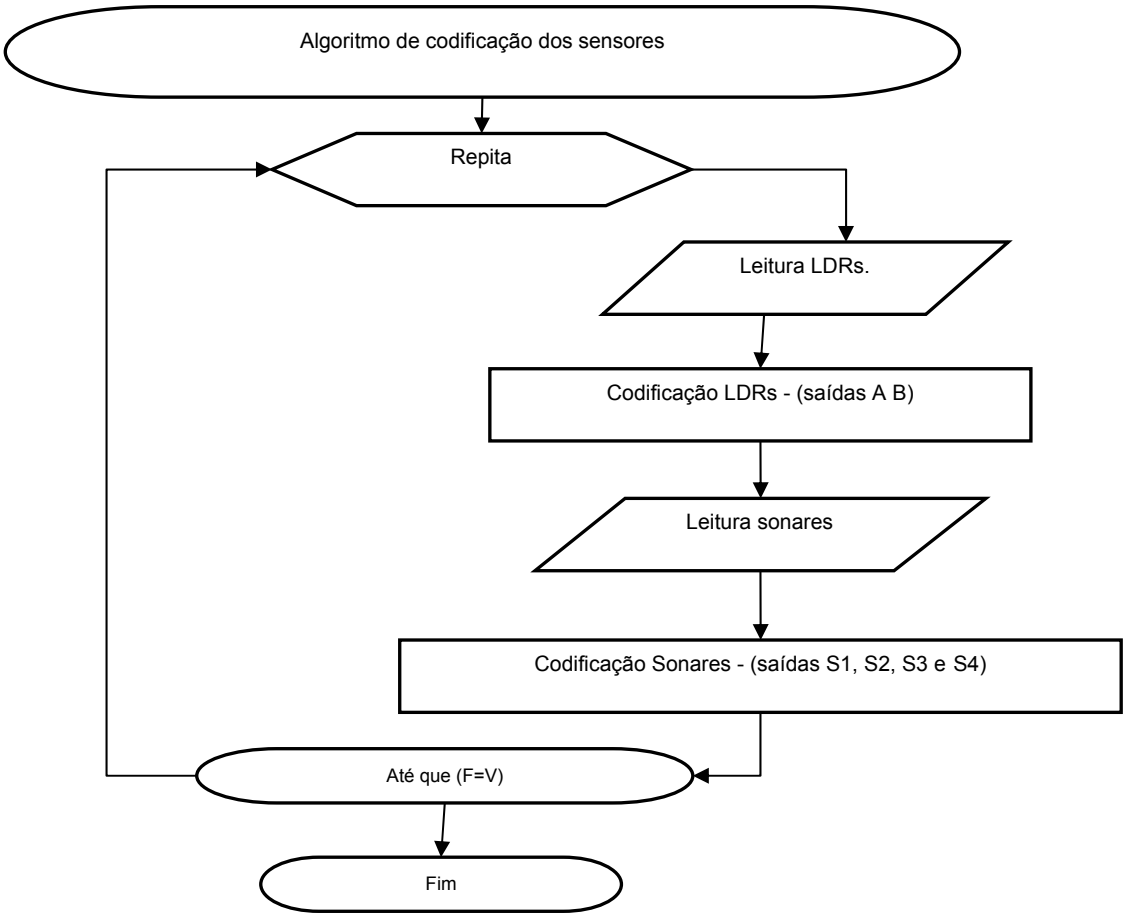


Figura 5-4 - Fluxograma do programa implementado no ATMEGA8.

5.3.3 Sonares

Os sonares foram adquiridos na indústria nacional e possuem dois conectores para alimentação (5V) e um conector de sinal, por onde um pulso é enviado de forma a detectar a distância de um obstáculo.

Quanto maior tempo do pulso, mais longe está o obstáculo, e vice-versa.

Para este trabalho utiliza-se o sonar para detectar a presença ou não de obstáculos, quando o mesmo estiver a uma distância menor que 10 centímetros aproximadamente.

A calibração dos sonares foi executada com o robô móvel parado e marcando-se no chão o alcance do sonar em relação aos objetos fixos, obtendo-se as seguintes especificações: cone de leitura = 48 graus e distância de alcance máximo = 1,20 m.

A disposição dos sonares no robô foi determinada empiricamente com um ângulo aproximado de 12 graus a fim de evitar interferências, e os pares que fazem a leitura da evidência contrária do sonar paraconsistente instalado em posição simétrica ao que faz a leitura da evidência favorável.

5.3.4 LDRs

LDR (*Light Dependent Resistor*) é um componente eletrônico que tem a função de variar sua resistência em função da intensidade da luz.

Para converter os dados analógicos dos LDRs em sinais digitais e tratá-los a fim de se obter um código binário com a indicação da origem da maior fonte de lua, utiliza-se o microcontrolador ATMEGA8 da ATMEL, que deverá prover o código diretamente para o Hardware Reconfigurável.

Os oito sensores LDRs estão ligados dois a dois em série, a fim de se obter uma melhor resolução de leitura da luz, e dispostos na região mais alta do robô em intervalos de 90 graus.

A função desses sensores é encontrar aproximadamente qual é a direção de destino do robô.

5.3.5 Hardware Reconfigurável

5.3.5.1 Kit didático Altera - UP2

O *Hardware* Reconfigurável utilizado no projeto é o FLEX 10K da empresa ALTERA, está presente no kit educacional denominado UP2 e programado pelo software QUARTUS II que é fornecido pela própria empresa proprietária da tecnologia.

O *chip* recebe o nome comercial de EPF10K70, é baseado na tecnologia SRAM, possui 3.744 elementos lógicos e encapsulamento RQFP com 240 pinos.

5.3.5.2 Circuito de análise paraconsistente e definição do comportamento

Após a modelagem do problema, iniciou-se a construção do circuito de análise paraconsistente e definição do comportamento, seguindo os seguintes passos:

- definição da tabela verdade;
- extração das expressões booleanas simplificadas;
- implementação do circuito por meio do software Quartus II no modo esquemático.

O circuito recebe do microcontrolador ATMEGA8 os sinais codificados provenientes dos sensores de luz e dos sonares da seguinte forma:

Sinais codificados provenientes dos sensores de luz (tabela 5-1):

A	**B**	**Significado**
0	0	Luz na frente do robô
0	1	Luz a direita do robô
1	0	Luz a esquerda do robô
1	1	Luz a trás do robô

Tabela 5-1 – sinais dos sensores de luz.

Sinais codificados provenientes dos sonares (tabela 5-2):

	Sinal	Sonar	Com obstáculo	Sem obstáculo
Sonar paraconsistente 1	S1	Sonar 1	1	0
	S2	Sonar 2	1	0
Sonar paraconsistente 2	S3	Sonar 3	1	0
	S4	Sonar 4	1	0

Tabela 5-2 – sinais dos sonares.

Os sinais de entrada S1, S2, S3, S4, A e B entram na tabela verdade e ativa exclusivamente um dos seguintes comportamentos (tabela 5-3):

C1	Deslocamento para frente
C2	Deslocamento para trás
C3	Curva para a esquerda
C4	Curva para a direita
C5	Giro de 45 graus para a esquerda
C6	Giro de 45 graus para a direita
C7	Giro de 180 graus
C8	Fuga pela esquerda
C9	Fuga pela direita

Tabela 5-3 - comportamentos do robô móvel Luiza.

Os comportamentos C1, C2, C3 e C4 são comportamentos servos e os comportamentos C5, C6, C7, C8 e C9 são comportamentos balísticos.

A fuga significa girar 45 graus para o lado e andar 3 centímetros para frente.

A construção da tabela-verdade leva em consideração as variáveis de entrada: S1, S2, S3, S4, A, B e as variáveis de saída: C1, C2, C3, C4, C5, C6, C7, C8 e C9. Os conceitos da Lógica Paraconsistente Anotada $E\tau$ e dos comportamentos de robôs foram considerados para a definição do comportamento a ser ativado para cada situação de entrada. (tabela 5-4).

	S1	S2	S3	S4	A	B	C1	C2	C3	C4	C5	C6	C7	C8	C9
1	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0	**1**	0	0	0
3	0	0	0	0	1	0	0	0	0	0	**1**	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0	0	0	**1**	0	0
5	0	0	0	1	0	0	0	**1**	0	0	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0	0	**1**	0	0	0
7	0	0	0	1	1	0	0	0	0	0	**1**	0	0	0	0
8	0	0	0	1	1	1	0	0	0	0	0	0	**1**	0	0
9	0	0	1	0	0	0	0	**1**	0	0	0	0	0	0	0
10	0	0	1	0	0	1	0	0	0	0	0	**1**	0	0	0
11	0	0	1	0	1	0	0	0	0	0	**1**	0	0	0	0
12	0	0	1	0	1	1	0	0	0	0	0	0	**1**	0	0
13	0	0	1	1	0	0	0	**1**	0	0	0	0	0	0	0
14	0	0	1	1	0	1	0	0	0	0	0	**1**	0	0	0
15	0	0	1	1	1	0	0	0	0	0	**1**	0	0	0	0
16	0	0	1	1	1	1	0	0	0	0	0	0	**1**	0	0
17	0	1	0	0	0	0	0	**1**	0	0	0	0	0	0	0
18	0	1	0	0	0	1	0	0	0	0	0	**1**	0	0	0
19	0	1	0	0	1	0	0	0	0	0	**1**	0	0	0	0
20	0	1	0	0	1	1	0	0	0	0	0	0	**1**	0	0
21	0	1	0	1	0	0	**1**	0	0	0	0	0	0	0	0
22	0	1	0	1	0	1	0	0	0	**1**	0	0	0	0	0
23	0	1	0	1	1	0	0	0	**1**	0	0	0	0	0	0
24	0	1	0	1	1	1	0	0	0	0	0	0	**1**	0	0
25	0	1	1	0	0	0	0	0	0	0	0	0	0	**1**	0
26	0	1	1	0	0	1	0	0	0	0	0	**1**	0	0	0
27	0	1	1	0	1	0	0	0	0	0	**1**	0	0	0	0
28	0	1	1	0	1	1	0	0	0	0	0	0	**1**	0	0
29	0	1	1	1	0	0	0	**1**	0	0	0	0	0	0	0
30	0	1	1	1	0	1	0	0	0	0	0	**1**	0	0	0
31	0	1	1	1	1	0	0	0	0	0	**1**	0	0	0	0
32	0	1	1	1	1	1	0	0	0	0	0	0	**1**	0	0
33	1	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
34	1	0	0	0	0	1	0	0	0	0	0	**1**	0	0	0
35	1	0	0	0	1	0	0	0	0	0	**1**	0	0	0	0
36	1	0	0	0	1	1	0	0	0	0	0	0	**1**	0	0
37	1	0	0	1	0	0	0	0	0	0	0	0	0	0	**1**
38	1	0	0	1	0	1	0	0	0	0	0	**1**	0	0	0
39	1	0	0	1	1	0	0	0	0	0	**1**	0	0	0	0
40	1	0	0	1	1	1	0	0	0	0	0	0	**1**	0	0
41	1	0	1	0	0	0	0	0	0	0	0	0	0	0	**1**
42	1	0	1	0	0	1	0	0	0	0	0	**1**	0	0	0
43	1	0	1	0	1	0	0	0	0	0	**1**	0	0	0	0
44	1	0	1	0	1	1	0	0	0	0	0	0	**1**	0	0
45	1	0	1	1	0	0	0	**1**	0	0	0	0	0	0	0
46	1	0	1	1	0	1	0	0	0	0	0	**1**	0	0	0
47	1	0	1	1	1	0	0	0	0	0	**1**	0	0	0	0
48	1	0	1	1	1	1	0	0	0	0	0	0	**1**	0	0

	S1	S2	S3	S4	A	B	C1	C2	C3	C4	C5	C6	C7	C8	C9
49	1	1	0	0	0	0	0	**1**	0	0	0	0	0	0	0
50	1	1	0	0	0	1	0	0	0	0	0	**1**	0	0	0
51	1	1	0	0	1	0	0	0	0	0	**1**	0	0	0	0
52	1	1	0	0	1	1	0	0	0	0	0	0	**1**	0	0
53	1	1	0	1	0	0	0	**1**	0	0	0	0	0	0	0
54	1	1	0	1	0	1	0	0	0	0	0	**1**	0	0	0
55	1	1	0	1	1	0	0	0	0	0	**1**	0	0	0	0
56	1	1	0	1	1	1	0	0	0	0	0	0	**1**	0	0
57	1	1	1	0	0	0	0	**1**	0	0	0	0	0	0	0
58	1	1	1	0	0	1	0	0	0	0	0	**1**	0	0	0
59	1	1	1	0	1	0	0	0	0	0	**1**	0	0	0	0
60	1	1	1	0	1	1	0	0	0	0	0	0	**1**	0	0
61	1	1	1	1	0	0	0	**1**	0	0	0	0	0	0	0
62	1	1	1	1	0	1	0	0	0	0	0	**1**	0	0	0
63	1	1	1	1	1	0	0	0	0	0	**1**	0	0	0	0
64	1	1	1	1	1	1	0	0	0	0	0	0	**1**	0	0

Tabela 5-4 – Tabela-verdade da análise paraconsistente do robô móvel Luiza.

Para extrair a expressão simplificada e construir o circuito digital referente à tabela-verdade descrita, foi utilizado o *software Karnaugh Minimizer* obtendo os seguintes resultados:

Expressões simplificadas:

C1 = |A*B*|C*D*|E*|F

C2 = C*D*|E*|F+A*B*|E*|F+|C*|D*|E*|F+|A*|B*|E*|F

C3 = |A*B*|C*D*E*|F

C4 = |A*B*|C*D*|E*F

C5 = A*E*|F+C*E*|F+|D*E*|F+|B*E*|F

C6 = A*|E*F+C*|E*F+|D*|E*F+|B*|E*F

C7 = E*F

C8 = |A*B*C*|D*|E*|F

C9 = A*|B*C*|D*|E*|F+A*|B*|C*D*|E*|F

O bloco lógico do circuito responsável pela análise paraconsistente pode ser visto na (figura 5-5) e o circuito completo no (anexo B).

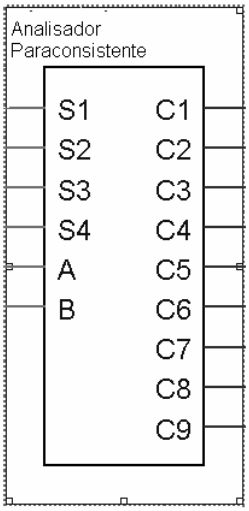


Figura 5-5 – Bloco lógico da análise paraconsistente implementado no ATMEGA8

5.3.5.3 Circuito de ativação dos comportamentos.

Após a análise paraconsistente e definição do comportamento a ser executado pelo robô, o circuito de ativação dos comportamentos irá gerar os sinais necessários para o comando dos motores, bem como a codificação dos sinais a serem enviados para a placa *driver* deles.

A existência de um sinal denominado “habilita”, o qual será responsável por bloquear a mudança de comportamento enquanto o robô estiver executando um comportamento balístico.

A construção deste circuito teve como base a tabela-verdade apresentada a seguir (tabela 5-5), que tem como variáveis de entrada os sinais C1, C2, C3, C4, C5, C6, C7, C8 e C9 e como variáveis de saída: HMD

(habilita motor direito), SMD (sentido do motor direito), HME (habilita motor esquerdo), SME (sentido do motor esquerdo).

C1	C2	C3	C4	C5	C6	C7	C8	C9	HMD	SMD	HME	SME
1	0	0	0	0	0	0	0	0				
0	1	0	0	0	0	0	0	0				
0	0	1	0	0	0	0	0	0				
0	0	0	1	0	0	0	0	0				
0	0	0	0	1	0	0	0	0				
0	0	0	0	0	1	0	0	0				
0	0	0	0	0	0	1	0	0				
0	0	0	0	0	0	0	1	0				
0	0	0	0	0	0	0	0	1				

Tabela 5-5 – Tabela-verdade do codificador dos comportamentos

O motor será “ligado” quando o sinal de habilita (HMD ou HME) receber o nível lógico 1 e o sentido da sua rotação será definido pelas variáveis (SMD e SME) de forma que o nível lógico 0 configura o motor para girar para frente e o nível lógico 1 configura o motor para girar para trás.

O bloco lógico do circuito de ativação dos comportamentos pode ser visto na (figura 5-6) e o circuito completo no (anexo C).

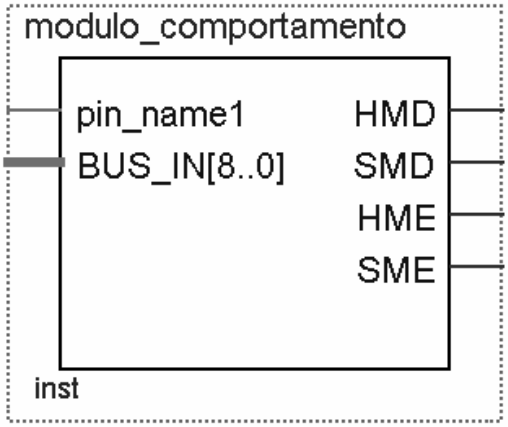


Figura 5-6 - Bloco lógico do circuito de ativação dos comportamentos

5.3.6 Placa *driver* dos motores

A placa driver de potência dos motores foi implementada utilizando-se o *chip* L293E.

Basicamente, o L293E irá receber o código binário diretamente do *Hardware* Reconfigurável e em seguida acionar os motores de acordo com a configuração definida no código.

A figura 5-7 mostra o diagrama do *chip* L293E

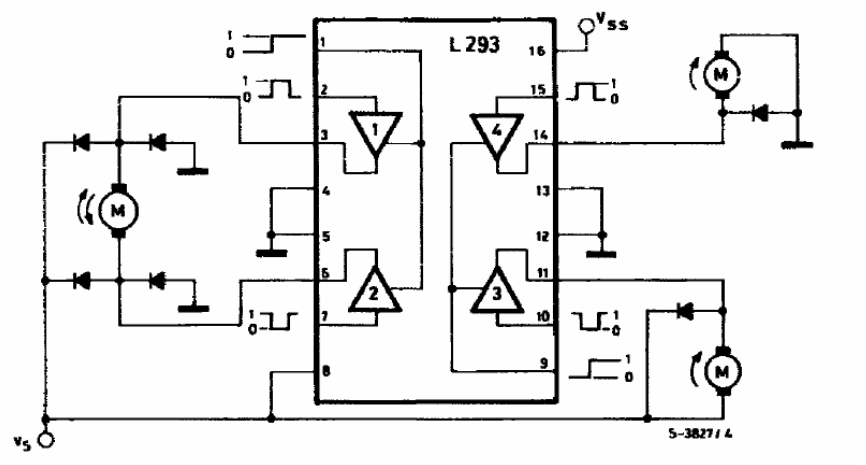


Figura 5-7 - **D**iagrama do *chip* L293E

5.3.7 Testes da arquitetura

Os testes foram efetuados em quatro cenários diferentes de acordo com os critérios a seguir:

1 – Em um ambiente sem obstáculos;

2 – Em um ambiente com um obstáculo retangular;

3 – Em um ambiente com dois obstáculos retangulares;

4 – Em um ambiente com um obstáculo cilíndrico.

No primeiro cenário, o objetivo foi validar o comportamento de seguir a luz.

Nos outros cenários, foi possível verificar os comportamentos referentes ao desvio dos obstáculos ao mesmo tempo em que o robô estava navegando em direção a maior fonte de luz.

Em todos os casos, o comportamento de “seguir a luz” foi mais eficiente com a luz do ambiente reduzida, fazendo assim que os sensores de luz não captassem interferências da luz ambiente.

Para um cenário livre de obstáculos, ou com a presença de um obstáculo retangular, o robô se comportou de forma adequada e de acordo com as expectativas do projeto.

Já com a presença de mais um obstáculo retangular (cenário 3), ele teve, em alguns casos, uma queda significativa de desempenho causada pela falta de sonares laterais.

Não foi obtido sucesso no desvio de obstáculos cilíndricos, fato justificado pela limitação que os sensores do tipo “sonar” têm ao receber de volta o eco da onda sonora quando bate em superfícies que não sejam planas.

6. Conclusão

6.1 Considerações finais

Este trabalho contemplou a união de conceitos já consagrados das técnicas da Lógica Paraconsistente Anotada Eτ, aplicadas à solução de problemas relacionados à navegação de robôs móveis, com a implementação focada na tecnologia emergente, denominada Hardware Reconfigurável.

Pode-se observar que as vantagens teóricas das uniões dessas tecnologias se confirmam na prática, fazendo com que as vantagens do uso de uma arquitetura de hardware mais flexível sejam somadas com sucesso a outras tecnologias já reconhecidas.

A Lógica Paraconsistente Anotada Eτ contribui de forma eficiente e eficaz no tratamento dos sinais muitas vezes conflitantes provenientes dos sensores, proporcionando uma análise dos dados de forma muito mais simples e natural.

As técnicas de comportamentos de robôs proporcionam uma atuação rápida e ao mesmo tempo adequada nas questões relacionadas com os movimentos servos ou balísticos que devem ser executados para que o mesmo atinja um determinado objetivo, que para este caso foi o de seguir a direção da maior fonte de luz.

E, especialmente, o *Hardware* Reconfigurável, que de forma satisfatória proporciona flexibilidade e desempenho a toda arquitetura, de acordo com seu propósito.

6.2 Contribuições deste trabalho

O uso concomitante das tecnologias apresentadas neste trabalho é relativamente recente na área da robótica.

As contribuições deste trabalho que merecem destaque são:

- A validação da união das três tecnologias utilizadas (Lógica Paraconsistente Anotada $E\tau$, robôs baseados em comportamentos e Hardware Reconfigurável.

- Durante o desenvolvimento deste projeto, muitas questões relacionadas ao desenvolvimento de um circuito digital capaz de controlar um sistema inteligente foi estudado e aplicado.

-As técnicas apresentadas aqui poderão ser utilizadas em aplicações dos mais variados tipos.

-O domínio de eficientes técnicas de navegação de robôs móveis proporciona à engenharia de produção a substituição das esteiras fixas por esteiras virtuais e flexíveis.

6.3 Trabalhos futuros

As técnicas abordadas neste trabalho podem ser utilizadas por diversas áreas científicas, com adequações que devem ser feitas de acordo com os objetivos a serem atingidos.

Para melhorar o desempenho do robô, pode-se ainda trabalhar as seguintes questões:

- A codificação dos sinais oriundos dos sonares pode ser estabelecida dentro do *Hardware* Reconfigurável.

- O número de sonares utilizados no projeto influencia diretamente no sensoriamento do mundo do robô. Por esse motivo, a utilização de mais sonares paraconsistentes deverá proporcionar um aumento significativo na capacidade de navegação do robô.

- A utilização de outros tipos de sensores, como por exemplo, sensores infravermelhos, sensores de contato, sensores laser, etc. poderão aumentar significativamente as chances de alcançar sucesso durante a maior parte do tempo de navegação do robô, seja em ambientes livres de obstáculos ou em ambientes tumultuados.

REFERÊNCIAS BIBLIOGRÁFICAS

(Abe, 1992) Abe, J. M. Fundamentos da Lógica Anotada. Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo SP, 1992.

(Abe, Costa *et al.*, 1999) Abe, J. M., N. C. A. D. Costa*, et al.* Lógica Paraconsistente Aplicada. São Paulo: Atlas. 1999

(Abe e Silva, 1998) Abe, J. M. e J. I. D. Silva. Inconsistency and electronic circuits. Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), v. 3, Artificial Intelligence.E. Alpaydin (ed.),
. Canada/Switzerland: ICSC Academic Press International Computer Science Conventions, 1998. 191-197 p.

(Agah e Bekey, 1997) Agah, A. e G. Bekey. Philogenetic and Ontogenetic Learning in a Colony of Interacting Robots. Autonomous Robots. 1997.

(Aporntewan e Chongstitvatana, 2001) Aporntewan, C. e P. Chongstitvatana. A Hardware Implementation of the Compact Genetic Algorithm. IEEE Congress on Evolutionary Computation. Seoul, South Korea, 2001. 624-629 p.

(Aragão, 1998) Aragão, A. C. D. O. S. Uma Arquitetura Sistólica para Soluções de Sistemas Lineares Implementada com Circuitos FPGAs. ICMC, USP, São Carlos, 1998.

(Arkin, 1998) Arkin, R. C. Behavior-Based Robotics. Cambridge, MA.: MIT Press. 1998

(Baraza, 2002) Baraza, J. C. A Prototype of a VHDL-based fault injectio tool: description and application. Journal of System Archicteture, v.47 Boston, p.847-867. 2002.

(Braitenberg, 1984) Braitenberg, V. Experiments in Synthetic Psychology. Cambridge MA.: MIT Press. 1984

(Brooks, 1991) Brooks, R. New Approaches to Robotics. Science, v.253. 1991.

(Costa, 2006) Costa, C. D. Projetando Controladores Digitais com FPGA. São Paulo: Novatec. 2006

(Costa, Gomes *et al.*, 2003) Costa, E. R., M. L. Gomes*, et al.* Um Mini Robô Móvel Seguidor de Pistas Guiado por Visão Local. VI Simpósio Brasileiro de Automação Inteligente. Bauru 2003.

(Costa e Abe, 2000) Costa, N. C. A. D. e J. M. Abe. Paraconsistência em informática e inteligência artificial. Estudos Avançados, v.14, Agosto. 2000.

(Costa, Abe *et al.*, 1991) Costa, N. C. D., J. M. Abe*, et al.* The Paraconsistent Logics. Pt, Zeitschr f. Math. Logik und Grundlagen Math, v.37, p.561-570. 1991.

(Da-Silva-Filho e Abe, 1999) Da-Silva-Filho, I. J. e J. M. Abe. Para-Analyser and Inconsistencies in Control Systems. Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'99). Honolulu, Hawaii, USA 1999.

(Da-Silva-Filho, 1997) Da-Silva-Filho, J. I. Implementação de Circuitos Lógicos Fundamentados em uma Classe de Lógicas Paraconsistentes Anotada. Universidade de São Paulo, São Paulo SP, 1997.

(Da-Silva-Filho, 1999) _____. Paraconsistente anotada de anotação com dois valores LPA2v com construção de algoritmo e implementação de circuitos eletrônicos. IEA, USP, São Paulo, 1999.

(Da-Silva-Filho, Abe *et al.*, 1997) Da-Silva-Filho, J. I., J. M. Abe*, et al.* Circuitos de Portas Lógicas Primitivas Fundamentados em Lógica Paraconsistente Anotada. III Workshop de Ibership. México D.F. 19-21 fev. 1997, 1997. p. 227-237 p.

(Fikes e Nilsson, 1971) Fikes, R. e N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, v.2, p.189-208. 1971.

(Flynn e Brooks, 1989) Flynn, A. e R. Brooks. Battling Reality. Al Memo. Cambridge MA. 1148 1989.

(Giralt, Chatila *et al.*, 1984) Giralt, G., R. Chatila*, et al.* An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots. First International Symposium on Robotics Research M. Brady and R. Paul 1984.

(Gómez, Rodrigues *et al.*, 1996) Gómez, A. T., A. G. Rodrigues*, et al.* Estudo da Variação do Tamanho da Lista Tabu no Problema de Escalonamento em um Sistema de Manufatura Flexível. Universidade do Vale do Rio dos Sinos – Unisinos. São Leopoldo RS 1996.

(Hogg, Martin *et al.*, 1991) Hogg, D., F. Martin*, et al.* Braitenberg Creatures. Cambridge MA.: MIT Epistemology and Learning. 1991

(Jablonsky e Posey, 1985) Jablonsky, J. e J. Posey. Robotics Terminology. New York. 1985. 1271-1303 p.

(Jones, 2004) Jones, J. L. Robot Programming. New York: McGraw-Hill. 2004

(Khatib, 1985) Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Proceedings of the IEEE International Conference on Robotics and Automation. St. Louis MO, 1985. 500-05 p.

(Mccarthy, Minsky *et al.*, 1955) Mccarthy, J., M. Minsky*, et al.* A Proposal of the Dartnouth Summer Research Project on Artificial Intelligence. august 31. 1955.

(Minsky, 1986) Minsky, M. The Society of the Mind. New York. 1986

(Moravec, 1977) Moravec, H. Towards Automatic Visual Obstacle Avoidance. Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Cambridge MA., 1977. p.

(Nickerson e Jasiobedzki, 2000) Nickerson, S. B. e P. Jasiobedzki. An Autonomous Mobile Robot for Known Industrial Environments. ARK (Autonomous Robot for a Knownnvironment) 2000.

(Nilsson, 1969) Nilsson, N. A Mobile Automaton: An Application of Artificial Intelligence Techniques. Proceedings of the International Joint Conference on Artificial Intelligence Washington DC: Autonomous Mobile Reports, 1969. p.

(Ozbayrak e Al., 1997) Ozbayrak, M. e A. K. T. E. Al. Part and Tool Flow Management in Multicell Flexible Manufacturing System. Proceedings of the 29th conference on Winter Simulation. Atlanta, Georgia, US: ACM Press, 1997. p.

(Prado, 1997) Prado, J. P. A. Uma Arquitetura para Inteligência Artificial Distribuída Baseada em Lógica Paraconsistente Anotada. Universidade de São Paulo, São Paulo SP, 1997.

(Prado e Abe, 1998) Prado, J. P. A. e J. M. Abe. Uma Arquitetura para Inteligência Artificial Baseada em Lógica Paraconsistente Anotada. São Paulo. 1998 (Coleção Documentos)

(Ribeiro, 2002) Ribeiro, A. A. D. L. Reconfigurabilidade Dinâmica e Remota de FPGAs. ICMC-USP, Universidade de São Paulo, São Carlos SP, 2002.

(Sacerdoti, 1974) Sacerdoti, E. Planning in a Hierarchy of Abstraction Spaces. Artificial Intelligence. 5**:** 115-35 p. 1974.

(Scatena, 2003) Scatena, J. M. Implementação de Mapas Topológicos para Navegação de Robôs Móveis baseadas em Computação Reconfigurável. ICMC, USP, São Carlos, 2003.

(Selfridge e Neisser, 1960) Selfridge, O. e U. Neisser. Pattern Recognition by Machine. Scientific American. 203**:** 60-68 p. 1960.

(Silva, 2005) Silva, S. R. Aplicações da Lógica Paraconsistente Anotada no método de campos potenciais para nevegação de robôs móveis. Universidade Paulista, São Paulo, 2005.

(Sussman, 1975) Sussman, G. A Computer Model of Skill Acquisition. New York: American Elsevier. 1975

(Vaughan, Stoy *et al.*, 2000) Vaughan, R. T., K. Stoy*, et al.* Blazing a trail: Insect-inspired Resource Transportation by a Robot Team. Robotics Research Laboratories. Los Angeles CA 2000.

(Walter, 1953) Walter, W. G. The Living Brain. W. W. Norton & Company Inc. . 1953.

(Wang, 1986) Wang, S. P. T. Animated Simulation of a Flexible Manufacturing System. Proceedings of the 18th conference on Winter simulation. Washington DC: ACM Press, 1986. p.

(Weiner, 1948) Weiner, N. Cybernetics, or Control and Communication in Animals and Machines. Wiley, v.New York. 1948.

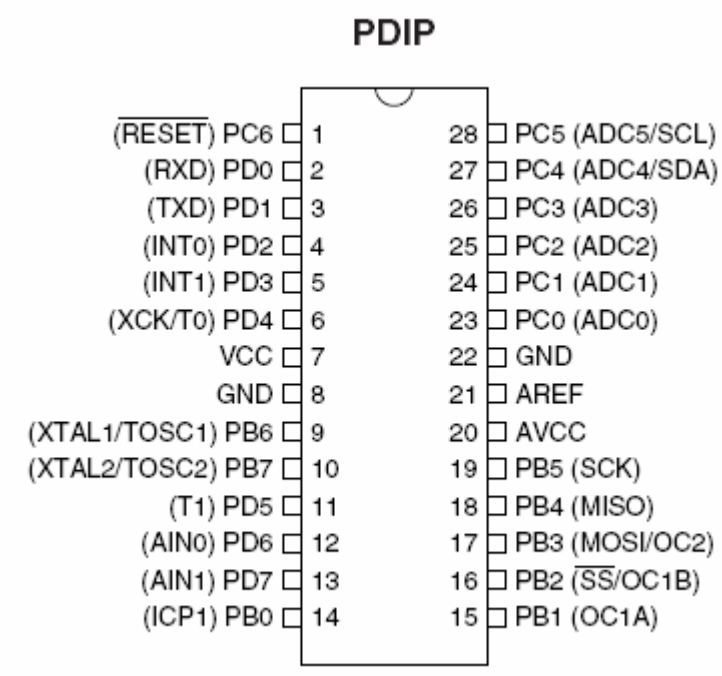
Anexo A

Características do microcontrolador ATMEGA8

O microcontrolador ATMEGA8 possui as seguintes características:

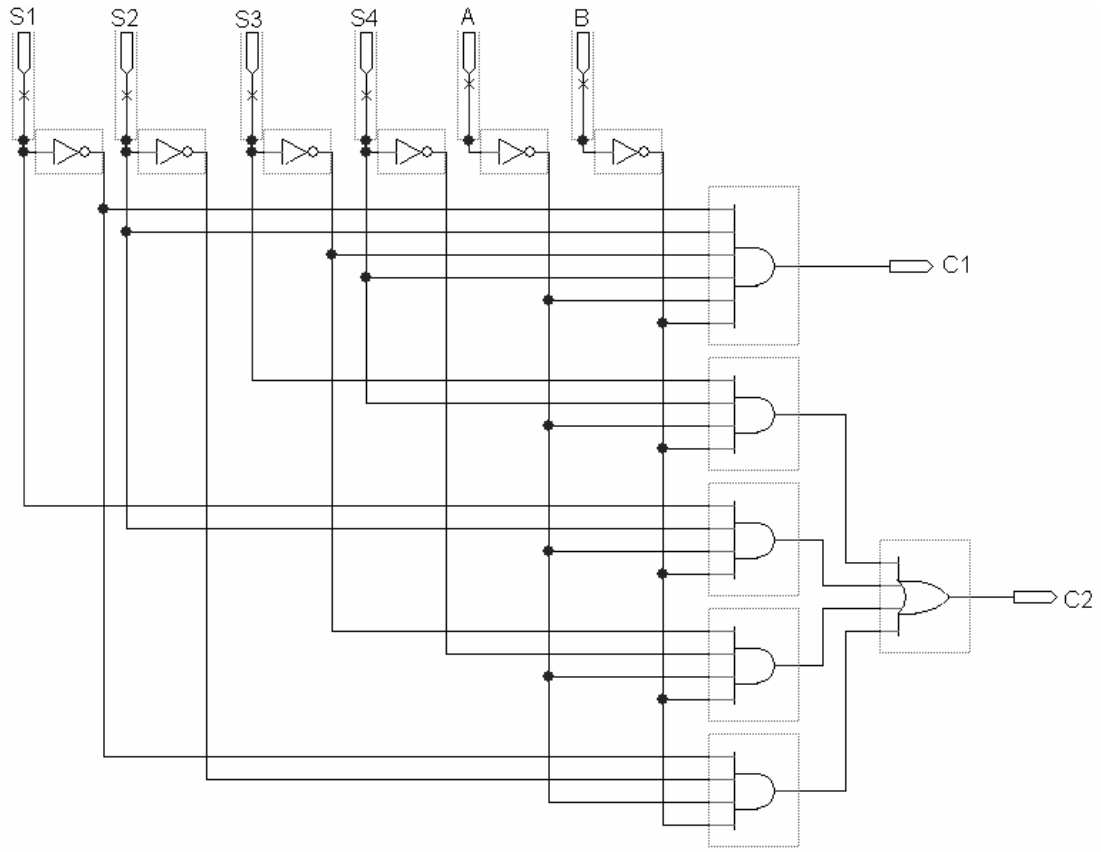
8 KB de memória flash ISP, 512 B de EEPROM, 1 KB de RAM interna, oscilador RC interno, várias fontes de interrupção externas e internas, cinco modos de baixo consumo, cinco fontes de relógio do sistema, 4,5 a 5V de operação, até 16MHz de relógio, dois contadores/temporizadores de 8 bits, um contador/temporizador de 16 bits, RTC com oscilador próprio, geração por hardware de até três canais PWM, conversor A/D de 6 canais, two-wire serial interface (TWI), interface serial USART, interface serial SPI, cão-de-guarda com oscilador próprio, comparador analógico interno.

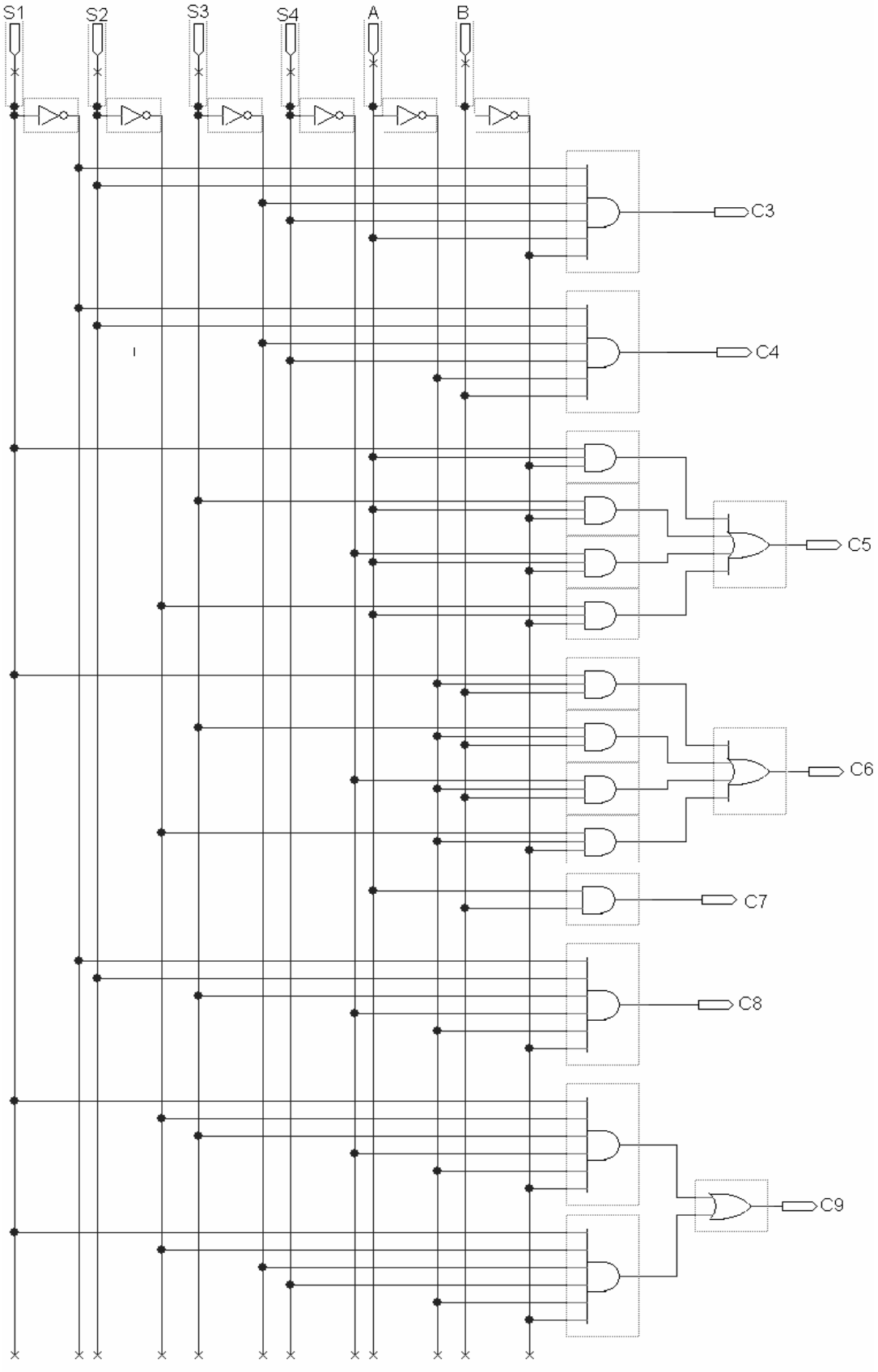
A pinagem do microcontrolador pode ser vista na figura a seguir:



Anexo B

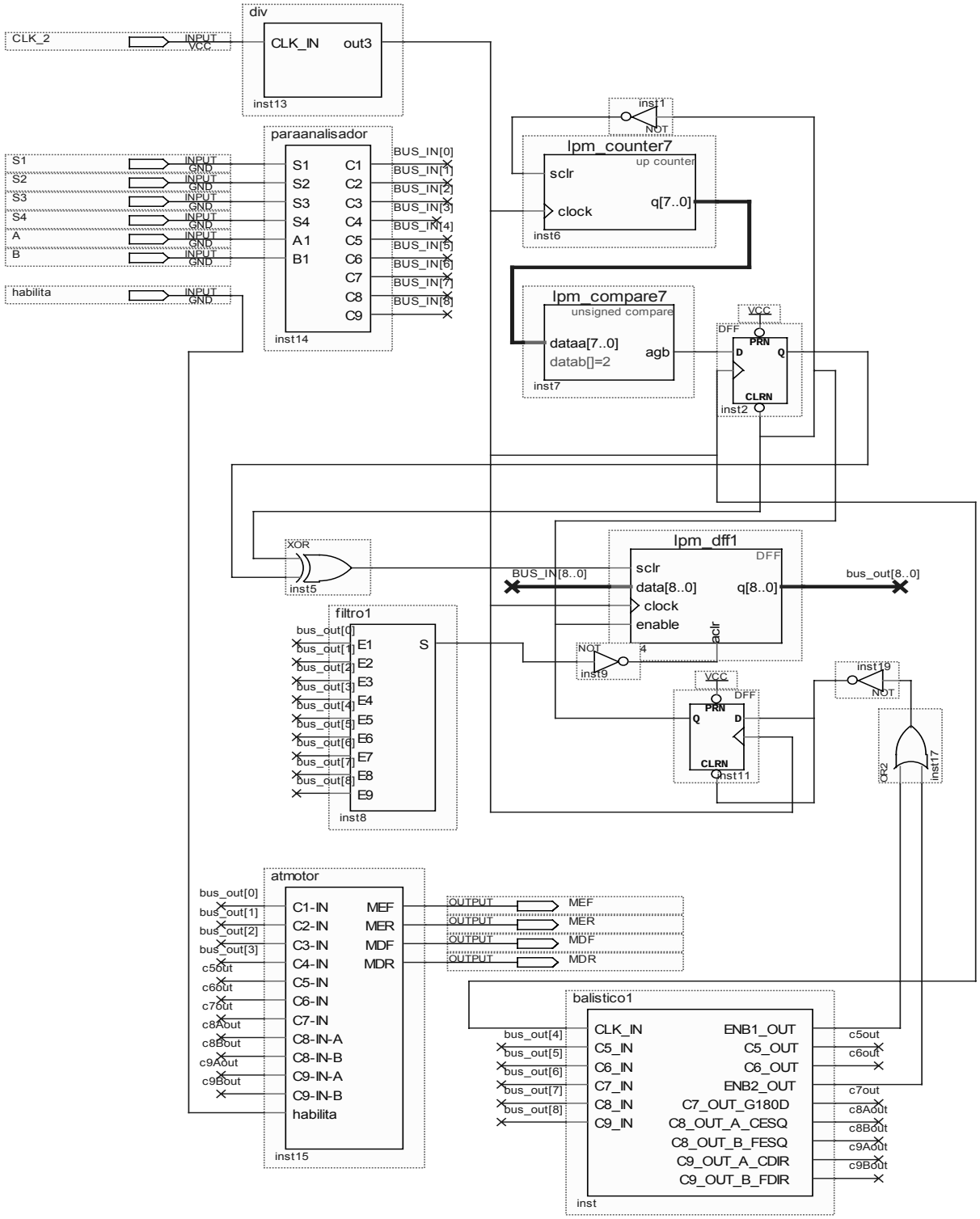
Diagrama completo do circuito de análise paraconsistente e comportamento




S1
S2
S3
S4
A
B
C3
C4
C5
C6
C7
C8
C9

Anexo C

Diagrama completo de ativação dos comportamentos



Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo