FACULDADE DE FÍSICA

Departamento de Electrónica e Computación

TESIS DOCTORAL

Evolutivo de Inteligencia em la optimización de funciones

Jorge Manuel Ferreira Barbosa Ribeiro

Santiago de Compostela, Dezembro de 2010

FACULDADE DE FÍSICA

Departamento de Electrónica e Computación

TESIS DOCTORAL

Evolutivo de Inteligencia em la optimización de funciones

Apresentada por:

Jorge Manuel Ferreira Barbosa Ribeiro

Dirigida por:

Prof. Doutor Manuel Fernández-Delgado

Prof. Catedrático José Carlos Maia Neves

Santiago de Compostela, Dezembro de 2010

MANUEL FERNÁNDEZ-DELGADO
Professor Contratado Doutor
Departamento de Electrónica e Computación
Universidade de Santiago de Compostela, Espanha

JOSÉ CARLOS MAIA NEVES
Professor Catedrático
Departamento de Informática
Universidade do Minho, Portugal

FAZEM CONSTAR

Que a memória intitulada "*Evolutivo de Inteligencia em la Optimización de funciones*" foi realizada por Jorge Manuel Ferreira Barbosa Ribeiro no âmbito do programa de doutoramento "*2151-071P01 Programa Interuniversitario Em Tecnoloxia Da Información",* sob nossa direcção no Departamento de Electrónica e Computação da Universidade de Santiago de Compostela (Espanha) e no Departamento de Informática da Universidade do Minho (Portugal), e constitui a Tese que apresenta para obter o grau de Doutor em Tecnologia de Informação (Área da Ciência da Computação e Inteligência Artificial).

Santiago de Compostela, 21 de Dezembro de 2010

_____ _____

Assinado: Manuel Fernández-Delgado
Director da Tese

Assinado: José Carlos Maia Neves
Co-diretor da Tese

_____

Assinado: Jorge Manuel Ferreira Barbosa Ribeiro
Doutorando

Evolutivo de Inteligencia em la optimización de funciones

por

Jorge Manuel Ferreira Barbosa Ribeiro

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Faculdade de Física - Departamento de Electrónica e Computación

DISSERTAÇÃO

Submetida para a obtenção do grau de

DOUTOR

em

TECNOLOGIAS DA INFORMAÇÃO
(Área da Ciência da Computação e Inteligência Artificial)

Dezembro de 2010

Direitos de Autor

A dissertação de Doutoramento intitulada "Evolutivo de Inteligencia em la optimizacin de funciónes", foi realizada por Jorge Manuel Ferreira Barbosa no âmbito do programa "*2151-07-1V01 - Programa Interuniversitario En Tecnoloxía Da Información*" da Universidade de Santiago de Compostela – Espanha, juntamente com o Departamento de Informática da Universidade do Minho – Braga, Portugal.

Tendo em consideração os assuntos e temáticas abordadas nesta dissertação **não é autorizada** a difusão do exemplar electrónio da tese, podendo ser disponibilizada para consulta a sua verso impressa apenas para fins de investigação. Neste sentido, é proibida a cópia total ou integral sem autorização prévia por escrito por parte do autor, do orientador e do co-orientador.

Santiago de Compostela, 22 de Dezembro de 2010,

.....

O Autor: Jorge Manuel Ferreira Barbosa Ribeiro

Prefácio

Esta dissertação apresenta o resultado do meu trabalho de investigação na área dos Sistemas Inteligentes, efectuado no âmbito do programa "*2151-07-1V01 Programa Interuniversitario En Tecnoloxía Da Información*" da Universidade de Santiago de Compostela – Espanha. Este trabalho teve início em Novembro de 2007, tendo sido realizado sobe a orientação do Professor Doutor Manuel Fernández-Delgado do grupo de *Sistemas Intelixentes* do *Departamento de Electrónica e Computación* da Universidade de Santiago de Compostela e do Professor Catedrático José Carlos Maia Neves do *Departamento de Informática* da Universidade do Minho, Braga – Portugal. A tese de doutoramento tem como título "Evolutivo de Inteligencia em la optimización de funciones". Tem como principal objectivo ampliar a capacidade dos tradicionais paradigmas da Inteligência Artificial, no sentido de tirar partido dos paradigmas simbólico, evolutivo e conexionista. Pretende-se disponibilizar mecanismos para criar sistemas formais evolutivos que ajudem na optimização e resolução de problemas complexos. Neste contexto, a combinação destes paradigmas permite obter a melhor quantificação do *universo do discurso* representado através de funções lógico matemáticas materializadas através da *Programação em Lógica Estendida*. Na realidade, não pretendemos apresentar nenhum acto revolucionário mas sim seguir uma linha de investigação bastante consolidada, melhorando-a e aperfeiçoando-a, de modo a contribuir para a sociedade do conhecimento através do acréscimo de ideias concretas numa parte específica do seu domínio.

Agradecimentos

Esta tese é o resultado de cerca de três anos de investigação realizado durante um período intenso de trabalho profissional e académico na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo. Este trabalho foi realizado entre duas Instituições académicas, a Universidade de Santiago de Compostela – Espanha e a Universidade do Minho – Braga, Portugal. Em primeiro lugar gostaria de agradecer aos meus orientadores, Professor Catedrático José Carlos Maia Neves e ao Professor Doutor Manuel Fernández-Delgado pela sua dedicação e empenho no acompanhamento dos trabalhos e pela liberdade que me deram de escolher o caminho de investigação que entendi e, ao mesmo tempo fornecendo-me uma visão clara sobre os assuntos que ia assimilando. Foi, e é um prazer e uma honra trabalhar com vocês.

Estou muito grato ao grupo de Sistemas Intelixentes do Departamento de Electrónica e Computacion do Departamento de Física da Universidade de Santiago de Compostela, em especial à Eva Cernadas e ao Professor Doutor Sénem Barro. Um profundo agradecimento ao grupo de Inteligência Artificial do Departamento de Informática da Universidade do Minho, por todo o apoio e partilha de experiências e ideias. Gostaria de agradecer em especial aos colegas José Machado, António Abelha, Paulo Novais e César Analide.

Gostaria de expressar a minha gratidão aos meus avós, José e Elisa que sempre acreditaram em mim e me deram apoio em todos os momentos da minha vida, mesmo durante a sua ausência neste mundo. *Estamos juntos*....

Por serem a luz da minha vida e o tesouro que tenho, gostaria de agradecer às minhas duas filhas, Mariana e Ana Jorge pelo seu carinho e ternura. Gostaria de lhes pedir desculpa pela falta de acompanhamento que lhes dei durante estes anos,

em especial durante o período de doutoramento.

Como "... *atrás de um grande homem está uma grande mulher...*", gostaria de dedicar esta tese de doutoramento à minha esposa, Anabela, pelo seu apoio e encorajamento que me deu ao longo dos anos, mesmo em momentos extremamente difíceis. *És muito especial para mim.*

Por fim, dedico a todos os que me querem bem....

We have here no revolutionary act but the natural continuation of a line that can be traced through centuries.

[Einstein in His own words, Anne Rooney, 2006]

Resumo

Esta tese de doutoramento aborda diferentes áreas de investigação, nomeadamente os paradigmas, simbólico, conexionista e evolutivo. A combinação de técnicas da *Inteligência Artificial*, como o paradigma lógico e os sistemas evolutivos permite tirar partido das vantagens de cada uma delas na resolução de problemas complexos. Os *Algoritmos Evolucionários* [Ang2000] correspondem a modelos da evolução natural [Dar1859] amplamente utilizados em cenários de optimização. Estes algoritmos baseiam-se na adaptação colectiva e na capacidade de aprendizagem dos indivíduos, em que indivíduos de uma população representam uma solução possível para a resolução de um problema. No inicio do processo evolutivo, os indivíduos na população são inicializados aleatoriamente podendo ser modificados pelos operadores de selecção, mutação e cruzamento, levando estas modificações à evolução desses indivíduos. Durante o processo, melhores e melhores indivíduos irão aparecer, baseando-se a avaliação de acordo com uma métrica de avaliação.

Os *Algoritmos Genéticos* e a *Programação Genética* [Koz1992] [Koz1994] são dois diferentes tipos de *Algoritmos Evolucionários*. A *Programação Genética*, utilizada neste trabalho, segue uma abordagem diferente dos *Algoritmos Genéticos*. Em vez de codificar *strings* que representam uma solução para um problema particular, a *Programação Genética* gera programas executáveis, sendo neste estudo, apresentados numa representação em árvore através de programas lógicos.

Numa outra área da *Inteligência Artificial*, a *Representação do Conhecimento*, como forma de descrever o mundo real (i.e. *universo do discurso*) baseada na forma mecânica, lógica, ou outros meios será sempre considerada em função da capacidade de descrever o conhecimento existente, assim como estando intimamente associadas a

mecanismos de raciocínio [Hal1986]. O uso do paradigma *Lógico* (materializado aqui através da *Programação em Lógica Estendida*) tem vindo a ser aplicado com sucesso em várias áreas do conhecimento [Sha2000], [RM$^+$2002], [MA$^+$2006a], [CN$^+$2008], [NA$^+$2010]. Por outro lado, o conhecimento e as crenças são geralmente incompletas, contraditórias, ou sensíveis ao erro, sendo desejável o uso de ferramentas formais para lidar com problemas que emergem do uso de informação contraditória, incompleta, ambígua, imperfeita ou mesmo omissa. Nas últimas décadas várias técnicas não clássicas orientadas para a modelação do *universo do discurso* (em cenários em que a informação é incompleta), assim como em disponibilizar procedimentos de raciocínio na área dos sistemas inteligentes tem vindo a ser propostos [She1991] [Sha1992] [NM$^+$1997] [LP$^+$1998] [Zad2001] [BK2005] [Str2006] [PL2007].

A extensão à *Programação em Lógica* provou ser uma ferramenta adequada para fins de representação do conhecimento e raciocínio, em particular, quando se tenciona contemplar situações em que a informação é vaga, imperfeita, ou incompleta. Historicamente, o raciocínio inserto tem vindo a ser associado à teoria das probabilidades [Sub2001], contudo, investigações promissoras tem sido realizadas usando outros formalismos ligando a lógica com a teoria das probabilidades. Estes formalismos incluem a teoria dos conjuntos *fuzzy* [Zad2001], lógicas multivalor [Loy2004], a teoria da evidência *Dempster-Shafer* [Sha1992], formalismos híbridos (e.g. numéricos e não numéricos) e lógicas não *standards*.

A *Abductive Logic Programming* [KK1998] é um promissor paradigma computacional, sendo reconhecido como uma forma de resolver algumas limitações da programação em lógica no que respeita ao elevado nível da representação do conhecimento, assim como para tarefas de raciocínio. Por outro lado, a *abduction* corresponde a uma forma de raciocínio para tratar conhecimento incompleto ou contraditório

através da forma de raciocínio hipotético, sendo mais apropriado para a geração de modelos e verificação da satisfabilidade.

No entanto, modelos qualitativos e raciocínio qualitativo [For1996] tem vindo a ser estudados na área da *Inteligência Artificial*, em particular face ao crescimento da necessidade de oferecer suporte a processos de tomada à decisão. Neste sentido, a avaliação do conhecimento que emerge do resultado dos programas lógicos tornou-se um ponto de investigação [SA2000] [Hal2005] [KR2010]. No contexto da teoria qualitativa da incerteza *Kuipers e Forbus* [Kui1994] [For1996] apresentadam com sucesso uma forma axiomática baseada na substituição do intervalo probabilístico [0,..,1] com um conjunto ordenado de valores simbólicos a partir da lógica de predicados multivalor. Neste sentido, esta teoria aborda o problema da representação, mas não a qualidade da informação associada à representação do *universo do discurso*.

Este trabalho de doutoramento apresenta uma abordagem formal na área da *Inteligência Artificial* para a modelação da criatividade de sistemas evolutivos que materializam a *Representação do Conhecimento* numa forma de raciocínio, através do uso dos *Algoritmos Evolucionários*, em especial a *Programação Genética*. Com a modelação deste tipo de sistemas, seremos capazes de construir mundos dinâmicos virtuais de entidades complexas, interagindo umas com as outras, materializando o seu conhecimento através de programas evolucionários, permitindo-nos obter a melhor teoria lógico matemática e consequentemente, a melhor modelação do sistema em relação ao problema em observação. Nesta abordagem, a selecção das soluções candidatas são avaliadas por um único critério: a medida da qualidade da informação associada a cada teoria ou programa lógico que representa e modela o *universo do discurso*.

De uma maneira geral, o objectivo principal deste trabalho centra-se no desen-

volvimento de comunidades (ou sistemas) virtuais evolutivos que ajudem na optimização de cenários reais, ou por outras palavras, baseiam-se nos fundamentos teóricos e na formalização de ideias intuitivas através da sua aplicabilidade em casos reais. Pretende-se combinar os sistemas conexionistas e sistemas simbólicos da *Inteligência Artificial* a fim de optimizar o *universo do discurso*, representados por funções da teoria lógico matemática, ou seja, investigar até que ponto a aplicação de conjuntos de técnicas da *Inteligência Artificial* trazem para o poder descritivo da *Programação em Lógica Estendida* [NM$^+$1997] a fim de obter a melhor quantificação do *universo do discurso* que pode ser representado como funções lógico matemáticas e, consequentemente, optimizar essas funções.

Abstract

This thesis draws upon several different areas of research, namely the symbolic, connectionist and evolutionary paradigms. The combination of Artificial Intelligence techniques, like the logic paradigm and the evolutive systems permit to mix the advantages of each one in the resolution of optimization problems. Evolutionary Algorithms (EA) [Ang2000] are a model of natural evolution [Dar1859] that is often used to solve optimization scenarios. These algorithms are based on the collective adaptation and learning ability of individuals. Individuals from a population and each individual represent a possible solution of the problem. At the beginning of the evolving process, the individuals in the population are initialized randomly and can be modified by the selection, mutation and recombination operators. These modifications lead to the evolution of individuals; during the process better and better individuals appear and the evaluation is based according to a fitness metric. Thus in later generations individuals representing better solutions appear more likely.

On other hand, Genetic Algorithms (GA) and Genetic Programming (GP) are two different EA [ES2003]. GP, the subject of this study, follows a different approach from the GA. Instead of encoding strings that represent a particular problem solution, GP breeds executable computer programs, here presented has tree-based representation of logical programs.

In another field of *Artificial Intelligence*, the Knowledge Representation as way to describe the real world (universe of discourse) based on mechanical, logical, or other means, will always be a function of the system's ability to describe the existent knowledge and their associated reasoning mechanisms. The use of Logic paradigm (materialized with Logic Programming) has been applied with success in many areas

[Sha2000], [RM$^+$2002], [MA$^+$2006], [CN$^+$2008], [NA$^+$2010]. From other point of view, knowledge and belief are generally incomplete, contradictory, or error sensitive, being desirable to use formal tools to deal with the problems that arise from the use of incomplete, contradictory, ambiguous, imperfect, nebulous, or missing information.

In the past few decades, many non-classical techniques for modeling the universe of discourse and reasoning procedures of intelligent systems have been proposed [She1991] [Sha1992] [NM$^+$1997] [LP$^+$1998] [Zad2001] [BK2005] [Str2006] [PL2007]. The Extended Logic Programming proved to be an adequate tool for knowledge representation and reasoning, in particular, when one intends to endorse situations where the information is vague, imperfect, or incomplete. Historically, uncertain reasoning has been associated with Probability Theory [Sub2001] but promising research have been done using other formalisms linking logic with probability theory. These formalisms include the theory of fuzzy sets [Zad2001], multi-valued logics [Loy2004], the Dempster-Shafer theory of evidence [Sha1992], hybrid (i.e. numerical and non-numerical) formalisms, and non standard logics.

The Abductive Logic Programming (ALP) [KK1998] is a promising computational paradigm and has been recognized as a way to solve some limitations of logic programming with respect to higher level knowledge representation and reasoning tasks. Abduction is a way of reasoning on incomplete or uncertain knowledge, in the form of hypothetical reasoning, more appropriate to model generation and satisfiability checking.

However, qualitative models and qualitative reasoning have been around in Artificial Intelligence research for some time [For1996] [Kui1994], in particular due the growing need to offer support in decision-making processes. The evaluation of knowledge that stems out from logic programs becomes a point of research [SA2000]

[Hal2005] [KR2010]. In the context of the Qualitative Uncertainty Theory, *Kuipers and Forbus* [Kui1994] [For1996] presented with success an axiomatic approach based on the substitution of the probabilistic interval $[0 \ldots 1]$ with an ordered set of symbolic values for multi-valued predicate logic. In this sense, this theory approaches the problem of the representation but not the quality-of-information.

This doctoral work presents an approach to the modulation of creative evolutive systems that materializes the knowledge representation in a reasoning way, through the use of problem resolution methodologies and by techniques associated to the use of EA, in special the GP. With the modulation of these type of systems, we be able to build a dynamic virtual world of complex and interacting entities, here materialized as evolutionary programs, that allows to get the best logical theory, and consequently the best modulation of the system for the problem in observation. The selection of the candidate solutions is judged by one criterion alone: a measure of the quality-of-information that stems from those logical theories or programs that represent the universe of discourse.

The main objective of this work is the development of evolving virtual communities (or systems) in order to help the optimization of real scenarios, or by other words, based on the theoretical fundaments and in the formalization of intuitive ideas through the applicability in real scenarios. We intent to combine the connectionist and symbolic systems of the Artificial Intelligence in order to optimize the universe of discourse represented as functions of the logic mathematic theory, i.e., investigate till where the application of a set of Artificial Intelligence techniques brings for the descriptive power of the Logic Programming in order to achieve to the best quantification of the universe of discourse, being represented as logic mathematic functions and, consequently, optimize those functions.

Índice

1 Introdução **1**

1.1 Contexto e Motivação 5

1.2 Questões e Hipótese de Trabalho 7

1.3 Paradigmas Utilizados 7

1.4 Objectivos, Delimitação e Pressupostos 8

1.4.1 Objectivos 8

1.4.2 Delimitação 9

1.4.3 Pressupostos 11

1.5 Metodologia 12

1.6 Estrutura e Resumo da Dissertação 13

1.7 Notação e Terminologia 14

2 Lógica e Programação em Lógica **17**

2.1 Introdução Histórica 17

2.2 Tipos de Lógicas 21

2.2.1 Lógicas Clássicas 23

2.2.2 Lógicas Não Clássicas 27

2.2.3 Outros Tipos de Lógicas 28

2.3 Fundamentos 29
2.3.1 Fórmulas Lógicas 29
2.3.2 Semântica das Fórmulas 34
2.3.3 Inferência Lógica 35
2.3.4 Teoria dos Modelos e Teoria da Prova 37
2.4 Lógica e Teoria do Raciocínio 41
2.4.1 Introdução 41
2.4.2 Dedução, Indução e Abdução 43
2.4.3 Raciocínio Não Monótono 45
2.5 Programação em Lógica 47
2.5.1 Introdução 47
2.5.2 Limitação da Programação em Lógica 49
2.5.3 Extensão à Programação em Lógica 49
2.5.4 Sistemas de Programação em Lógica 51
2.6 Conclusão 52

3 Inteligência Evolucionária e Programação em Lógica 53

3.1 Introdução 54
3.2 Computação Evolucionária 55
3.2.1 Introdução 55
3.2.2 Componentes de um Algoritmo Evolucionário 56
3.2.3 Ciclo Evolutivo 57
3.2.4 Programação Genética 58
3.3 Sistemas Conexionistas 61
3.3.1 Introdução 61
3.3.2 Características 62

3.3.3 Topologias 64
3.3.4 Contribuição 66
3.4 Computação Evolucionária e Programação em Lógica 67
3.5 Sistemas Conexionistas e Programação em Lógica 69
3.6 Conclusão 70

4 Inteligência Evolucionária - Modelação Computacional 73
4.1 Motivação 74
4.2 Representação do Conhecimento 77
4.2.1 Introdução 77
4.2.2 Trabalho Relacionado 78
4.2.3 Representação da Informação e Mecanismo de Raciocínio . . . 81
4.3 Qualidade da Informação 88
4.3.1 Trabalho Relacionado 88
4.3.2 Medida da Qualidade da Informação 91
4.4 Inteligência Evolucionária 94
4.4.1 Esquemático Arquitectural 94
4.4.2 Genoma 99
4.4.3 Genes de Processamento 101
4.4.4 Genes de Conexão 102
4.4.5 Processo de Inferência 105
4.4.6 Processo Evolutivo 119
4.5 Conclusão 129

5 Inteligência Evolucionária - Operacionalização do Sistema 131
5.1 Introdução 131

5.2 Representação da Informação e Mecanismo de Raciocínio 132
5.2.1 Representação da Informação 132
5.2.2 Tratamento da Evolução Temporal da Informação 140
5.3 VirtualInspector 145
5.3.1 Motivação 145
5.3.2 Implementação 146
5.3.3 Acompanhamento do processo de inferência 153
5.4 Conclusão 155

6 Conclusões **157**
6.1 Síntese da Dissertação 157
6.2 Tese 158
6.3 Contribuições, Originalidades e Aplicabilidade 160
6.4 Limitações e Trabalho Futuro 162

7 Anexos **165**
A Conjunto de soluções ou cenários para as questões sobe observação . . 165
B Excerto do código em PLE usado no estudo 199
C Excerto do código em PLE contemplando a evolução temporal e a negação explícita 205

Bibliografia **211**

Lista de Figuras

2.1 - Estrutura exemplificativa da relação familiar [NM2000]. 31
3.1 - Forma básica do processo evolutivo de um Algoritmo Evolucionário. 58

3.2 - Exemplos da representação da Programação Genética usando fórmulas aritméticas e fórmulas lógicas. 59
3.3 - Exemplo da operação de mutação usando a Programação Genética. 60

3.4 - Exemplo da operação de cruzamento usando a Programação Genética. 61

3.5 - Estrutura típica de um neurónio das Redes Neuronais Artificiais 63

3.6 - Estrutura arquitectural das RNA do tipo Unidireccional. 64

3.7 - Estrutura arquitectural das RNA do tipo Recorrente 65

4.1 - Exemplo da representação gráfica da medida da Qualidade da Informação associada a uma teoria ou programa lógico. 93
4.2 - Esquemático arquitectural do sistema. 95

4.3 - Arquitectura processual da criação do Intelecto Virtual. 98

4.4 - Exemplo de um genoma com três tipos de neurónios, cada um com um gene de processamento simples, dois, um ou três genes de conexão. 100
4.5 - Esquemático dos genes de processamento. 101

4.6 - Esquemático de um gene de conexão. 103

4.7 - Esquemático da aplicação do operador de cruzamento. 104

4.8 - Esquemático da aplicação do operador de mutação. 105

4.9 - Exemplificação de um simples modelo de base de dados relacional. 106

4.10 - Ilustração da quantificação de todas as teorias (ou programas lógicos) que interpretam a questão 1. 112

4.11 - Exemplificação da criação do Intelecto Virtual para uma Questão num momento t=1.. 116
4.12 - Exemplificação da instanciação do Intelecto Virtual para uma Questão num momento t=1.. 117
4.13 - Exemplificação do genoma arquitectural de uma solução para a interpretação de uma questão no momento t=1... 118
4.14 - Exemplificação do genoma das soluções para a interpretação de uma questão. 119
4.15 - Pseudo-código da criação de uma Rede Neuronal Evolucionária. 120
4.16 - Pseudo-código para a pesquisa dos pesos das conexõoes de uma Rede Neuronal Evolucionária. 121
4.17 - Exemplificação da criação do intelecto virtual para uma questão num momento t=2. 122
4.18 -Exemplificação do genoma arquitectural de uma solução para a interpretação de uma questão no momento t=2. 123
4.19 -Dependências de relacionamento na interpretação das questões colocadas ao sistema. 124
4.20 -Exemplificação da parametrização do gene de conexão 124
4.21 - Ilustração do processo de operacionalização do operador de cruzamento. . 125
4.22 - Ilustração do processo de operacionalização do operador de mutação. . . .126
4.23 -Exemplificação da criação de novo conhecimento a partir de duas questões colocadas ao sistema 126
4.24 -Exemplificação da autoorganização do Intelecto Virtual após o momento t=2. 127
5.1 – VirtualInspector – Arquitectura. 146
5.2 - VirtualInspector – Interface Principal. 147
5.3 - VirtualInspector – Carregamento da Informação e submissão de questões.. 148

5.4 - VirtualInspector – Apresentação das soluções para a interpretação do problema..

. 149
5.5 - VirtualInspector – Instanciação do arquétipo estrutural para a interpretação do

problema em observação. 151
5.6 - VirtualInspector – Exemplo da evolução do itelecto arquitectural.. 152

5.7 - VirtualInspector – Instanciação do intelecto arquitectural... 153

A.1 - Representação em PLE das soluções ou cenários que interpretam a questão (24).
A.2 - Representação e Quantificação da QoI das soluções ou cenários que interpretam

a questão (24).
A.3 - Gráfico da QoI global associada aos cenários envolvidos na interpretação da

questão (24).
A.4 - Representação segundo o modelo relacional das soluções ou cenários que inter-

pretam a questão (24).
A.5 - Representação em PLE das soluções ou cenários que interpretam a questão (25).

A.6 - Representação e Quantificação da QoI das soluções ou cenários que interpretam

a questão (25)
A.7 - Gráfico da QoI global associada aos cenários envolvidos na interpretação da

questão (25).
A.8 - Representação segundo o modelo relacional das soluções ou cenários que inter-

pretam a questão (25).
A.9 - Representação em PLE das soluções ou cenários que interpretam a questão (26).

A.10 - Representação e Quantificação da QoI das soluções ou cenários que interpre-

tam a questão (26).
A.11 - Gráfico da QoI global associada aos cenários envolvidos na interpretação da

questão (26).
A.12 - Representação segundo o modelo relacional das soluções ou cenários que in-

terpretam a questão (26).

"The major achievement of Aristotle was the founding of the science of logic"
[E.g Kline p.53, 1972]

Capítulo 1

Introdução

A *Lógica* [GHR1995] é uma ciência que está tradicionalmente associada à investigação sistemática da validade ou invalidade dos esquemas de inferência, sendo constituída por um conjunto de técnicas orientadas para a resolução de uma classe de problemas, pelo menos parcialmente, através da interpretação e manipulação de um cálculo formal (associado à *Lógica*). Nas últimas décadas, a *Lógica Matemática* evoluiu de uma fase de pura actividade teórica para se tornar numa ferramenta de uso prático, influenciando decisivamente a evolução das ciências da computação, em especial, através da criação de mecanismos de raciocínio [Hal1986] na resolução de problemas complexos.

A *Lógica* e os *Programas Lógicos* [GL1990] [Gin1991] [Kow2008] emergiram como formalismos atractivos para a representação do conhecimento, assim como uma abordagem à pesquisa na resolução de problemas. Por outro lado, a *Representação do Conhecimento*, como forma de representar o mundo real, baseada na mecânica, na lógica, ou noutro meio, é amplamente caracterizada em função da actividade do sistema em descrever o conhecimento existente, estando intimamente associada a mecanismos de raciocínio [BG1994]. O uso do paradigma lógico materializado através da programação em *Lógica*, tem vindo a ser aplicado com sucesso em várias áreas do conhecimento, como na medicina, na lei civil, nos sistemas multi-agente, nas entidades virtuais, em ambientes de assistência na área da saúde, em ambientes de tomada à decisão, entre outros [RM$^+$2002] [PP2005] [AA2006] [MA$^+$2006a] [MA$^+$2006b] [PL2007] [CN$^+$2008] [MA$^+$2008] [MM$^+$2009]. Por outro lado, o conhecimento e as convicções (ou crenças) são geralmente incompletos, contraditórios e sensíveis ao erro, sendo desejável o uso de ferramentas formais para lidar com problemas associados à ***in-***

formação incompleta[1] ou ambígua.

Vários esforços têm sido realizados para interligar a *Lógica Matemática* com metodologias orientadas para a representação do conhecimento e a criação de metodologias de raciocínio baseadas na lógica [Hal1986]. Historicamente, o raciocínio incerto tem sido associado à teoria das probabilidades [LS1997] [Sub2001], mas recentes e promissoras investigações tem sido realizadas usando outros formalismos ligando a lógica com outras teorias. Estes formalismos incluem a teoria dos *Fuzzy Set*, *Multi-valued Logic*, teoria da evidência *Dempster-Shafer*, formalismos híbridos e lógicas não standards.

A *Abductive Logic* [KK1998] tem sido um promissor paradigma computacional, vindo a ser reconhecido como uma forma de resolver algumas limitações da *Programação em Lógica* no que respeita ao elevado nível da *representação do conhecimento*, assim como em estar orientado para tarefas de raciocínio. A *abdução* (ou *abduction* na terminologia anglo-saxónica) corresponde a uma forma de raciocínio para tratar conhecimento incompleto ou contraditório através da forma de raciocínio hipotético, sendo tradicionalmente mais apropriada para a geração de modelos e verificação da sua satisfabilidade.

A *Programação em Lógica Estendida* [Nev1984] [LS1993] [PP1994] [NM$^+$1997] [LP$^+$1998] é outra técnica não clássica para modelar o *universo do discurso*[2], e tem vindo a provar ser uma ferramenta adequada para a representação do conhecimento e de raciocínio, em particular, quando se tenciona lidar com situações em que a informação é vaga, imperfeita ou incompleta. Contudo, nos últimos anos, modelos qualitativos e raciocínio qualitativo [For1996] tem vindo a ser estudados na área da *Inteligência Artificial*, em particular face ao crescimento da necessidade de oferecer suporte a processos de tomada à decisão. Neste sentido, a aplicação do conhecimento que emerge do resultado dos programas lógicos tornou-se um ponto de investigação. Por outro lado, uma das mais simples e promissoras formas de avaliar o conhecimento

[1] Informação incompleta – cenário do mundo em que toda ou parte da informação é ambígua, desconhecida, imprecisa ou sensível ao erro.

[2] O ***universo do discurso*** é o assunto de uma base de dados ou modelo: que corresponde a parte do mundo em consideração. Corresponde a uma ferramenta analítica usada na lógica dedutiva, em especial na lógica de predicados. Por outro lado, na semântica da teoria dos modelos, corresponde ao conjunto de entidades em que o modelo se baseia. O *universo do discurso* é geralmente atribuído a *Augustus De Morgan* (1846) e usado por *George Boole* (1854) no seu estudo "*Laws of Thought*" [Boo1854].

a partir de programas lógicos é o conceito de *Qualidade-da-Informação* [AN+2006] [NM+2007]. Este conceito demonstrou com sucesso a sua aplicabilidade em ambientes dinâmicos, assim como para propósitos da tomada à decisão [LC+2008] [MM+2009] [RN+2009] [NA+2010] [NS+2010] [MA+2010]. O seu objectivo é a construção de um processo de quantificação da *qualidade da informação* associada aos programas lógicos (ou teorias) durante o processo de inferência na resolução de problemas em cenários em que exista informação incompleta.

Numa outra área de investigação da *Inteligência Artificial*, os *Algoritmos Evolucionários* [ES2003] têm vindo a ser aplicados com sucesso em várias áreas do conhecimento, em particular em processos complexos de optimização. Correspondem a algoritmos baseados no modelo de evolução natural [Dar1859], os quais tentam encontrar uma solução óptima para resolver problemas. Estes algoritmos baseiam-se na colecção adaptativa e na capacidade de aprendizagem dos indivíduos, em que os indivíduos formam uma população e cada indivíduo representa uma possível solução para o problema. No inicio do processo de evolução, os indivíduos numa população são inicializados aleatoriamente, podendo ser modificados por um conjunto de operadores. Estas modificações conduzem à evolução dos indivíduos em que durante o processo, melhores e melhores indivíduos vão aparecer. A avaliação do desempenho de cada indivíduo é baseada de acordo com uma função de avaliação. Desta forma, em futuras gerações, indivíduos representando melhores soluções vão surgindo.

Os mais importantes métodos associados aos *Algoritmos Evolucionários* são [Ang 2000]: Os *Algoritmos Genéticos*, *Programação Genética*, *Estratégias Evolucionárias*, *Programação Evolucionária* e *Sistemas de Classificação da Aprendizagem*. Associados também aos *Algoritmos Evolucionários* estão algumas heurísticas de pesquisa, nomeadamente: *Simulated Annealing* [AK1989], *Hill Climbing* [OO1994], *Particle Swarm* [Ken2006], *Ant Systems* [DS2004] e *Tabu Search* [Glo1996]. No entanto, os *Algoritmos Genéticos* e a *Programação Genética* [Koz1992] [Koz1994] são dois diferentes tipos de *Algoritmos Evolucionários*. A *Programação Genética* utilizada neste trabalho segue uma aproximação diferente dos *Algoritmos Genéticos*. Ambos simulam o processo de aprendizagem da evolução biológica usando a selecção, cruzamento e mutação, mas em vez de codificar *strings* que representam uma solução para um problema particular, a *Programação Genética* gera programas executáveis, apresentando graficamente uma representação em forma de árvore. Ao longo dos anos têm sido apresentados vários trabalhos interligando os *Algoritmos Evolucionários* com a

Lógica (e *Fuzzy Logic*) aplicados em particular nas vertentes dos sistemas de aprendizagem [Div2001] [RCN2003] [Div2004] [SE2009]. Contudo, estes trabalhos apresentam algumas limitações no que se refere à avaliação do conhecimento em termos da sua quantificação durante o processo evolutivo na obtenção de respostas para a resolução de problemas.

As *Redes Neuronais Artificiais* [Hay1998] correspondem a um paradigma computacional inspirado na forma como o cérebro humano processa a informação. Estes modelos conexionistas têm sido aplicados com sucesso em várias áreas do conhecimento [WRL1994], assim como na sua interligação com os sistemas evolutivos [SKS 2007], em especial na parametrização dos dados de entrada e dos pesos das conexões da estrutura das redes. O estudo da interligação entre mecanismos de representação simbólica e neuronal é considerado uma tarefa difícil, ainda que interessante pelas capacidades de aprendizagem das *Redes Neuronais Artificiais* e pela natureza declarativa e capacidades de raciocínio dos sistemas simbólicos.

Vários investigadores debruçaram-se sobre o estudo da interligação entre sistemas conexionistas e formalismos associados às lógicas, em particular na representação do cálculo proposicional e tarefas de raciocínio [BHH2004] [HP2007], assim como usando a lógica de primeira ordem [Pin1991] [BS2001] [dA$^+$2001] [dA$^+$2002] [BAH2005] [dA$^+$2007]. Em geral, estes estudos tentam treinar as redes no sentido de extrair regras lógicas [BH$^+$2007] [BHH2008] que proporcionará mecanismos de raciocínio sobre o conhecimento adquirido.

Neste trabalho de doutoramento, o sistema de raciocinio é baseado nos formalismos associados à representação simbólica, tratando e quantificando o conhecimento adquirido em ambientes em que a informação é incompleta. Este sistema irá permitir tirar partido das potencialidades das *Redes Neuronais evolucionárias* [Yao1999] [Nit2008] [FDM2008], em especial no que se refere à preparação da informação (incompleta) para treinar a rede. Desta forma neste nosso estudo, a interligação entre os sistemas conexionista e simbólico é realizado por um objectivo secundário no que se refere ao tratamento prévio do conhecimento imperfeito, dando entrada na rede neuronal evolucionária de conhecimento quantificável e conhecido para situações em que a informação é contraditória ou incompleta.

Deste modo pretendemos dar um salto qualitativo na Inteligência Artificial, combinando os três paradigmas para modelar sistemas de raciocínio. Para isso, em termos formais, utilizamos os fundamentos da *Lógica Matemática* (em especial a simbólica) e

a *Programação em Lógica* (materializada através da *Programação em Lógica Estendida*) para representar o conhecimento e disponibilizar métodos de raciocínio. Port outro lado, como o processo de inferência do raciocínio é tipicamente um processo dinâmico, inúmeras soluções para um determinado problema podem ser alcançadas. Aliado às potencialidades dos *Algoritmos Evolucionários* durante o processo evolutivo na interpretação (ou resolução) do problema em causa, pretende-se seleccionar as melhores soluções (ou cenários). Aqui, as soluções candidatas são programas lógicos ou teorias e a optimização baseada na *qualidade da informação* associada a esses programas lógicos.

No entanto, neste trabalho não iremos estar concentrados com os fundamentos teóricos associados à *Lógica*, mas antes com as questões práticas que torna útil a construção de aplicações complexas para modelar o raciocínio e consequentemente ser utilizado na resolução de problemas. No seguimento dos resultados dos trabalho realizados nesta área da Inteligência Artificial [AN+2006] [NM+2007] [LC+2008] [RM+2010c] [LC+2008] e, considerando o seu carácter inovador no que se refere à criação de um modelo computacional englobando os três paradigmas (simbólico, conexionista e evolutivo), a sua aplicabilidade é abrangente a várias áreas. No entanto, pretendemos demonstrar e avaliar a sua aplicabilidade nas áreas da medicina, ambientes industriais e processos de tomada à decisão.

1.1 Contexto e Motivação

Este trabalho estuda a fusão da teoria conceptual entre os sistemas simbólicos (e.g. *Programação em Lógica*), conexionistas (e.g. *Redes Neuronais Artificiais*) e evolutivos (e.g. Algoritmos *Evolucionários*), considerando-se como uma *framework* para implementar métodos de convergência dentro de programas criativos em termos da *Programação em Lógica*. Tirando partido das vantagens associadas às metodologias de raciocínio representadas através de funções lógico matemáticas e das potencialidades criativas e dinâmicas dos sistemas simbólicos e evolutivos, este trabalho centra-se na criação de um sistema formal que especifica e modela um sistema de raciocínio evolutivo para a resolução de problemas quando representado através de fórmulas lógicas mapeadas através da *Programação em Lógica Estendida* [NM+1997].

As *Redes Neuronais Artificiais* serão utilizadas pela sua simplicidade de rep-

resentar categorias e por permitirem optimizar os sistemas simbólicos, mediante a evolução de uma topologia da rede. A interligação entre a *Programação em Lógica Estendida* e as *Redes Neuronais Artificiais* servirá para fazer a representação hiperespacial geométrica, ou seja, representar o *universo do discurso* e atribuir valores de entrada para a rede. Com a fusão deste tipo de sistemas (simbólicos, conexionistas e evolutivos), pretendemos estudar em que medida os sistemas conexionistas e evolutivos trazem para o poder descritivo da *Programação em Lógica Matemática* a quantificação da descrição do *universo do discurso*, o qual é expresso em teorias lógico matemáticas como forma de um sistema simbólico.

Conforme apresentamos, ao longo dos anos a utilização dos paradigmas simbólico, evolutivo e conexionista tem-se demonstrado adequados na formalização e resolução de problemas complexos. Estas observações sugerem que estes paradigmas possam ser aplicados de forma complementar na área da representação do conhecimento e mecanismos de raciocínio, em particular, em cenários em que a informação é imprecisa. Mais importante ainda, estes paradigmas e a sua interligação indicam que um sistema capaz de incorporar as vantagens de ambas as abordagens podem tirar partido dos diferentes benefícios dessas mesmas abordagens. Isto motiva o propósito desta tese de doutoramento que se centra no desenvolvimento de um sistema computacional baseado nos paradigmas evolutivo e conexionista que incorpora metodologias de representação do conhecimento e formas de raciocínio associadas ao paradigma simbólico, expresso aqui pela *Programação em Lógica Estendida*, através de teorias ou programas lógicos.

Em suma, a motivação do desenvolvimento deste trabalho centra-se no estudo, especificação e desenvolvimento de um sistema formal que permita a modelação de sistemas dinâmicos para a resolução de problemas através de uma composição de funções lógico matemáticas. Pretende-se estudar em que medida os sistemas simbólicos, juntamente com os sistemas conexionistas trazem para o poder descritivo da programação em *Lógica Matemática*, a quantificação da descrição do *universo do discurso*.

1.2 Questões e Hipótese de Trabalho

Vários estudos têm sido apresentados sobre a aplicabilidade do paradigma simbólico, conexionista e evolutivo. Estes estudos têm demonstrado resultados promissores na modelação de problemas complexos, dispersos e de comportamento incerto, permitindo contornar várias limitações das aproximações tradicionais em lidar com grandes volumes de dados em ambientes em que o conhecimento é contraditório, imperfeito ou mesmo desconhecido. Contudo, existe uma limitação no estudo do relacionamento entre estas técnicas e os formalismos de representação do conhecimento e de mecanismo de raciocínio.

Neste contexto, e considerando o sucesso das técnicas destes paradigmas, a principal questão e desafio deste trabalho é estudar até que ponto é possível aliar as vantagens dos paradigmas na criação de sistemas inteligentes para a resolução de problemas. Partimos como hipótese de trabalho que será possível obter as melhores teorias para responder a um determinado problema, ou por outras palavras, questionar se será possível obter a melhor quantificação do *universo do discurso* e consequentemente optimizar as melhores funções lógico matemáticas.

1.3 Paradigmas Utilizados

Neste trabalho, são utilizados o paradigma simbólico, conexionista e evolucionário. Tradicionalmente, para a *Representação do Conhecimento* recorre-se muitas vezes à utilização de uma linguagem *Lógica* [RN1995]. As linguagens lógicas permitem uma descrição formal e não ambígua de factos, que podem ser verificados e validados formalmente [Lig1997]. O paradigma da *Programação em Lógica* baseia-se na utilização da lógica como uma linguagem de programação declarativa, onde se descreve um problema em termos de axiomas lógicos. No entanto, cenários de informação incompleta ocorrem normalmente nas Bases de Conhecimento e nos processos de tomada à decisão [Nev1984] [TG1993].

Por sua vez, a *Programação em Lógica* é baseada nalguns pressupostos (*e.g.*, *Mundo Fechado*, que significa que todo o conhecimento de determinados factos é conhecido) que impõem limitações ao tipo de processamento necessário para tratar informação incompleta. Os sistemas reais podem no entanto, beneficiar largamente de abordagens que evitem estas limitações. Ao adicionar capacidade para a rep-

resentação e raciocínio sobre informação incompleta a um sistema, a sua base de conhecimento passa a poder descrever o mundo real de forma muito mais correcta. A *Programação em Lógica Estendida* corresponde a uma extensão da *Programação em Lógica*, permitindo a inclusão de informação negativa explícita, sendo que, por defeito, todo o conhecimento é aberto, ou seja, é possível deduzir se determinado facto é verdadeiro, falso ou desconhecido. Desta forma o paradigma simbólico é materializado através da *Programação em Lógica* reflectindo os fundamentos da *Lógica Matemática*.

Vários trabalhos têm sido apresentados na área da neurociência, em especial, na interligação dos sistemas simbólico e conexionista. Em geral, o seu objectivo é extrair regras lógicas e por conseguinte, mecanismos de raciocínio a partir do sistema de aprendizagem associado às *Redes Neuronais Artificiais*. De uma forma indirecta, este trabalho permite preparar a informação para alimentar a rede quando aplicada a cenários em que a informação é imperfeita. Isto porque, a representação da informação incompleta é materializada através da *Programação em Lógica Estendida* e quantificada pela *Qualidade da Informação* [AN+2006] [NM+2007] associada a cada programa lógico e assim, submeter à rede informação conhecida (e quantificada) onde na fonte de dados essa mesma informação tinha caracteristicas de imprecisão.

Por outro lado, os sistemas evolutivos, em particular, a utilização da *Programação Genética* reflectem a utilização dos principios da teoria *Darwinista*. O seu objectivo é ser utilizado pela sua potencialidade de criar novas possíveis soluções (programas lógicos), no sentido de convergir para uma solução do problema. Em suma, a ligação entre os três paradigmas permitirá a criação de sistemas evolutivos esquematizados por estruturas em rede evolutivas [FDM2008] que materializem a representação do conhecimento e formas de raciocínio simbólico, o que nos é dado aqui pela *Programação em Lógica Estendida* [NM+1997].

1.4 Objectivos, Delimitação e Pressupostos

1.4.1 Objectivos

De uma maneira geral, o objectivo principal deste trabalho centra-se no desenvolvimento de sistemas virtuais evolutivos que ajudem na optimização de cenários reais. Pretende-se combinar os sistemas conexionistas e sistemas simbólicos da *Inteligência*

Artificial a fim de optimizar o *universo do discurso*, representado por funções da teoria lógico matemática, ou seja, aproveitar as técnicas da *Inteligência Artificial* para optimizar funç ões lógico matemáticas que trazem um poder descritivo para a *Programação em Lógica Estendida* [NM+1997] a fim de obter a melhor quantificação do *universo do discurso*.

Os objectivos específicos com que se pretenderam atingir em termos de investigação foram:

1. Criar um modelo formal para raciocinar com informação incompleta e quantificar todas as soluções usando a medida da *Qualidade da Informação* [AN+2006] [NM+2007] associada aos programas lógicos (ou teorias);

2. Disponibilizar uma *framework* usando o *Paradigma da Computação Evolucionária* , que permita criar programas, empregando metodologias para a resolução de problemas que beneficiam dos *abducibles*[3] e maximizam o *universo do discurso*;

3. Criar extensões à *framework* para serem usadas em várias áreas do conhecimento, como por exemplo, na área médica, em ambientes industriais e em ambientes de tomada à decisão;

4. Criar um sistema interactivo apresentando uma interface simples e amigável para os seus utilizadores, no sentido de demonstrar os conceitos abordados neste trabalho de doutoramento.

1.4.2 Delimitação

Dada a amplitude dos paradigmas que se utilizam neste trabalho, torna-se necessário definir alguma delimitação em cada um deles:

- Em relação ao paradigma simbólico, este trabalho baseia-se na *Programação em Lógica Estendida* [NM+1997] para representar o conhecimento e formas de raciocínio. Este paradigma é caracterizado pela representação de factos, regras

[3] A **abdução**, ou hipóteses (ou *abducibles*) [Mag2001] ocorre quando nos deparamos com determinadas circunstâncias capazes de serem explicadas pela suposição de que se trata de um caso particular de uma regra geral, adoptando-se em função disso a suposição.

e predicados, que fazem a correspondência com a *Lógica Matemática* através da esquematização ou representação de funções lógico matemáticas definindo o conhecimento. Como mecanismo de raciocínio é utilizada a dedução lógica, sendo o conhecimento validado e satisfeito por mecanismos de prova (*i.e. a teoria da prova*) [Nev1984] [Kow1995].

- A delimitação do paradigma conexionista centra-se na utilização dos conceitos associados às *Redes Neuronais Artificiais*, usando processos evolucionários da criação da topologia da rede (*Redes Evolucionárias*) [Yao1999] [Abra2004] [Nit 2008]. Desta forma tiramos partido em utilizar as suas vantagens como esquemático arquitectural no sistema evolutivo criado. De modo lateral ao trabalho desta tese, realizamos duas contribuições ao âmbito das *Redes Neuronais Artificiais*.No primeiro estudo [FR$^+$2010a] propomos um método denominado *Direct Kernel Perceptron* para calcular directamente os pesos de um *perceptron* simples usando a expressão *closed-form* a qual não requer qualquer fase de treino. O segundo estudo [FR$^+$2010b] propõe uma expressão analítica *closed-form*, no sentido de calcular sem iterações, os pesos do *Parallel Perceptron* para tarefas de classificação. Neste trabalho de doutoramento o resultado da criação do sistema evolutivo é a disponibilização das melhores teorias ou programas lógicos em ambientes contemplados com informação incompleta. Desta forma, toda a informação (mesmo a incompleta) é quantificada. Neste sentido, é possível alimentar uma *Rede Neuronal Artificial* com informação conhecida a qual à partida não o era. Finalmente numa terceira contribuíção, o terceiro estudo [NR$^+$2011] já no âmbito da presente tese, aplica os seus resultados à classificação das características do pavimento (asfalto) das estradas.

- O paradigma evolutivo materializado através da *Programação Genética* será utilizado para a geração dos vários níveis da rede evolucionária, sendo aplicados os operadores genéticos para a criaç˜o de novos programas lógicos, que serão avaliados seleccionando-se aqueles com melhor qualidade da informação [NA$^+$2006] [NM$^+$2007].

Por outro lado, tipicamente os problemas de optimização de funções [Rao2000] [GC$^+$ 2007] centram-se na sua minimização ou maximização, podendo ser utilizadas vários tipos de técnicas para o efeito consoante o campo de aplicabilidade, como as *Redes*

Neuronais Artificiais, os *Algoritmos Genéticos,* as técnicas *Fuzzy*, entre outras. Neste trabalho de doutoramento, quando nos referimos à optimização de funções, referimo-nos a funções lógico matemáticas que modelam o *universo do discurso*, pretendendo-se seleccionar as que oferecem uma melhor qualidade da informação.

1.4.3 Pressupostos

Globalmente, este trabalho pretende estudar e aprofundar os conceitos associados aos paradigmas acima referidos, seguindo uma determinada linha de investigação [NM$^+$2007], tentando contribuir com melhoramentos para o conhecimento e para a sociedade. Neste sentido, torna-se importante enunciar os pressupostos com que este trabalho assenta:

- Em relação ao paradigma simbólico, partimos do pressuposto que todos os fundamentos associados à *Lógica*, *Lógica Matemática* e *Programação em Lógica* estão validados e consolidados para a representação do conhecimento e para a resolução de problemas mediante mecanismos de raciocínio. Utilizamos a *Programação em Lógica Estendida* [Nev1984] como formalismo de representação do conhecimento e mecanismo de raciocínio (teoria da prova), assim como o conceito da *Qualidade da Informação* [AN$^+$2006] associada às teorias ou programas lógicos, usado em várias áreas de aplicação [MA$^+$2006] [LC$^+$2008] [MM$^+$2009] [RN$^+$2009] [LN$^+$2009] [MA$^+$2010] [NA$^+$2010] [NS$^+$2010].

- Em relação às *Redes Neuronais Artificiais*, utilizaremos os conceitos associados às *Redes Evolucionárias* [Yao1999] [Abra2002] [Abra2004] para criar dinâmicamente as arquitecturas de rede.

- Tendo em consideração que a representação do conhecimento e o mecanismo de raciocínio é materializado através de programas lógicos, o pressuposto associado ao paradigma evolutivo centra-se na aceitação dos fundamentos associados à *Programação Genética*. Partindo dos conceitos e mecanismos computacionais dos Algoritmos Evolucionários, vamos tirar partido das suas vantagens para atingir o objectivo deste trabalho.

1.5 Metodologia

O método experimental que se utilizou para a realização deste trabalho iniciou-se pela identificação do problema, de modo a formalizar uma hipótese de trabalho sobre a qual o mesmo foi desenvolvido. Subsequentemente, a informação foi recompilada, organizada e analisada continuadamente, construindo uma proposta para resolver o problema identificado (secção 1.2). Finalmente, elaboraram-se as conclusões baseadas nos resultados obtidos durante a investigação. O trabalho foi subdividido em quatro fases:

1. *Revisão da literatura existente sobre os paradigmas utilizados*, i.e. tornou-se essencial apresentar a evolução dos conceitos e metodologias que assentaram neste estudo, a revisão e resumo da bibliografia, assim como dos mais recentes trabalhos de investigação na área delimitados por este trabalho de doutoramento, nomeadamente: *Lógica*, *Lógica Matemática* como mecanismo de representação do conhecimento e formas de raciocínio simbólico, *Algoritmos e Computação Evolucionária*, *Redes Neuronais Artificiais*, assim como apresentar a evolução histórica de alguns estudos com propostas de interligação entre estes paradigmas.

2. *Concepção de um sistema protótipo* – após uma análise profunda dos conhecimentos adquiridos nas fases anteriores, partiu-se para a concepção e modelação de um protótipo para a criação de um sistema virtual evolutivo que permitisse representar qualquer cenário real materializado através de funções lógico matemáticas especificadas em *Programação em Lógica Estendida* e, proceder à optimização desses cenários, medindo em cada iteração a *Qualidade da Informação* transportada por cada solução. Para modelar a evolução do sistema foram utilizados para o processamento, os *Algoritmos Evolucionários* (em especial a *Programação Genética*) e as *Redes Neuronais Artificiais Evolucionárias* como esquemático arquitectural.

3. *Implementação do protótipo e teste* – esta fase pretendeu passar à prática as ideias conceptualizadas na fase anterior (capítulo 5). O sistema permitirá estar disponível em várias plataformas computacionais, de modo a puder ser utilizado, avaliado e melhorado futuramente. Com base no sistema protótipo será avaliado o desempenho do mesmo em diversos cenários de utilização. Por outro lado, pretendeu-

se efectuar a apresentação dos resultados em congressos internacionais [RN+2009] [RM+2010a] [RM+2010b] [RM+2010c] [FR+2010a] [FR+2010b] [NS+2010], bem como submeter artigos a revistas científicas [FR+2010] [NR+2011].

4. *Elaboração da tese* – finalmente, passou-se á fase de escrita da dissertação com base nas observações, experiências e resultados obtidos nas fases anteriores.

1.6 Estrutura e Resumo da Dissertação

Este trabalho de doutoramento está dividido em duas partes: a primeira (capítulos 2 e 3) referente à apresentação dos fundamentos teóricos sobre os quais se assentou o estudo e, uma segunda parte (capítulos 4 e 5), referente à formalização e operacionalização conceptual do modelo evolutivo.

Cada um dos capítulos apresenta um resumo do seu conteúdo, no sentido de fornecer uma noção geral do trabalho antes de entrar em detalhe na sua descrição. Este trabalho trata do tema da criação de um sistema formal que permite a modelação de sistemas em que o modelo é dado pela composição de funções lógico matemáticas. Para criar este sistema formal utilizamos os sistemas simbólicos, conexionistas e evolutivos. Por este facto, torna-se relevante no nosso entender, apresentar antes do sistema formal (capítulo quatro) alguns conceitos teóricos associados.

O **capítulo dois** apresenta uma breve descrição do principal fundamento teórico no qual este trabalho assenta, nomeadamente a *Lógica Matemática* e a *Programação em Lógica*. Iniciamos com uma breve introdução à *Lógica* como ciência associada à investigação sistemática da validade (ou invalidade) de esquemas de inferência para a resolução de problemas. Abordamos a forma de representação de conhecimento, apresentando os dois tipos de lógicas (clássicas e não clássicas), abordando de uma forma abstracta e genérica as características e formas diferenciadoras de cada uma. Apresentamos os fundamentos gerais para a utilização da *Lógica* na *Representação do Conhecimento* nomeadamente (fórmulas lógicas, semântica das fórmulas, modelos e consequência lógica e, inferência lógica).

No **capítulo três** apresentamos a abordagem aos sistemas evolutivos e conexionistas, em particular a *Programação Genética* e as *Redes Neuronais Artificiais*. Além de descrevermos as características destes sistemas, apresentamos também o trabalho

relacionado na área de aplicabilidade destes sistemas em relação à sua interligação com a *Programação em Lógica*.

No **capítulo quatro**, descrevemos os sistemas de raciocínio e as abordagens à representação do conhecimento, a teoria dos modelos e a teoria da prova, apresentando a representação do conhecimento através de *Programação em Lógica Estendida*, no sentido de contemplar informação incompleta. Nesta abordagem, apresentamos as características associadas ao raciocínio não monótono [EST2005], temporal, processo de actualização da informação, pré-condições, preferências e outros condicionalismos. Apresentamos também o conceito da *Qualidade da Informação* [AN+2006] [NM+2007] como forma de quantificação da informação que uma determinada base de conhecimento transporta num determinado momento.

Neste e no **quinto capítulo**, apresentamos o modelo proposto quer em termos da sua especificação formal, quer em termos da sua operacionalização, onde descrevemos formalmente o processo de fusão dos sistemas simbólicos, evolutivos e conexionistas.

No **capítulo seis** apresentamos as conclusões e trabalho futuro, descrevendo uma síntese da tese, as contribuições, conclusões e originalidades assim como as suas limitações e o trabalho futuro.

1.7 Notação e Terminologia

A notação utilizada ao longo do documento segue a convenção apresentada a seguir:

• *Texto em itálico* utilizado para dar ênfase a um determinado termo ou conceito; exemplo: "Vários esforços têm sido realizados para interligar a *Lógica Matemática* com [...]", ou por exemplo "A equivalência das *sentenças* na *lógica de predicados* é [...]";

• *Texto em negrito* – utilizado para realçar um conceito/palavra no meio de um parágrafo; exemplo: "A **lógica de primeira ordem** utiliza [...]";

• *Parágrafo identado em itálico* – citações bibliográficas;

•*Parágrafo com espaçamento simples em itálico* – para os extractos de código ou

extensões de predicados lógicos; exemplo: *"filho_de('João', 'Carlos")*;

• *Letras em caligrafia e letras do alfabeto Grego* – utilizadas para referenciar elementos (*e.g.*, conjuntos, tuplos) nas descrições formais; exemplo "seja A o alfabeto [...]"; exemplo: "A teoria ordenada $\langle \mathrm{T}, < \rangle$ representa [...]".

Ao longo do texto são utilizadas várias letras do alfabeto grego em algumas fórmulas e descrições formais, pelo que em seguida se apresenta uma relação desses símbolos no sentido de facilitar a sua leitura:

L	Linguagem lógica
T	Teoria, conjunto de Teoremas, conjunto de Termos ou conjunto de sentenças
H	Conjunto de hipóteses
$\neg$	Negação explícita ou negação forte
$\wedge$	"e"
$\vee$	"ou"
$\rightarrow$	Implicação
$\leftrightarrow$	Equivalência
$\supset$	Implicação
$\equiv$	Equivalência
F	Fórmula
$\perp$	Cláusula vazia
not	Negação por falha na prova, podendo ser também identificada por "nao"
:	"tal que"
A	Alfabeto
$\forall$	Quantificador Universal, significa "Para todo...."
$\exists$	Quantificador Existencial, significa "Existe um...."
Π	Operador de Projecção, no contexto de bases de dados relacionais.
∞	Operador de Junção, no contexto de bases de dados relacionais.

Esta página foi propositadamente deixada em branco.

"Logic Programming is the use of logic to represent programs and deduction to execute programs in logical form. To this end, many different forms of logic and many varieties of deduction have been investigated..."

[Kowalski in MIT Encyclopedia of Cognitive Science, 1999]

Capítulo 2

Lógica e Programação em Lógica

A *Lógica* e os *Programas Lógicos* emergiram como um formalismo atractivo de representar o conhecimento, assim como abordagem para a resolução de problemas. Desde sempre que os fundamentos da *Lógica Matemática* estão intimamente ligados à teoria do raciocínio. Neste capítulo apresentamos uma breve introdução à utilização da lógica, abordando de uma forma geral os seus vários tipos (clássicas e não clássicas), iniciando com uma abordagem aos fundamentos da lógica, nomeadamente, fórmulas lógicas, semântica das fórmulas lógicas e a inferência lógica. De seguida e tendo em consideração que o conteúdo matemático da lógica se organiza em duas teorias, apresentamos de uma forma geral os conceitos associados à teoria dos modelos e à teoria da prova. Na secção 2.4 abordamos a lógica e a teoria do raciocínio, dando particular atenção à dedução, indução e abdução, assim como aos fundamentos do raciocínio não monótono [EST2005]. Na secção 2.5 apresentamos a *Programação em Lógica*, abordando as suas características, as suas limitações, as suas extensões, assim como os sistemas baseados na lógica desenvolvidos ao longo das últimas décadas.

2.1 Introdução Histórica

Desde o aparecimento da humanidade que a lógica teve a sua origem, estando intimamente associada à área da filosofia. Concentrados na crítica da análise lógica dos argumentos filosóficos do pensamento humano, vários filósofos desenvolveram descrições formais sobre a linguagem natural, defendendo que o cerne do "normal" raciocínio poderia ser capturado pela lógica, se se conseguisse encontrar um método certo para

transcrever a linguagem ordinária do ser humano em lógica (***philosofical logic***). No entanto, foi com *Aristóteles* (384 DC)[McK1941] que a lógica foi criada como disciplina, passando assim a estar associada também à *Matemática*. *Aristóteles* introduz o termo "**lógica**" (inicialmente designada por ***syllogistic logic***) que corresponde à análise de *sentenças* (ou "*judgments*") em proposições, acrescidas de expressões de inferência e numa conclusão, no sentido de traduzir as actividades intelectuais dos seres humanos em formalismos matemáticos.

A literatura elaborada ao longo da história, apresenta que o estudo da *Lógica Matemática* teve os seus primeiros passos no século 19. Até então, o conceito de lógica correspondia ao estudo dos argumentos da linguagem natural (***informal logic***). No final do século 19, vários matemáticos e filósofos apresentaram as primeiras notações associadas ao conceito da lógica e a sua correspondência em termos matemáticos. Os seus objectivos centraram-se na apresentação de símbolos computáveis para as *palavras* e *statememements*[1], sendo estas notações designadas por ***formal logic*** ou ***simbolic logic***. Foi o caso da *Álgebra* de *Boole* (*George Boole*, 1815-1864), da *Cantor's set theory* (*Georg Cantor*, 1870) e a *Frege's formal logic* (*Gottlob Frege*, 1848-1864). Nesta altura a novidade dos estudos causou uma grande expectativa e controvérsia. Contudo, o arranque do estudo da lógica foi impulsionado pelo trabalho de *Frege*.

Frege apresenta um conceito inovador para a altura denominado por ***propositional logic*** *(baseada no cálculo proposicional)*, em que utiliza letras para corresponder a simples *statements*, os quais eram interligados a um conjunto restrito de símbolos (*not, and, or, if,* and *if and only if).* Na *Lógica Matemática*, o *cálculo proposicional* ou *Lógica* (também designada por "*sentential calculus*") corresponde a um sistema formal em que as fórmulas de uma linguagem formal podem ser interpretadas como uma representação de proposições. Por conseguinte, um sistema de regras de inferência e axiomas permite que certas fórmulas sejam derivadas (denominadas por "*teoremas*"), podendo ser interpretadas como proposições verdadeiras. Neste sentido, uma série de fórmulas que são constituídas dentro de um sistema deste tipo são designadas por "*derivação*", em que a última fórmula da série é designada por *teorema*, cuja derivação poderá ser interpretada como prova de verdade das proposições representadas pelo *teorema*. No entanto, o aperfeiçoamento do estudo da lógica foi

[1] *Statements* ou *Sentenças* na linguagem natural consiste em palavras onde os objectos da descrição do universo são representadas como nomes.

mais contributivo no século 20, após a apresentação do paradoxo de *Russel* (*Bertrand Russel*, 1872-1970), o qual encontrou uma inconsistência na inovadora lógica proposicional apresentada por *Frege*. Esta limitação centrava-se na possibilidade de se criar um conjunto, contendo todos os conjuntos que não contém a si mesmo como membro. O problema é que, se este conjunto se contém a si mesmo, então ele não se contém a si mesmo e vice-versa.

Tendo em consideração esta limitação e o trabalho desenvolvido por *Frege*, *Bertrand Russel* e *Alfred Whitehead* apresentam o trabalho "*Principia Mathematica*", uma série de três volumes onde estabelecem a lógica como fundamentação da matemática, o qual engloba os fundamentos matemáticos em axiomas da teoria de conjuntos e da lógica. Este estudo, apresenta uma tentativa de derivar todas as verdades matemáticas a partir de um conjunto bem definido de axiomas e regras de inferência para a lógica simbólica. Foi a partir deste trabalho que o estudo da lógica teve maiores avanços no século 20, permitindo explorar novas potencialidades e discutir limitações às anteriores abordagens. *Jan Lukasiewicz* (1917) apresenta o primeiro conceito de *lógica multi-valor*, em que um *statement* poderá assumir mais do que o valor verdadeiro ou falso.

Com a introdução no conceito da *Lógica* de mais um possível valor, a aplicabilidade da lógica à área da computação entrou numa nova era que correspondeu à passagem da lógica clássica para a lógica não clássica. Com base na introdução de mais um valor, além do verdadeiro e falso, vários estudos foram apresentados nos anos 60, sendo introduzidos o conceito de ***multi-valued logic*** [Loy2004] nas quais se baseiam, a *fuzzy logic* [Zad2001], *quantum logic* [Pir1976], entre outras.

Tendo como base os objectivos de que tudo se poderia derivar formalmente em termos de um conjunto de axiomas, vários investigadores questionaram sobre a possibilidade de usar a lógica para derivar qualquer *statement* verdadeiro acerca da matemática a partir de um conjunto de axiomas, sem derivar falsos *statements*. Estas questões levaram ao aparecimento do conceito de *consistência*[2] (*consistent*) e *completude*[3], no sentido de validar estas questões de derivação de *statements*. Neste sentido em 1931, *Kurt Godel* apresenta a *Incompleteness Theorem* [Hei1971]. Este teorema

[2] *consistência* – significa que nenhum teorema do sistema se contradiz um ao outro.

[3] *completude* – se um teorema é verdadeiro então pode ser provado.

apresenta que dado um conjunto de axiomas para um sistema matemático, existe sempre uma sentença em que nem ela nem a sua negação podem ser derivadas a partir de axiomas usando a teoria da prova, o que implica que existem verdades no sistema para os quais não se encontra uma prova. Demonstra também que qualquer intenção de reduzir a matemática a um sistema consistente de axiomas produz o mesmo resultado, ou por outras palavras, este investigador (considerado um dos grandes matemáticos do século 20), apresentou que qualquer formalização aritmética deve ser *incompleta*[4] (i.e. *incomplete* [Hei1971]), ou seja, que há *afirmações*[5] [Res2008] que não podem ser demonstradas nem refutadas. Neste sentido, uma característica da fundamentação matemática é que vários problemas a resolver podem ser expressos através de teorias. Desta forma, a resolução de um determinado problema é equivalente a questionar se novas hipóteses são consistentes com a teoria existente.

A ***Lógica Matemática*** corresponde a uma parte da matemática que envolve o entendimento de uma linguagem, semânticas, sintaxes, prova, modelos, entre outros formalismos. Todos os estudos associados à ligação da *Lógica* com a *Matemática* geraram muita discussão e controvérsia. Surge com naturalidade a passagem da lógica simbólica, a qual se centrava no estudo de símbolos abstractos para capturar características da inferência lógica, para uma nova definição de *Lógica Matemática*.

Neste sentido, os estudos da *Lógica Matemática* passaram a centrar-se em duas áreas de investigação. A primeira corresponde à aplicação de técnicas da lógica formal à matemática e raciocínio matemático e a segunda, na aplicação de técnicas matemáticas à representação e análise da lógica formal. A *Lógica Matemática* passou a ser uma extensão da lógica simbólica para outras áreas de investigação, em particular, para o estudo da teoria dos modelos, teoria da prova, teoria dos conjuntos e teoria da recursão (conhecida também como "teoria da computabilidade" ou "teoria das funções recursivas"), sendo estas quatro teorias vistas como pilares fundamentais da *Lógica Matemática*. A noção de conjunto e o estudo dos conjuntos (teoria dos conjuntos [Jec1978] [Jec2006]), essencial em muitos campos da matemática, foram introduzidos por *Cantor* nos finais do século 19. A teoria de *Cantor*, um tanto intuitiva, foi posteriormente tratada e organizada de forma axiomática.

[4] Uma teoria *incompleta* em oposição a uma teoria completa [Hei1971], na qual qualquer teoria capaz de expressar aritmética elementar não pode ser em simultâneo consistente (não contém contradições) e completa (i.e. corresponde ao máximo conjunto consistente de sentenças).

[5] *afirmações* ou *assertions* na terminologia inglesa e nos fundamentos da lógica.

A *teoria da recursão* [KS1995] corresponde a uma metodologia aplicada na matemática e na ciência, que tem como objectivo a definição de funções recursivas para a resolução de problemas. A teoria dos modelos tem como objectivo o estudo de linguagens formais, pelas suas interpretações e pelos tipos de classificações, enquanto que a teoria da prova tem como objectivo representar o raciocínio matemático através de sistemas formais dedutivos.

Nos últimos anos a *Lógica Matemática* evoluiu de uma fase de pura actividade teórica para se tornar numa ferramenta de uso prático, influenciando decisivamente a evolução das ciências da computação. A lógica passou a ser estudada para entender o raciocínio humano quer em termos computacionais, quer em termos lógicos. Neste sentido, uma das definições de *Lógica* é a da análise de métodos de raciocínio, na qual é possível expressar o raciocínio humano usando a *Lógica* como notação matemática através de formalismos associados ao raciocínio indutivo e dedutivo [Har2002]. Trata-se de uma ciência que está associada com a investigação sistemática da validade e invalidade dos esquemas de inferência, e é construída como um conjunto de técnicas para resolver uma classe de problemas, pelo menos parcialmente, através da interpretação e da manipulação de um cálculo formal (associado com a *Lógica*).

Em suma, a ***Lógica*** corresponde ao estudo da validade dos argumentos lógicos válidos ou inválidos. Um argumento lógico corresponde a um conjunto de uma ou mais premissas seguidas de uma conclusão que está interligada por um ou mais *statements* intermédios. Por outro lado, as *premissas* e a conclusão são sempre *statements*, ou seja, *sentenças* que apresentam informação que pode ser verdadeira ou falsa. Por sua vez, um argumento é válido se todas as *premissas* forem verdadeiras, então a conclusão tem de ser verdadeira.

2.2 Tipos de Lógicas

Desde os anos 50 que a *Lógica Matemática* evoluiu de uma fase puramente teórica para se tornar numa ferramenta de uso prático, influenciada pela evolução dos sistemas de computação. As teorias baseadas em lógica são classificadas em lógicas clássicas e lógicas não clássicas, sendo estas últimas uma extensão de algumas limitações das primeiras. As ***lógicas clássicas*** (*proposicional* ou *predicativa*), sempre assumiram um papel preponderante no desenvolvimento da lógica como mecanismo computa-

cional. A forma rigorosa da sua definição sintáctica e semântica permite a eliminação de muitos aspectos de ambiguidade na formalização de modelos associados aos processos de representação do conhecimento e raciocínio.

Outros tipos de lógicas chamadas de ***lógicas não clássicas*** (modal proposicional, modal de primeira ordem, raciocínio não monótono [EST2005], raciocínio defeito, não monótona, intuicionista, auto epistémica [Gel1987], temporal e linear), permitem abordar o estudo do acreditar, o raciocínio por defeito ou incompleto, assim como ultrapassar as dificuldades que surgem no processamento do factor tempo. As lógicas não clássicas correspondem a extensões ou modificações das lógicas clássicas, podendo em alguns casos existir mesmo a negação ou rejeição de alguns axiomas fundamentais das lógicas clássicas ou a definição de novos operadores lógicos unários ou binários. Por outro lado, se em alguns casos algumas lógicas *não clássicas* competem directamente com as *lógicas clássicas* (por exemplo a *lógica multi-valor*, lógica "*fuzzy*", lógica intuicionista ou lógica linear), noutros casos as lógicas não clássicas tem como objectivo ser uma generalização da lógica proposicional ou predicativa (por exemplo a lógica modal, multi-modal ou temporal).

Por outro lado, extensões à lógica clássica como a *lógica multivalor*, *lógica não monótona* [Bro1980] e a *lógica temporal* fornecem uma ferramenta útil para desenhar sistemas realistas de *Inteligência Artificial*. A relação entre a lógica clássica e as extensões à lógica clássica pode ser apresentado da seguinte forma: na formulação clássica, qualquer coisa que segue a partir de um conjunto de factos, também segue a partir de um extenso conjunto de factos. Outra forma de dizer é que a lógica clássica é monótona. Além disso, o conceito de tempo é também muito importante para o raciocínio humano. A lógica clássica poderá ser estendida para incluir lógica temporal. Finalmente, no raciocínio com incerteza a lógica clássica poderá ser modificada para incorporar "*fuzzy logic*" [Zad2001], *probabilistic logic* [Sub2001] e lógica *multivalor* [Loy2004], ou outros formalismos.

Quando se pretende resolver um problema usando sistemas computacionais, a primeira tarefa a ser realizada é transcrever o problema para termos formais. Várias abordagens têm vindo a ser propostas para efectuar a representação e codificação, variando do cálculo proposicional até às lógicas de 2ª ordem. Nas secções seguintes iremos apresentar de uma forma breve e geral, os conceitos e formalismos associados à lógicas clássicas e não clássicas, não pretendendo aprofundar com detalhe a informação de cada uma delas, uma vez que a história e as referências bibliográficas

associada à lógica apresentam com mais pormenor os seus fundamentos. Pretendemos aqui, enquadrar a utilização da lógica neste trabalho de doutoramento, em especial através dos fundamentos associados à *Programação em Lógica Estendida*, nomeadamente, a *Lógica Matemática*, a *Lógica* de primeira ordem, a teoria da prova e o raciocínio não monótono [EST2005].

2.2.1 Lógicas Clássicas

A ***Lógica Proposicional*** *ou* ***Cálculo Proposicional*** [Men1997] formaliza a estrutura lógica mais elementar do discurso matemático, correspondendo a uma linguagem que permite o tratamento de expressões lógicas simples sobre a sua forma abstracta, definindo o significado dos operadores lógicos "não", "e", "ou", "se então", entre outros. Na lógica proposicional é usado o alfabeto proposicional que consiste num conjunto de proposições que são verdadeiras ou falsas. Trata-se de um sistema formal em que fórmulas de uma linguagem formal podem ser interpretadas como representação de proposições. Desta forma, um sistema de regras de inferência e *axiomas* permite que certas *fórmulas* possam ser derivadas, sendo denominadas por ***teoremas***[6], que por sua vez podem ser interpretadas como preposições verdadeiras. Neste sentido, as teorias matemáticas distinguem-se entre *axiomas* e *teoremas*.

Por outro lado, os ***axiomas*** não podem ser provados mas sim aceites como factos, enquanto que os *teoremas* podem ser aprovados a partir dos axiomas usando inferência lógica. Neste sentido, o alfabeto A de uma linguagem proposicional consiste num conjunto de operadores (ou ***conectives***) tais como: negação ($\neg$), bi-implicação ($\equiv$), implicação ($\rightarrow$), conjunção ($\wedge$), disjunção($\vee$), a pontuação (,), os parêntesis e um conjunto de símbolos proposicionais também designados por ***átomos*** (*e.g.*, X, Y, ...), que representam factos do mundo real. A formalização da lógica proposicional é extremamente simples em que a representação do mundo real é formalizada através de ***factos*** que podem ser *verdadeiros* ou *falsos*. As regras semânticas da linguagem definem o conjunto de fórmulas como sendo os próprios símbolos proposicionais ou expressões, construídas ligando tais símbolos através dos operadores. As *regras semânticas* da linguagem capturam o significado pretendido dos operadores

[6] Na matemática, um *teorema* corresponde a um sentença declarativa ou expressão da liguguagem natural (também denominada por *statement*) que foi provado, tendo como base *statements* estabelecidos anteriormente, tal como outros teoremas e axiomas [Hei1971].

e associam a cada fórmula um dos valores binários verdadeiro ou falso. Neste sentido, sendo P e Q fórmulas, a formalização das expressões é baseada no seguinte formalismo:

- $\neg P$ é verdadeiro sse P é falso;
- $(P \wedge Q)$ é verdadeiro sse P é verdadeiro e Q é verdadeiro;
- $(P \vee Q)$ é verdadeiro sse P é verdadeiro ou Q é verdadeiro;
- $(P \rightarrow Q)$ é verdadeiro sse P é falso ou Q é verdadeiro;
- $(P \equiv Q)$ é verdadeiro sse P e Q tem o mesmo valor lógico;

em que *sse* é para ser entendido como *se e somente se*.

Conforme foi referido, as lógicas clássicas, como a proposicional, sempre assumiram um papel importante no desenvolvimento da *Lógica* como formalismo computacional, além da sua esquematização que permite representar factos do mundo real através de uma linguagem expressa através de proposições, define também um conjunto de *regras de inferência* (regras de prova), que permite que uma fórmula possa ser estabelecida como *teorema*. Desta forma, estes sistemas de dedução natural descrevem regras de introdução e eliminação de *connectives*, permitindo desenhar inferências para lidar com sentenças envolvendo *connectives* dentro de um mecanismo de prova.

Contudo, a lógica proposicional apresenta algumas limitações. Não permite representar todos os tipos de afirmações (ou *assertions* na terminologia inglesa) que são utilizados nas ciências da computação e na matemática para expressar determinados tipos de relacionamentos entre proposições, como por exemplo a equivalência.

Uma outra linguagem lógica de mais poder expressivo é a ***Lógica de Predicados*** [EK1976] [Pal2008], também designada por ***Lógica Predicativa*** ou ***Cálculo de Predicados***, que permite representar o mundo através de objectos (que possuem *propriedades*), *relações* entre objectos e *funções* que manipulam esses mesmos objectos. A *Lógica Predicativa* pode ser caracterizada como um sistema formal apropriado à definição de teorias do *universo de discurso* da *Matemática*[Kow1986]. A *Lógica Predicativa* introduz o conceito de predicados com um número arbitrário de argumentos e quantificadores que permitem referir a todos ou a alguns elementos no universo

que está sobre consideração. No entanto, a *lógica predicativa* deve ser vista como uma extensão da lógica proposicional, onde são introduzidos três novos conceitos lógicos: (i) funções; (ii) predicados; e (iii) quantificadores. Em relação à lógica proposicional, a lógica predicativa apresenta a possibilidade de trabalhar com variáveis e não apenas com constantes, o que permite não ter de enumerar todos os elementos do universo de discurso.

Neste contexto, o alfabeto A de uma linguagem predicativa consiste num conjunto de operadores de negação ($\neg$), bi-implicação ($\equiv$), implicação ($\rightarrow$), conjunção ($\wedge$), disjunção ($\vee$), a pontuação (,), os parêntesis, variáveis (e.g., X, Y e Z) para denotar indivíduos arbitrários do *universo do discurso*, constantes (e.g. 0 ou 1) para denotar indivíduos específicos e símbolos, funções, relações e quantificadores, universal ($\forall$) e existencial ($\exists$). Para definir o significado de uma expressão lógica em lógica proposicional foi introduzido o conceito de interpretação, que atribui valores de verdade a todos os símbolos proposicionais de uma expressão.

Por outro lado, na *Lógica Predicativa* também se introduz o conceito de interpretação, no entanto, o processo de validação de uma expressão lógica é bastante mais complexo, uma vez que na lógica predicativa estão envolvidos outros símbolos, tais como variáveis e termos (símbolos funcionais e predicativos). O domínio de interpretação deve ser introduzido para se permitir a obtenção do valor de verdade de uma expressão lógica quantificada. Este tipo de lógica contém uma teoria da prova que consiste num conjunto de regras que descrevem a forma mecânica de derivar sentenças a partir de um conjunto de premissas, sendo as derivações denominadas por prova, enquanto que atribui significado às sentenças com respeito às estruturas, sendo que dada uma sentença ela poderá ser verdadeira ou falsa numa dada estrutura.

Face ao exposto, enquanto que na *Lógica das Proposições* concentra-se sobre as relações entre as frases de acordo com certas estruturas e com as funções de verdade, na *Lógica de Predicados* (ou de 1ª ordem) aborda-se os objectos, as suas propriedades e as relações entre estas propriedades. Por outro lado, nas lógicas de 2ª ordem temos a possibilidade de quantificar os predicados. Assim, a lógica de predicados pode ser caracterizada como um termo genérico para sistemas formais simbólicos como as lógicas de primeira e segunda ordem.

A ***lógica de primeira ordem*** utiliza o vocabulário proposicional, ou seja, os símbolos predicativos (variáveis, constantes, funções e predicados) e dois quantificadores $\forall$ (quantificador universal) e $\exists$ (quantificador existencial). Enquanto que, a

lógica proposicional lida com simples proposições declarativas, a lógica de primeira ordem cobre adicionalmente predicados e a quantificação. A lógica de primeira ordem tornou-se a lógica formal *standard* para o sistema axiomático, uma vez que permite raciocinar acerca de propriedades partilhadas pelos objectos através do uso de variáveis.

Um componente base da linguagem de primeira ordem é designada por ***termo***, os quais podem ser variáveis ou funções. Uma ***variável*** corresponde ao nome que denota qualquer objecto do domínio. Um símbolo de função denota a ***função*** de *aridade n* tomando como n o número de argumentos a partir de um domínio, sendo devolvido um objecto do domínio. Um símbolo de ***predicado*** corresponde ao nome da relação. Por exemplo, se f é uma função de *aridade n* e $t_1, \ldots, t_n$ são termos do mesmo domínio, então $f(t_1,\ldots,t_n)$ é um termo indicando uma função, sendo $t_1, \ldots, t_n$ designados por argumentos do literal. Neste contexto, um programa em lógica de primeira ordem pode ser definido da seguinte forma: seja P um símbolo de predicado de *aridade n* e $t_1, \ldots, t_n$ termos, então, $P(t_1 \ldots, t_n)$ e $\neg P(t_1 \ldots, t_n)$ são literais. Um programa lógico de primeira ordem corresponde a um conjunto finito de cláusulas de *Horn* [CH1985] da forma:

$$A \leftarrow L_1 \wedge \ldots \wedge L_n$$

onde $L_i \in 1 \leq i \leq n$ são literais. Se a cláusula consistir apenas na "cabeça", a cláusula é designada por ***facto***.

O método de inferência usado neste tipo de lógica baseia-se no mecanismo de prova de teorias [Kow1995]. Por outras palavras, baseia-se na demonstração de teoremas por redução ao absurdo [Har1993] [Kow1995], ou seja, tentando-se provar a negação de um teorema e chegando-se a uma inconsistência, assume-se como sendo verdadeiro, i.e., assumindo-se um axioma da teoria; no caso contrário assume-se como sendo falso. Tipicamente uma cláusula tem duas interpretações, uma interpretação declarativa e uma procedimental que define como resolver a cláusula. Por exemplo, a interpretação declarativa da cláusula *p(X,Y)* $\leftarrow$ *r(X,Z), q(Y,C)* é $\forall X, Y, Z$ *(r(X,Z), q(Y,c)*$\rightarrow$ *p(X,Y))* e a sua interpretação procedimental como modo de resolver *p(X,Y)* é resolver *r(X,Z)* e *q(Y,c)*. Neste sentido, uma questão colocada a um programa lógico é uma cláusula da forma $\leftarrow L_1, \ldots, L_n$ onde $L_i \in 1 \leq i \leq n$ são literais. Por exemplo, uma

questão $\leftarrow$ *p(X,Y)* pode ser interpretada como $\exists$ *X,Y: p(X,Y)?*.

Quando se submete uma questão a um programa em lógica é invocado um procedimento de resolução (ou *regra de inferência*) denominado por *SLD* (*Selection rule driven Linear Resolution for Definite clauses*) [GR1986] [NM2000]. O princípio da regra de inferência *SL*(*Selection rule driven Linear Resolution*), foi apresentada por *John Robinson* (1965) [Rob1965] o qual desenvolveu um método denominado por ***resolução***, sendo proposto como um procedimento de prova para provar teoremas. A *resolução* faz uso da prova por contradição, por exemplo, no sentido de provar P, $\neg P$ (a negação de P) é colocada na base de conhecimento (ou base de dados) e a contradição é derivada. Neste sentido, a resolução *SLD* utiliza a interpretação procedimental das cláusulas formando o programa lógico, de modo a validar se a questão tem derivação bem sucedida.

Por outro lado, a ***lógica de segunda ordem*** corresponde a uma extensão da lógica de primeira ordem. Enquanto que, na lógica de primeira ordem se utiliza apenas variáveis no intervalo dos indivíduos (elementos do *universo do discurso*, que correspondem a um conjunto de elementos de indivíduos que podem ser quantificáveis). Uma outra caracteristica é que na lógica de segunda ordem estas variáveis, assim como variáveis adicionais cobrem um intervalo de conjunto de indivíduos.

2.2.2 Lógicas Não Clássicas

Como referimos anteriormente, as ***lógicas não clássicas*** correspondem a uma extensão das lógicas clássicas, tentando colmatar algumas limitações das lógicas clássicas e por outro lado, permitir abordar o estudo do acreditar, do raciocínio por defeito e incompleto, assim como ultrapassar as limitações no tratamento do factor tempo. São conhecidas vários tipos de lógicas não clássicas consoante os seus fundamentos, nomeadamente, quando baseados nas lógicas clássicas (e.g. *lógica multi-valor*, *lógica fuzzy* e *lógica linear*) e quando baseada na generalização da lógica proposicional (*lógica modal*, *multimodal* ou *temporal*).

A ***lógica modal*** [LL1959] corresponde a uma expressão que é usada para qualificar a verdade de uma *sentença*. Centra-se no estudo de comportamentos dedutivos de expressões do tipo "é necessário que..." e "*é possível que...*", ou seja, utiliza além dos operadores da lógica proposicional, dois outros operadores proposicionais "□" e "◇", interpretados respectivamente com "*necessariamente*" e "*possivelmente*".

A ***lógica linear*** [Gir1987] corresponde a um refinamento da lógica clássica e intuicionista em que, em vez de enfatizar o conceito de verdade como a lógica clássica ou prova como a intuicioinista, mas antes sobre o papel das fórmulas. Este tipo de lógica teve o seu objectivo inicial na análise semântica do sistema de modelos. Baseia-se [KM2008] na teoria da prova como método formal, contendo uma evolutiva negação enquanto mantém uma forte interpretação construtiva. Por outro lado, apresenta novas formas de compreensão na natureza das provas.

A ***lógica temporal*** é um ramo da lógica modal e foi apresentada por *Arthur Prior* (1957). O objectivo da lógica temporal (denominada por vários autores por "*tense logic*") é sistematizar o raciocínio com proposições que têm um aspecto temporalizado, ou por outras palavras, para representar informação temporal dentro da arquitectura da lógica.

Para declarar ou provar propriedades de programas concorrentes, é frequentemente necessário tratar não apenas com comportamento de entrada/saída do programa, mas também com a sua sequência de execução. Isto tem levado ao desenvolvimento de linguagens de especificação para programas concorrentes que são orientadas para a descrição de sequências. Entre estas linguagens, podemos distinguir as que são baseadas em expressões regulares e aquelas baseadas numa lógica de sequência como a lógica temporal. Por sua vez, a lógica temporal tem mostrado ser adequada para expressar uma grande variedade de propriedades de programas concorrentes, tais como a terminação, a exclusão mútua, a acessibilidade, entre outras.

2.2.3 Outros Tipos de Lógicas

A ***lógica por defeito*** foi apresentada por *Reiter* (1980) [Rei1980] o qual se baseou no formalismo do conjunto de respostas (*Answer Set*) [LW1992]. Este tipo de lógica apresenta uma abordagem formal ao raciocínio por defeito, introduzindo regras de inferência não monótonas, no sentido de caracterizar suposições assumidas por defeito. O trabalho de *Bidoit* e *Froidevaux* (1987) [BF1987] baseou-se na redução de programas lógicos para teorias por defeito. De um modo similar, o trabalho de *Gelfond* [GL1991] relaciona programas lógicos a outros formalismos não monótonos, como o sistema da lógica auto epistémica [Gel1987] introduzida por *Moore* (1985) [Moo1985].

A primeira lógica modal não monótona [MD1980] é uma lógica clássica de primeira ordem com a introdução do operador modal. Neste tipo de lógica, todos os axiomas

da linguagem são expressos através de proposições sem ser necessário recorrer a regras de defeito. Por outro lado, *McDermott* [McD1982] desenvolveu esforços para resolver os problemas da primeira lógica modal não monótona [Bro1980], surgindo a lógica não monótona II através da introdução do operador modal "necessariamente". Numa outra corrente de investigação, *Dom Gabbay* [Gab1982] escolheu o modelo intuícionista para representar o raciocínio não monótono [EST2005] baseando a sua teoria no facto do conhecimento ganhar consistência à medida que o tempo vai passando.

2.3 Fundamentos

Como referimos, a *Lógica* corresponde à ciência dos princípios do raciocínio ou inferência correcta [Sim2000]. Segundo *Aristóteles "reasoning is any argument in which certain assumptions or premises are laid down and then something other than these necessarily follows."* [McK1941]. Neste contexto, vários formalismos e notações podem ser encontrados na literatura no que se refere aos fundamentos da lógica (matemática), como por exemplo [Sim2000] [NM2000]. Como fundamentos da lógica podemos destacar o conceito de fórmulas, operadores lógicos, validade lógica, consequência lógica, teorema da plenitude (*completeness*), teorias formais, entre outras. No entanto, neste trabalho de doutoramento não estamos interessados em aprofundar todos estes conceitos, mas sim apresentar de uma forma simples e sucinta alguns destes formalismos, os quais serão utilizados e referenciados nos capítulos seguintes, correspondendo a parte dos fundamentos teóricos do trabalho principal deste estudo. Assim, iremos com algum detalhe apresentar de seguida as notações de fórmulas lógicas, semântica das fórmulas lógicas e inferência lógica.

2.3.1 Fórmulas Lógicas

Nesta secção coleccionamos um conjunto de conceitos básicos que no nosso entender são necessários para introduzir a *Programação em Lógica*, um dos paradigmas utilizados neste trabalho de doutoramento. Ao longo dos anos foi produzida bastante literatura associada aos fundamentos da lógica e da programação em lógica, como por exemplo [GHR1995] [Lif1996]. *Nilsson e Malusynski* [NM2000] apresentam estes conceitos de uma forma simples e intuitiva, indo de encontro aos objectivos

que pretendemos descrever nesta secção. Por este facto, iremos seguir os exemplos apresentados nesse estudo. Quando descrevemos algum estado de qualquer coisa no mundo real muitas vezes utilizamos *sentenças declarativas*[7], como por exemplo:

(i) "*Toda a mãe ama o seu filho*"
(ii) "*Mary é mãe e Tom é filho de Mary*"

Aplicando algumas regras gerais de raciocínio, estas descrições podem ser usadas para obter novas conclusões. Por exemplo, conhecendo (i) e (ii), é possível de concluir que:

(iii) "*Mary ama Tom*"

Uma análise mais atenta revela que (i) e (ii) descrevem algum universo de pessoas e algumas relações entre estes indivíduos, como "*... é mãe*", "*... é filho de...*" ou a relação "*....ama...*", que poderá ou não interligar entre pessoas. Este exemplo reflecte a ideia principal da *Programação em Lógica*, no sentido de descrever possivelmente infinitas relações em objectos e aplicar sistemas de programação de modo a obter conclusões como em (iii).

Para um computador lidar com sentenças do tipo (i) (ii) e (iii), a sintaxe das *sentenças* terão de ser precisamente definidas. O que é ainda mais importante, são as regras de raciocínio, como aquela que permite inferir (iii) a partir de (i) e de (ii), devendo ser cuidadosamente formalizadas.

Este tipo de problemas tem vindo a ser estudados no campo da lógica matemática. As ***fórmulas lógicas*** fornecem uma sintaxe formalizada para escrever sentenças do tipo (i) e (iii), ou seja, corresponde a uma expressão construída a partir de fórmulas atómicas pela repetição da aplicação de operadores lógicos. Essas *sentenças* referem-se a indivíduos em algum mundo, assim como na relações entre esses indivíduos. Neste sentido, o ponto de partida é a assumpção acerca do alfabeto da linguagem, a qual deverá incluir:

- Símbolos para denotar indivíduos (e.g. o símbolo "*tom*" deverá ser usado para denotar a pessoa "*Tom*" no exemplo acima). Estes símbolos serão designados por constantes.

[7] Uma sentença declarativa é uma expressão completa da linguagem natural que é verdadeira ou falsa, em oposição às sentenças imperativas ou interrogativas (comandos ou questões). Apenas sentenças declarativas podem ser expressas em predicados lógicos.

- Símbolos para denotar relações (*ama*, *mãe*, *filho_de*). Estes símbolos são chamados de símbolos dos predicados.

Cada símbolo de predicado está associado a um número natural, designado por *aridade* ("*arity*"). A relação designada a cada símbolo de predicado *n-ary* é um conjunto de *n-tuplos* de indivíduos. No exemplo acima o símbolo de predicado "*ama*" denota um conjunto de pares de pessoas, incluindo o par "*Mary* " e "*Tom*", denotado pelas constantes "*mary*" e "*tom*".

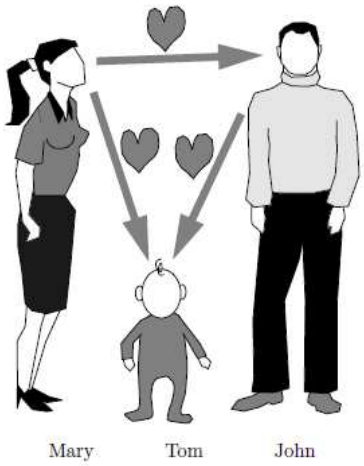


Figura 2.1 - Estrutura exemplificativa da relação familiar [NM2000].

Com o alfabeto de constantes, os símbolos de predicados, alguns caracteres auxiliares e sentenças da linguagem natural como "*Mary ama Tom*", poderá ser formalizado como fórmulas do tipo *ama(mary, tom).*

A linguagem formal deverá também disponibilizar a possibilidade de expressar sentenças como (i) que refere a todos os elementos da descrição do mundo ("*universo do discurso*"). Estas sentenças apresentam que "*para todos os indivíduos X e Y, se X é mãe e se Y é filho de X, então X ama Y*". Para este propósito, a linguagem da lógica introduz o símbolo do quantificador universal "∀" (devendo ser lido como "*para todo*" ou "*para qualquer um*") e o alfabeto das variáveis. A ***variável*** corresponde a um símbolo que se refere a um indeterminado indivíduo, como X e Y. Assim, as sentenças (i)-(iii) poderão ser formalizadas da seguinte forma:

$\forall X\ (\forall Y\ ((\text{mãe(X)} \wedge \textit{filho\_de(X,Y)}) \supset\ \textit{ama(X,Y)}))$ (1)
$\textit{mãe(mary)} \wedge \textit{filho\_de(tom, mary)}$ (2)
$\textit{ama(mary, tom)}$ (3)

Os símbolos "$\wedge$" e "$\supset$ " são exemplos de conectores lógicos que são utilizados para combinar fórmulas lógicas. O símbolo "$\wedge$" lê-se "e" e é designado por conjunção, enquanto que o símbolo "$\supset$ " é designado por implicação e corresponde à construção "*se-então*". Os parênteses são utilizados para retirar ambiguidade à linguagem.

Um outro conector que é usado com frequência é o da expressão da negação. É denotado pelo símbolo "$\neg$" (lendo-se "não"). Por exemplo, a sentença "*Tom não ama Mary*" poderá ser formalizada pela seguinte fórmula:

$\neg$ *loves(tom, mary)*

O símbolo "$\exists$" é designado por quantificador existencial e lê-se "*existe um*". O quantificador existencial torna possível de expressar factos em que, no mundo em consideração, existe pelo menos um indivíduo que tem uma certa relação com alguns outros indivíduos. Por exemplo, a sentença "*Mary tem um filho*" poderá ser formalizado pela seguinte fórmula:

$\exists X$ *filho_de(X, mary).*

Também os conectores lógicos "$\vee$" e "$\leftrightarrow$ " são utilizados, formalizando os conectores "*ou*" e "*se e somente se*". Até agora, os indivíduos foram representados apenas por constantes. Contudo, também pode ocorrer que no mundo em consideração, alguns "indivíduos" são compostos por objectos. Em certas circunstâncias poderá ser necessário discutir as relações entre famílias assim como relações entre pessoas. Neste caso será desejável de referir que para uma família pela construção de uma composição de constantes identificar os membros de uma família (o que se pretende é a construção de uma função que constrói a família a partir dos seus membros). A linguagem lógica oferece meios para resolver este problema. É assumido que o seu alfabeto contém símbolos designados por *functores* que representam funções sobre objectos do domínio. Cada *functor* está atribuído um número natural designado por *aridade* que determina o número de argumentos da função. As constantes poderão ser vistas como *functores* de *aridade* zero. Admitamos que existe o *functor* família de *aridade* três, o *functor* "*filho*" de *aridade* dois e a constante "nome". A família consiste nos pais "*Bill*" e "*Mary*" e os filhos "*Tom*" e "*Alice*", podendo ser representadas pela construção seguinte:

family(bill, mary, filho(tom, filho(alice, none))).

A esta construção é designada por uma combinação de termos ("*compound term*"). Este linguagem informal baseada em exemplos de simples sentenças declarativas dá motivação para a introdução de construtores básicos da linguagem da lógica simbólica. O tipo de lógica utilizada aqui é a lógica predicativa ("*predicate logic*") [EK1976]. A partir do ponto de vista lógico, as fórmulas são sequências finitas de símbolos tais como variáveis, *functores*, e símbolos de predicados. O alfabeto da linguagem da lógica de predicados consiste nas seguintes classes de símbolos:

- *variáveis*, que serão escritas como identificadores alfanuméricos, sendo inicializados com letras maiúsculas (e.g. X, X_s, Y, Y_n,..);
- *constantes*, que são números ou identificadores alfanuméricos, sendo inicializados por letras minúsculas (e.g. x, *alf*, *none*, 12,..);
- *functores*, que são identificadores alfanuméricos iniciados por letra minúscula estando-lhes associados uma *aridade* maior do que zero. Para realçar a *aridade* n de um *functor* f é normalmente escrito sobre a forma f/n;
- *símbolos de predicados*, que são normalmente identificadores alfanuméricos iniciados com letras minúsculas e associados com uma *aridade* maior ou igual a zero. Também a notação p/n é usada para símbolos de predicados.
- *operadores lógicos*, que são "$\wedge$" (conjunção),"$\neg$" (negação),"$\leftrightarrow$ " (equivalência lógica), "$\supset$ " (implicação), e "$\vee$" (disjunção);
- *quantificadores*, $\forall$ (universal) e $\exists$ (existencial);
- *símbolos auxiliares*, como parênteses e vírgulas.

Como notação convencional iremos utilizar *a*,*b*,*c* para denotar constantes e "X, Y, Z,..."para denotar variáveis. *Functores* são denotados por f, g, h, ... e p, q, r são utilizados para denotar ***símbolos de predicados***. ***Constantes*** são por vezes vistas como *functores* nulos (e.g. com *aridade* zero). Por outro lado, conjuntos de *functores* e símbolos de predicados poderão conter identificadores idênticos com diferentes *aridades*.

As *sentenças* da linguagem natural consistem em palavras onde os objectos da descrição do mundo são representados como nomes. Na formalização da linguagem

da lógica de predicados será representada por *strings* denominadas por termos cuja sintaxe é definida da seguinte forma:

Definição 1 (Termo): O Conjunto T de termos sobre um dado alfabeto A é o mais pequeno conjunto tal que:

- Qualquer constante A está em T;
- Qualquer variável em A está em T;
- Se f/n é um *functor* em A e $t_1, \ldots, t_n \in T$ então $f(t_1, \ldots, t_n) \in T$;

Na linguagem natural apenas algumas combinações de palavras tem significado de *sentenças*. A equivalência das *sentenças* na lógica de predicados é especialmente construída a partir de termos, sendo designados por ***fórmulas*** ou ***fórmulas bem formadas*** (*well-formed fórmulas – wff*) e a sua sintaxe é definida da seguinte forma:

Definição 2 (Fórmulas): Seja T um conjunto de termos sobre o alfabeto A. O conjunto F de *wff* (com respeito a A) é o menor conjunto tal que:

- Se p/n é um símbolo de predicado em A e $t_1, \ldots, t_n \in T$ então $p(t_1, \ldots, t_n) \in F$;
- Se F e $G \in F$ então também $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \supset G)$ e $(F \leftrightarrow G) \in F$;
- Se F $\in F$ e X é uma variável em A então $(\forall XF)$ e $(\exists XF) \in F$.

Fórmulas da forma $p(t_1, \ldots, t_n)$ são denominadas de ***fórmulas atómicas*** (ou simplesmente ***átomos***). No sentido de adoptar a mesma sintaxe da linguagem de programação *PROLOG* [Brat1990], as fórmulas na forma $(F \supset G)$ são instanciadas e escritas na forma $(G \leftarrow F)$. De modo a simplificar a notação, os parênteses serão removidos sempre que possível. Por outro lado, no sentido de eliminar a ambiguidade serão assumidos os operadores de conexão (ou operadores lógicos).

2.3.2 Semântica das Fórmulas

Na secção anterior, apresentamos a linguagem das fórmulas como a formalização da classe das sentenças declarativas da linguagem natural. Essas sentenças referem-se

a algum "mundo" podendo ser verdadeiras ou falsas. O significado de uma fórmula lógica é também definida em relação a "um mundo abstracto" designado por estrutura, sendo tambm, ou verdadeiro ou falso. Nesta secção para definir o significado de fórmula, será estabelecida uma ligação formal entre a linguagem e a estrutura.

A abstracção matemática do "mundo", designada por estrutura, é um conjunto não vazio de indivíduos (designado por *domínio*) com um número de relações e funções definidas nesse domínio. Por exemplo, a estrutura referida pelas sentenças (i)-(iii) poderá ser uma abstracção do mundo apresentado na figura 2.1 (relação entre pai e mãe e entre pai e filho e mãe e filho). O seu domínio consiste em três indivíduos – "*Mary*", "*John*" e "*Tom*". Adicionalmente, neste conjunto serão consideradas três relações: uma relação unária, "*... é mãe*", e duas relações binárias, "*... é filho de ...*" e "*... ama ...*". A construção dos blocos da linguagem de fórmulas são constantes, *functores* e símbolos de predicados.

Neste sentido, a ligação entre a linguagem e a estrutura é interligada através do conceito de ***interpretação*** semântica de termos e semântica de fórmulas bem fundadas. Sobre a semântica associada às fórmulas lógicas poderá ser encontrada mais informação em [NM2000] e em [GHR1995].

2.3.3 Inferência Lógica

Seguindo os exemplos ilustrados anteriormente, as sentenças (iii) foram obtidas pelo raciocínio acerca das sentenças (i) e (ii). A linguagem foi formalizada e as sentenças expressas como fórmulas lógicas (1), (2) e (3). Com esta formalização, o raciocínio pode ser visto como um processo de manipulação de fórmulas a partir das quais um conjunto de fórmulas, como (1) e (2) designadas por *premissas*, produzem uma nova fórmula designada por conclusão (3). Um dos objectivos da lógica simbólica é o de formalizar "princípios de raciocínio" como reescrita formal de regras que podem ser utilizadas para gerar novas fórmulas a partir de outras. Estas regras são designadas por regras de inferência (*inference rules*). Por outro lado, é requerido que as regras de inferência correspondam às correctas formas de raciocínio, sempre que as premissas são verdadeiras em qualquer mundo sobe consideração.

No entanto, alguma conclusão obtida pela aplicação de uma regra de inferência deverá também ser verdadeira nesse mundo. Por outras palavras, é requerido que as regras de inferência produzam apenas consequências lógicas das suas premissas para

as quais poderão ser aplicadas.

As regras de inferência mais conhecidas da lógica de predicados para a computação de programas lógicos são [Per2002] [MUW2002]:

• *Modus ponens* ou regra de eliminação por implicação: Esta regra refere que sempre que fórmulas da forma F e $(F \supset G)$ pertençam a, ou são concluídas a partir de um conjunto de premissas, G pode ser inferida. Esta regra é apresentada da seguinte forma:

$$\frac{F ... F \supset G}{G}(\supset E)$$

• Regra de eliminação pelo quantificador universal: Esta regra apresenta que sempre que uma fórmula da forma $(\forall XF)$ pertence a, ou é concluída a partir de uma premissa, uma nova fórmula pode ser concluída pela substituição de todas as ocorrências livres de X e F por algum termo t que é livre por X. Esta regra é apresentada da seguinte forma:

$$\frac{\forall XF(X)}{F(t)}(\forall E)$$

• Introdução de regras por disjunção: Esta regra estabelece que se as fórmulas F e G pertencem a ou são concluídas a partir das premissas, então a conclusão $F \wedge G$ podem ser inferidas. Esta regra é apresentada da seguinte forma:

$$\frac{F ... G}{F \vee G}(\wedge I)$$

A sua utilização pode ser ilustrada considerando o exemplo anterior: As premissas são as seguintes:

$$\forall X\ (\forall Y\ (mãe(X) \wedge filho\_de(Y,X) \supset\ ama(X,Y))) \quad (4)$$
$$mãe(mary) \wedge filho\_de(tom,mary) \quad (5)$$

Eliminação do quantificador universal (4) produz:

$$\forall Y\ (\textit{mãe(mary)} \wedge \textit{filho\_de(Y, mary)} \supset\ \textit{ama(mary,Y))} \qquad (6)$$

Eliminação do quantificador universal (6) produz:

$$\textit{mãe(mary)} \wedge \textit{filho\_de(tom, mary)} \supset\ \textit{ama(mary, tom)} \qquad (7)$$

Finalmente aplicando *modus ponens* a (5) e a (7) produz:

$$\textit{ama(mary, tom)} \qquad (8)$$

Assim, a conclusão (8) foi produzida de uma forma formal pela aplicação das regras,de inferência. O exemplo ilustra o conceito de *derivabilidade*. Como podemos observar (8) é obtida a partir de (4) e (5), mas não directamente, mas antes através de uma sequência de passos de inferência, cada um deles adicionando uma nova fórmula ao conjunto inicial de premissas. Qualquer fórmula F que possa ser obtida desta forma a partir de um conjunto P de premissas é dito de derivável a partir de P. Uma importante questão relacionada ao uso de regras de inferência é o problema de sempre que todas as consequências lógicas de um conjunto arbitrário de premissas P podem também ser derivadas a partir de P.

Vários investigadores apresentam o relacionamento entre a lógica e as abordagens procedimentais em que que a abordagem procedimental tem uma semântica diferente da matemática do que a semântica da lógica matemática defendida pela teoria dos modelos. Na secção seguinte iremo-nos debruçar com mais algum detalhe sobre estes dois tipos de validações, i.e., teoria dos modelos e teoria da prova.

2.3.4 Teoria dos Modelos e Teoria da Prova

O desenho e análise de linguagens de programação declarativa requerem que a computação seja descrita em duas formas: (i) por um interpretador que extrai valores de um programa; (ii) por uma especificação declarativa de que os valores deverão ser atribuídos a um programa. Na *Lógica* existem *regras de inferência*, as quais permitem derivar *consequências lógicas* (*conclusões*) de outras fórmulas (premissas). Em geral, as regras de inferência são simples, podendo ser utilizadas como passos elementares

de longas derivações. Assim, por detrás da *Matemática*, temos sempre um conjunto de *fórmulas* (*axiomas*) que definem uma teoria, a qual corresponde ao conjunto de todas as fórmulas (*teoremas*) verdadeiras em todos os modelos dos *axiomas*.

Uma derivação formal de um novo *teorema* chama-se ***prova*** ou ***demonstração***. O nome *axioma* é geralmente atribuído a um conjunto mínimo de *teoremas* que especificam uma dada teoria. Graças a um número de regras de inferência, podemos derivar uma *fórmula* directamente através de uma técnica de ***redução ao absurdo*** [Har1993] [Kow1995]. Neste contexto, para derivar a fórmula A, a partir de um conjunto de axiomas, assume-se que $\neg A$ é um teorema: se a teoria resultante for *inconsistente*, então A é um *teorema*.

A *Lógica* pode ser utilizada para representar o conhecimento em vários contextos e pode ser mecanizada através de muitos modos diferentes, como por exemplo a manipulação ("*o raciocínio acerca de*") das suas representações. O conteúdo matemático da *Lógica* organiza-se em duas teorias essenciais: teoria dos modelos e teoria da prova. Durante os anos 40 vários investigadores centraram-se na ideia de que os meta-teoremas da lógica clássica poderiam ser usados para provar teoremas matemáticos. *Tarski* [Tar1954] propôs a teoria dos modelos centrado na ideia de que modelar um fenómeno correspondia a construir uma teoria formal que a descreve e explica.

A ***teoria dos modelos*** teve como objectivo o estudo de linguagens formais (e as suas interpretações) e pelos tipos de classificações que uma linguagem formal particular tem através de estruturas de conjuntos teóricos. O estudo de teorias dos modelos é apresentada na literatura através de vários trabalhos como [CK1990] [MAR2000] os quais apresentam os formalismos associados a esta teoria nomeadamente, a linguagem e estruturas, teorias, modelos, entre outros. A teoria dos modelos assenta basicamente [Kow1995] em dois objectivos: (i) para explicar o relacionamento entre a linguagem e a experiência e (ii) para especificar a noção de consequência lógica.

Neste sentido, a teoria dos modelos está situada do lado da semântica de uma linguagem, permitindo o exame das relações entre as frases da Lógica (*axiomas*), uma vez que tenham sido interpretadas através da associação de domínios externos, graças a lhes serem atribu'idos valores de verdade. Neste sentido, se uma interpretação aplica uma fórmula numa frase verdadeira, então é designada por *modelo* desta fórmula.

Conforme apresentamos na secção 2.2.1 *John Robinson* (1965) [Rob1965] desenvolveu um método denominado por resolução *SLD* [GR1986] [NM2000] como mecan-

ismo de inferência na derivação de conclusões a partir de programas lógicos através de uma abordagem procedimental. Contudo, desde os anos 60 que a validade e eficiência da resolução proposta por *Robinson* tem sido atacada e defendida [NM2000], como são exemplos os estudos: [Win1971] [McC1973] [NA1988] [Kow1988] [Emd2006] [Kow2008].

Kowalski (1973) [Kow1973] [Kow1984] apresenta a tese de que a computação pode ser assumida pela dedução. Por outro lado, *Pat Hayes* (1973) [Hay1973] apresenta que a computação é controlada pela dedução.

Contrariamente a *Kowalski* e *Hayes*, *Carl Hewitt* (1969) [Hew1969] desenvolveu a tese de que a dedução lógica apresenta limitações no tratamento da computação concorrente em sistemas abertos. *Kowalski* (1999) in *Encyclopedia of Cognitive Science* apresenta que "*Logic Programming is the use of logic to represent programs and deduction to execute programs in logical form. To this end, many different forms of logic and many varieties of deduction have been investigated...*".

Historicamente, a ***teoria da prova*** foi apresentada por *David Hilbert* [Hei1971] no sentido de representar o raciocínio matemático através de sistemas formais dedutivos. Vários estudos têm sido apresentados na literatura sobre a *teoria da prova* [Gre1969] [TS1996] [Poh1989] [Kow1995] onde são abordados os fundamentos associados a esta teoria nomeadamente, o cálculo formal, o cálculo dedutivo, consistência semântica, entre outras.

Troelstra e *Schwichtenberg* [TS1996] apresentam um estudo bem detalhado sobre a revisão das bases da teoria da prova. Referem que o foco no campo da estrutura da teoria da prova é na representação sintáctica de representações, axiomas e regras, centrando-se na exploração e transformações ou reduções que preservam a verdadeira prova. Isto é conseguido através do tratamento do cálculo dedutivo, translações de dupla negação, aplicações do teorema "cut-elimination" (*Herbrand's Theorem*) [Bus1995] e na normalização da prova.

Contudo, a ideia de *Hilbert* de reduzir tudo da matemática a um conjunto restrito de um sistema formal foi refutada pelo teorema de *Kurt Godel* (*teorema da incompletude*), o qual demonstrou que a ideia de *Hilberts* era irrealizável. Ao longo dos anos, vários trabalhos apresentaram novos formalismos à *teoria da prova* para contornar este problema.

Jan Lukasiewicz (1926) melhorou o sistema de *Hilbert* com base na representação axiomática da lógica se se permitisse o desenho de conclusões a partir de assumpções

nas regras de inferência da lógica. Outros trabalhos se seguiram como *Stanislaw Jaskowski* (1929) que apresentou o cálculo da dedução natural e *Gerard Gentzen* (1934) que introduziu a ideia do cálculo sequencial. Em conjunto, a apresentação da dedução natural e o cálculo sequencial introduziu a ideia fundamental da prova analítica para a prova dos modelos.

Neste contexto, a teoria da prova está situada ao lado da sintaxe, permitindo a análise das relações entre frases, em função das suas derivabilidades a partir de outras frases e utilizando regras (de inferência) que operam apenas sobre o conteúdo estrutural dessas frases. A noção de consequência lógica permite realizar um raciocínio formal, no qual se foca apenas na forma sintáctica dos padrões, sem qualquer preocupação com as interpretações. Uma *fórmula* corresponde a uma consequência lógica de um conjunto de outras fórmulas se todos os modelos deste conjunto forem também um modelo da primeira.

Como resposta ao trabalho de *Johnson-Laird* [JR1991] sobre a natureza dos modelos na dedução humana, *Kowalski* em [Kow1995] apresenta um estudo comparativo entre a teoria dos modelos e a teoria da prova como mecanismos de raciocínio. Neste estudo apresenta que "... *the notion of logical consequence can be specified in purely syntactic, knowledge-assimilation terms, without the extra syntactic structures of model theory. Not only is the syntactic specification executable, but it leads directly to more efficient and more conventionally defined proof procedures.*".

Tendo em consideração o exposto em termos de Lógica Matemática, a derivação formal de um novo teorema é denominado de prova ou demonstração, sendo o nome do *axioma* geralmente atribuído ao conjunto mínimo de teoremas que especificam a teoria. Desta forma, utilizando um conjunto de regras de inferência, é possível deduzir uma fórmula directamente a partir da técnica de redução ao absurdo [Har1993] [Kow1995]. Neste sentido, a interpretação de predicados é realizada através da invocação do predicado de prova *demo* especificado da seguinte forma [Kow1995]:

$$demo(T,P)$$

em que o primeiro argumento T é uma teoria e o segundo argumento P corresponde a uma *sentença* que se irá derivar ou demonstrar a partir de T, em que T poderá ser descrito como um conjunto de cláusulas.

Contudo, ambas as abordagens têm vindo a ser aceites na relação entre a lógica e as abordagens procedimentais no tratamento de programas lógicos. No último século,

vários investigadores defendem ambas as abordagens como mecanismo de validação de programas lógicos, isto é, a teoria dos modelos [Tar1954] [GL1998] [AHP2000] [PP2005] e a teoria da prova [PCA1992] [Gre1969] [Kow1995] [Lif1996] [Nev1984] [NM2000] [Ait2009].

2.4 Lógica e Teoria do Raciocínio

2.4.1 Introdução

Desde sempre que os fundamentos da lógica matemática estão intimamente ligados à teoria do raciocínio, em especial quando se estuda o comportamento humano em termos de tarefas lógicas. O raciocínio humano pode ser caracterizado do ponto vista prático e teórico. Do ponto de vista teórico, corresponde ao raciocínio (ou não) das crenças, enquanto que, o raciocínio prático é caracterizado pela alteração do raciocínio (ou não) dos planos e intenções. Estes tipos de raciocínio estão associados à ***teoria da racionalidade***[8] [Rob2006], uma vez que os desejos ajudam a que um conjunto de problemas que se pretende raciocinar sejam resolvidos.

Por outro lado, a racionalidade não deve afectar as conclusões que são obtidas pelo raciocínio teórico, mas sim que os desejos podem racionalmente afectar as conclusões teóricas afectando as questões usadas no raciocínio teórico, de modo a responder às questões. Do ponto de vista do raciocínio prático, os desejos podem influenciar não apenas as questões que se consideram, mas também as respostas práticas que se dão a essas questões. Segundo *Harman* [Har2002] para se entender o relacionamento entre raciocínio e a lógica, torna-se necessário distinguir os conceitos e objectivos assentes nos fundamentos da lógica, como a ***implicação*** e a ***inferência***.

A *implicação* corresponde às relações entre proposições enquanto que, a *inferência* e *raciocínio* são vistos como processos filosóficos, sendo parte da *teoria da racionalidade*. Por outras palavras, a *implicação* centra-se nas relações directas entre proposições, ao passo que a *inferência* e raciocínio conduzem a possíveis alterações das crenças (*raciocínio teórico*) ou possíveis alterações em planos e intenções (*raciocínio prático*).

[8] A *teoria da racionalidade* remonta ao tempo de *Descartes* (1637) em que as crenças fossem associadas a pensamentos e justificações (conjunto de crenças), ou por outras palavras, o ser humano acredita que pelo menos crenças eram potencialmente relevantes para as conclusões podem desencadear um raciocínio lógico.

Esta diferença pode ser representada da seguinte forma [Har2002]: para a noção de implicação "*A, B e C implica D*" e para a *inferência* "*Se acredita em A, B e C então poderá aferir-se D*".

Contudo, para a implicação a primeira parte da proposição poderá ser verdadeira sem que a segunda seja verdadeira. Por exemplo, uma pessoa pode acreditar em "*A, B e C*" sem ter qualquer razão para acreditar em "*D*". O mesmo acontece para a *inferência*, i.e., uma pessoa que acredita em "*A, B e C*" realiza que "*A, B e C implica D*" poderá concluir que *D* é falso. Desta forma, uma pessoa pode ter a razão de parar de acreditar que "*A, B ou C*" ou ambas tem razão de acreditar em *D*. Esta situação reflecte situações ambíguas do raciocínio humano trazendo alguma dificuldade de entendimento quando associado a fundamentos da lógica. Por este facto, na *implicação* terão de ser entendidas as diferenças entre *consistência* e *inconsistência*. O mesmo acontece para a análise e entendimento das diferenças entre racionalidade e irracionalidade no que se refere à utilização da *inferência* ou *raciocínio*.

No que se refere à *implicação*, a *consistência* e *inconsistência* reflectem a primeira instância das relações entre proposições, sendo que as proposições são *consistentes* quando e apenas quando é possível para todas elas serem em conjunto verdadeiras.

A *teoria da racionalidade* apresentada por *Descartes* (1637) centra-se no estudo do limite de recursos da atenção, memória e tempo, em que a ideia quando aplicada à utilização da lógica, está centrada na racionalidade ideal em que as crenças são sempre consistentes efectuadas numa implicação lógica. Neste contexto da *implicação* e *inferência* como associadas à teoria do raciocínio, surgem os conceitos de *dedução*, *implicação* e *abdução*.

A ***dedução*** está centrada em determinadas relações entre proposições, em especial aquelas relações de implicação e consistência. A ***indução*** está associada ao conceito de dúvida, sendo contempladas nas premissas graus de suporte (e.g. probabilidades). Por outro lado, a ***abdução*** tem como particularidade contemplar situações em que se depara com determinadas circunstâncias passíveis de serem explicadas pela suposição de que se trata de um caso particular de uma regra geral, adoptando-se por isso, a suposição. Todos estes três tipos de metodologias de raciocínio irão ser na secção seguinte abordadas com mais algum detalhe.

2.4.2 Dedução, Indução e Abdução

A *dedução lógica* [GHR1995] é apresentada na literatura como associada à noção de prova, cujo objectivo é apresentar um mecanismo baseado em premissas e regras (*regras de inferência*) até chegar a uma conclusão. Desta forma, existem regras que o mecanismo prova (ou dedução) terá de satisfazer, no sentido de provar a questão pretendida. Segundo *Harman* [Har2001]: *"..a lógica, concebida como teoria da dedução, não é por si própria a teoria do raciocínio... No entanto é verdade que as deduções e o percurso de prova são relevantes para o raciocínio*". Contudo, a utilidade da construção de deduções e a sua importância para o raciocínio centra-se em que por vezes é aceite uma conclusão, uma vez que foi construída uma prova de uma conclusão a partir de outros factos que consideramos como aceites.

Neste sentido, a dedução permite *derivar 'b' a partir de 'a' quando 'b' é uma consequência formal de 'a'*. Por exemplo, num simples cenário de conhecimento, sabendo-se que dentro de um saco contém feijões brancos, podemos questionar se todos os feijões que estão dentro do saco são brancos. A utilização de regras de inferência, como o *modus ponens*, *modus pollens* e *modus mistakens* [Per2002], permitem que a *dedução lógica* seja indirectamente associada à teoria do raciocínio, pois permite obter conclusões a partir de outras aplicando formalismos lógicos.

A *Indução Lógica* [Vic2006], por definição, corresponde a um tipo de raciocínio que contempla o conceito de possibilidade de que uma conclusão é falsa mesmo quando todas as premissas são verdadeiras. Este tipo de lógica está associado ao conceito de dúvida, sendo contemplados graus de suporte nas premissas (probabilidade indutiva).

De um modo formal, a *indução lógica* ou *indução indutiva* [MR1994] permite derivar conclusões gerais a partir de observações, ou por outras palavras, permite: *inferir 'b' a partir de 'a'*, em que 'b' não conduz necessariamente a partir de 'a'. 'a' poderá dar uma boa razão para aceitar 'b', mas não garante 'b'. Por exemplo, seguindo o exemplo apresentado para a dedução, sabendo-se que dentro de um saco contém feijões brancos, podemos questionar se todos os feijões que estão dentro do saco são brancos. Neste sentido, a *indução* corresponde a um processo de raciocínio que constrói novo conhecimento geral (denominado de hipóteses) a partir do conhecimento inicial e observações (ou exemplos).

Por outro lado, a *abdução* (ou *hipótese*) [JJ1994] [Mag2001] foi apresentada por *Charles Peirce* [Pei1992] como um tipo de inferência lógica. Este autor estudou a dis-

tinção entre *abdução* e *indução* no que refere ao desenvolvimento de teorias científicas. A abdução ocorre quando nos deparamos com determinadas circunstâncias capazes de serem explicadas pela suposição de que se trata de um caso particular de uma regra geral, adoptando-se em função disso, a suposição. *Peirce* afirma que, com base na *abdução*, devemos tentar constatar qual será o resultado das previsões efectuadas. O raciocínio abdutivo deve ser formulado como questão, antes que se façam observações que possam ser concluídas como verdadeiras. *Peirce* [Pei1992] define abdução como: "*a method of forming a general prediction without any positive assurance that it will succeed either in the special case or usually, its justification being that it is the only possible hope of regulating our future conduct rationally*".

Desta forma, a *abdução* permite inferir "a" como uma explicação de "b", uma vez que admitindo que a abdução permite que a precondição que "a" seja inferido a partir da consequência de "b". Por exemplo, admitindo-se a hipótese de que o resultado possa ser outro, podemos questionar se só existem feijões brancos no saco. Em suma, a definição apresenta-se como um modo de raciocínio de um estado futuro desconhecido. O processo de validade das hipóteses através do raciocínio abdutivo em que uma explicação é válida se é a melhor hipótese possível do conjunto de dados conhecidos. Neste contexto, o raciocínio baseado na *abdução probabilística* [PR2009] é uma forma de validação abdutiva, sendo usada extensivamente em áreas onde as conclusões acerca de possíveis hipóteses necessitam de ser derivadas.

Em termos comparativos, a *dedução* e a *indução* demonstraram ser inadequadas para alguns fins científicos, uma vez que a *dedução* apresenta uma prova absoluta, nunca contactando com o mundo real, não permitindo contemplar observações e experimentações, e desta forma, não se podendo testar a validade das premissas. Contudo, o raciocínio indutivo exclui a *indução matemática*, a qual é considerada como uma forma de raciocínio dedutivo. Contudo, a indução é conduzida por observações, não permitindo representar a prova de uma teoria.

Por outro lado, em termos de raciocínio, a *dedução* torna-se útil na aplicação de regras gerais a casos particulares, não adicionando informação ao conhecimento já existente. Em relação à *indução*, esta corresponde à *inferência* de uma regra a partir de um caso e do resultado, ocorrendo quando se generaliza a partir de um certo número de casos em que, algo é verdadeiro e se infere que a mesma coisa será verdadeira no total do conhecimento (ou parte). No entanto, apesar de algumas limitações da *indução lógica*, vários trabalhos foram desenvolvidos aplicando este

tipo de raciocínio à lógica de dois valores aliada à teoria da probabilidade [LS1997].

Flasch e Kakas [FK2000] apresentam um estudo comparativo entre a abdução e a indução, no qual referem que o objecto da abdução é estender uma teoria com uma explicação, permitindo raciocinar com essa nova teoria, permitindo a generalização potencial de uma dada teoria. Por outro lado, o objectivo da *indução* é o de estender uma dada teoria numa nova, que poderá fornecer novas possíveis consequências observáveis. Neste sentido, estas duas técnicas de raciocínio poderão ser complementares, trazendo mais-valias à integração no contexto da formalização de teorias [Ray2005].

Ambos os três tipos de raciocínio conduziram a uma formalização prática em termos de programação, como a *programação dedutiva* [Pie2008] [PS2009], *programação indutiva* [Moy2000] e *Abuctive Logic Programming* [FK1997] [KK1998].

A *Abductive Logic Programming* combina a abdução com a programação em lógica enriquecida por restrições de integridade, no sentido de restringir o intervalo de possíveis hipóteses ou soluções.

Nas últimas décadas vários investigadores basearam os seus estudos nesta programação assentando a formalização lógica pela teoria da prova e teoria dos modelos como pilares da lógica matemática. Na secção 4.2.2 iremos apresentar duas dessas linhas investigação, em que uma delas foi escolhida como fundamento neste trabalho de doutoramento.

2.4.3 Raciocínio Não Monótono

A *lógica clássica* apresenta que adicionando conhecimento ao já existente, não invalida qualquer conclusão. Neste sentido, a lógica clássica é designada por não monótona. Em termos formais [BNT2008] sempre que uma sentença "A" é uma consequência lógica de um conjunto de sentenças "T", então "A" é também uma consequência lógica de um arbitrário conjunto de "T".

Contudo, no mundo real e no senso comum do raciocínio, o ser humano delineia conclusões verdadeiras baseadas em assumpções previamente consideradas verdadeiras, mas que se podem evidenciar como falsas. A ideia por detrás da informação por defeito é a capacidade de adoptar assumpções ou saltar para uma conclusão plausível derivada a partir da base de conhecimento na ausência de informação do contraditório.

Neste sentido, a informação derivada é anulável, uma vez que no aparecimento de nova informação a conclusão poderá necessitar de ser removida, isto é, estamos em presença do ***raciocínio não monótono*** [Gin1991] [EST2005]. Assim uma lógica apropriada é necessária, no sentido de permitir a representação de informação incompleta [Hal2005], inconsistente e informação por defeito e que suporte o raciocínio não monótono.

A implementação de esquemas de raciocínio não monótono [AP1996] [Gin1991], proporcionam uma maior flexibilidade no mecanismo de inferência e no processo de adopção de conclusões. Contudo, a tipificação de um sistema de representação do conhecimento não pode ser realizada nos mesmos termos em que se caracteriza a informação segundo a lógica clássica, nomeadamente a lógica por defeito, onde as consequências podem ser derivadas apenas por falta de evidência do contrário. No entanto, nem sempre se pretende assumir que a informação representada é a única que é válida e, muito menos, que as entidades representadas sejam as únicas existentes no mundo exterior.

Neste sentido, para lidar com a informação negativa (em bases de dados ou em bases de conhecimento) foi definido o conceito de *Pressuposto do Mundo Fechado* (*Closed-World Assumption – CWA*) [Rei1978], o qual corresponde a uma regra de inferência anulável baseada na assumpção que um conjunto de sentenças designadas para representar um domínio de aplicação, determina todas as fórmulas atómicas (definição 2) que a sustentam. Por outras palavras, podemos dizer que toda a informação que não existe mencionada na base de conhecimento é considerada falsa. A implementação de esquemas de raciocínio não monótono recorrem à negação por falha [Gel1988] utilizando o *Pressuposto do Mundo Fechado*. Contudo, um sistema capaz de completar raciocínios não monótonos não significa por si só que possa tratar informação incompleta [Hus1994].

Clark (1978) [Cla1978] apresenta a primeira semântica declarativa para programas com a negação por falha. A *Well-founded* e *unfounded literals* foram introduzidos (para programas normais) por *Van Gelder et al* [VKS1990] e por *Teodor Przymusinski* [Prz1991]. Este conceito é utilizado como os fundamentos semânticos na programação em lógica. De acordo com a semântica bem fundamentada ("*well-founded semantics*"), apenas literais bem fundamentados são contados como consequências dos programas. Como referimos, a ideia da *negação por falha* foi introduzida por *Clark* (1978) [Cla1978] e o *Pressuposto do Mundo Fechado* por *Reiter* (1978). A

negação clássica foi incorporada nas regras de sintaxe da programação em lógica por *Gelfond e Lifschitz* (1990) e uma ideia similar independente foi desenvolvida por *Pearce e Wagner* (1990). Definições equivalentes à definição de um conjunto de respostas para programas normais foram propostos independentemente por *Bidoit and Froidevaux* (1987) e *Gelfond* (1987).

Gelfond e Lifschitz (1988) apresentam a definição de conjunto de respostas para programas normais (onde os conjuntos de respostas são designados por "*stable models*"), assim como para programas com os dois tipos de negação [GL1990]. A noção de conjunto de respostas para programas disjuntos (*Answer Set Programming*) [LS1993] sem negação por falha na cabeça das regras foram definidas por *Gelfond* e *Lifschitz* [GL1991], tendo o trabalho de *Lifschitz* e *Woo* [LW1992] eliminado esta limitação, ou seja, contemplado a negação por falha. Num outro estudo, *Inoue and Sakama* [IS1994] relacionaram a negação por falha na "cabeça"da regra com a importante área da programação em lógica abdutiva [KKT1992].

2.5 Programação em Lógica

2.5.1 Introdução

A origem da *Programação em Lógica* remonta aos anos 70 e centra-se na especificação de descrições sobre as relações observáveis num contexto através do uso de mecanismos computacionais para derivar conclusões a partir de descrições declarativas. O paradigma da *Programação em Lógica* [Lif1996] [NM2000] baseia-se no uso da lógica como linguagem de programação. A *Programação em Lógica* é feita usando uma aproximação descritiva em que um programa corresponde à descrição de um problema em termos de axiomas lógicos, em contraste com a aproximação das linguagens de programação tradicionais (e.g. *C*, *Pascal*), nas quais um programa descreve uma sequência de passos a serem executados (abordagem procedimental). Na programação em lógica a execução de um programa corresponde à dedução controlada de consequentes com base numa teoria lógica, normalmente expressa como um conjunto de cláusulas de *Horn*[9] [CH1985]. Esta dedução é efectuada pelo mecanismo

[9] Uma *cláusula* é uma classe especial de fórmulas, composta exclusivamente por disjunção de literais, em que um literal é uma fórmula atómica ou a sua negação (*e.g.*, $A \ (\ B \ (\ \neg C \ (\ \neg D \ (\ \neg E)))$). Expressões lógicas deste tipo podem, sem perda de equivalência, ser reescritas em formato

de inferência intrínseco da linguagem de *Programação em Lógica* (e.g., *PROLOG* [CM1981] [Brat1990] [SS1994].

A generalidade dos programas escritos em lógica representa implicitamente a informação negativa, assumindo a aplicação do raciocínio segundo o *Pressuposto do Mundo Fechado* [Hus1994]. Uma extensão de um *Programa em Lógica* pode incluir informação negativa explicitamente, bem como explicitar directamente o *Pressuposto do Mundo Fechado* para alguns predicados. Consequentemente, torna-se possível distinguir três tipos de conclusões para uma questão: esta pode ser *verdadeira*, *falsa* ou, quando não existe informação que permita inferir uma ou outra das conclusões anteriores, a resposta à questão será *desconhecida*.

Desta forma, um *Programa em Lógica* determina respostas em termos da veracidade ou falsidade das questões, não sendo possível assim, implementar um mecanismo de raciocínio que possibilite abordar a representação de informação incompleta.

Um ***Programa em Lógica Estendida*** corresponde a um conjunto finito de cláusulas (de *Horn*) na forma:

$$A_0 \leftarrow A_1 \wedge ... \wedge A_m \wedge not\ A_{m+1} \wedge ... \wedge not\ A_n \qquad (1)$$

em que A é um átomo e os termos A_i *e not* A_i são literais. A *negação fraca* – operador *not* na programação em lógica convencional é a *negação por falha* [Gel1988]: *not* A é verdadeira senão é possível provar A, e *not* A é falso quando é possível provar A [LN1994]. Este tipo de raciocínio será suficiente num sistema com o *Pressuposto do Mundo Fechado*, mas insuficiente quando existe informação incompleta.

Assim, surge a necessidade de criar uma extensão à *Programação em Lógica* através da qual é possível representar explicitamente informação negativa. Isto é conseguido usando a distinção entre dois tipos de negação: *negação por falha na prova* e *negação forte*. A *negação forte* é representada pelos caracteres "¬" significando que a informação é realmente falsa enquanto que, a negação por falha na prova representada pelos caracteres "*not*", refere que não é possível afirmar com certeza que um dado facto é falso a partir da informação disponibilizada na base de conhec-

"se conjunção de literais então disjunção de literais" (*e.g.*, $C \leftarrow D \leftarrow E \rightarrow A \leftarrow B$))). As *Cláusulas de Horn* são um tipo especial de cláusulas onde apenas existe um literal positivo (*i.e.*, $Q \leftarrow R \leftarrow S \leftarrow T \rightarrow V$)))).

imento, enquanto que a negação forte refere com certeza que um dado predicado é falso. A distinção entre estes dois tipos de negação torna possível não apenas de ter informação verdadeira e falsa, mas também desconhecida.

2.5.2 Limitação da Programação em Lógica

As linguagens tradicionais da *Programação em Lógica* proporcionam uma ferramenta poderosa para a representação do conhecimento, uma vez que a característica não monótona da negação por falha torna possível expressar várias situações de senso comum que não são disponibilizadas pela lógica clássica. Contudo, o estudo das vantagens e limitações da *Programação em Lógica* tem vindo a ser estudadas [Hew2008]. Alguns trabalhos apresentam algumas dessas limitações [Kow1996] [Kow2002] [Emd2006], em que por exemplo indicam que a *Programação em Lógica* não permite representar directamente informação incompleta, contemplando questões de representação e de inferência diferentes das que se podem identificar num sistema não monótono.

Apesar de numa teoria da lógica clássica, o conhecimento ser equacionado em termos do que se prova ser verdadeiro, o que se prova ser falso e o que representa informação sobre a qual não se pode ser conclusivo, num programa em lógica as respostas às questões colocadas são apenas de dois tipos: verdadeiro ou falso. Isto deve-se ao facto de um programa lógico apresentar várias limitações em termos de representação de conhecimento, uma vez que não permite representar explicitamente informação negativa ou disjuntiva, para além do facto de que, em termos de uma semântica operacional se aplicar automaticamente o *Pressuposto do Mundo Fechado* a todos (ou quase todos) os predicados [Hus1994]. Assim, uma extensão à *Programação em Lógica* poderá incluir informação negativa explicitamente, bem como explicitar directamente o *Pressuposto do Mundo Fechado* para alguns predicados.

2.5.3 Extensão Programação em Lógica

O uso do paradigma lógico, materializado através da *Programação em Lógica Estendida*, tem sido aplicado com sucesso em várias áreas do conhecimento [Sha2000] [RM$^+$2002] [MA$^+$2006] [CN$^+$2008]. No entanto, o conhecimento e as suas crenças são geralmente incompletos, contraditórios ou sensíveis ao erro, sendo desejável o uso

de ferramentas formais para lidar com o problema que resulta do uso de informação incompleta, contraditória, ambígua, imperfeita, nebulosa ou omissa. Nas últimas décadas, várias técnicas não clássicas para modelar o *universo do discurso* e procedimentos de raciocínio tem sido propostos [She1991] [Kow2006] [AN+2006] [PL2007] [NM+2007].

Contudo, um programa em lógica determina respostas em termos de veracidade e falsidade das questões, não sendo possível implementar um mecanismo que possibilite abordar a representação incompleta. O objectivo de estender a *Programação em Lógica* é o de que passe a permitir representar informação negativa explicitamente. A extensão de um programa em lógica passa a conter dois tipos de negação: a negação por falha, característica dos programas em lógica tradicionais, representado pelo termo "*not*",e a negação por forte (ou clássica), como forma de identificar informação negativa, ou falsa, representada por " $\neg$ ". Em sistemas construídos com base na extensão à programação em lógica, passam a ter uma terceira hipótese de responder às questões para além de serem verdadeiras as conclusões obtidas a partir da informação positiva e falsas as obtidas em função da informação negativa, torna-se possível concluir que a resposta é desconhecida se nenhuma das conclusões anteriores for possível.

Alferes et al. [ADP1993] baseado nos modelos estáveis (*stable models*) [GL1998] e na semântica bem fundamentada (*Well Founded Semantics – WFS*) [VKS1990] apresenta um estudo sobre as semânticas dos cenários de representação do conhecimento usando a *Programação em Lógica Estendida*. *Lifshitz* e *Woo* [LW1992]apresentam um outro estudo sobre a extensão da *Programação em Lógica* como teorias epistémicas, i.e., baseando-se nos modelos estáveis e na *WFS*. *Gerhard Brewka* [Brew1996] apresenta a *WFS* para a *Programação em Lógica Estendida*, com preferências dinâmicas.

Lamma et al. [LP+1998] baseado também na *WFS* com negação explicita, apresenta um estudo em que a *Programação em Lógica Estendida* corresponde a um mecanismo de representação apropriado para a adopção da característica de três valores (*multi-valued logics*), em especial quando associada ao conceito de aprendizagem indutiva. Ao longo dos anos, outros trabalhos se seguiram na exploração da programação em lógica baseada na teoria dos modelos, como por exemplo em [PP2005] [AA2006] [Sat2006] [PL2007].

Por outro lado, *Neves et al.* (1997) [NM+1997] baseado no trabalho de [Nev1984] sobre a apresentação de um interpretador lógico para tratar e manusear a carac-

terística temporal e a negação em bases de dados dedutivas, apresenta um estudo sobre a utilização da programação em lógica estendida aplicada à especificação de *sistemas Multi-Agente* quando utilizados em ambientes computacionais [MA+2006a] [MA+2006b] [NM+2007] [MA+2008] [MM+2009]. Neste contexto, a *Programação em Lógica Estendida* provou ser uma ferramenta adequada para a representação do conhecimento e mecanismos de raciocínio, em particular quando se tenciona atender a situações onde a informação é vaga, imperfeita ou incompleta.

Neste sentido, a *Extensão à Programação em Lógica* corresponde a uma aproximação à *Representação do Conhecimento* e raciocínio, favorecendo a aplicação de uma linguagem declarativa para a *Representação do Conhecimento* no campo da *Inteligência Artificial*. É considerada uma extensão à linguagem de *definite logic programs* [GR1986] [NM2000] pela negação clássica, disjunções, *meta-raciocínio* e raciocínio em sistemas abertos, que estendem o poder da representação da linguagem, no sentido de manipular informação incompleta.

2.5.4 Sistemas de Programação em Lógica

Ao longo dos séculos várias abordagens [Sha1989] têm sido realizadas sobre linguagens da *Programação em Lógica*. A primeira linguagem baseada no paradigma procedimental foi o sistema *Planner* apresentado por *Carl Hewitt* (1969) [Hew1969], o qual corresponde a uma forma híbrida dos paradigmas procedimental e lógico cujo objectivo se centrou na utilização da interpretação procedimental das sentenças lógicas, em que a implicação do tipo "*P implica Q*" fosse procedimentalmente interpretada. No entanto, vários investigadores apontaram algumas limitações a este sistema como por exemplo, limitações na invocação de padrões directos dos planos procedimentais a partir das afirmações "*assertions*" e no tratamento da negação lógica.

O *PROLOG (PROgramming em LOGique)* surgiu em França em 1972 por *Philippe Roussel* o qual correspondeu a um sistema baseado no sistema *Planner* resolvendo as suas limitações. No entanto, o *PROLOG* [Brat1990] [SS1994] [NM2000] apresentou algumas limitações, como por exemplo, não incluir a negação ou a disjunção da *Lógica Matemática*. Para colmatar esta limitação é comum o uso de alguns conectores especiais, como o "\+" ou o" ~ ".

Ao longo das últimas décadas, a linguagem *PROLOG* foi reconhecida e aceite como forma de codificar e executar programas lógicos, tendo vindo a ser utilizada

como ferramenta de desenvolvimento aplicacional e científico na utilização do paradigma lógico.

Hassan Ait-Kaci (2009) [Ait2009] apresenta no seu estudo que: "*it would be like saying Prolog and SLD-Resolution is the only way todo logic Programming. To some extent, the Logic Programming community's insistence on clinging to this "exclusive method" has contributed to the relative disinterest in Logic Programming following its development in the 1980's and 1990's.*".

Nas últimas décadas, vários sistemas têm sido desenvolvidos tendo como base estes formalismos, como por exemplo *XSB* [Xsb1999], *HyProlog* [Chi2008], *SWI-Prolog* [Swi2010], entre outros.

2.6 Conclusão

Neste capítulo apresentamos uma breve introdução aos conceitos e fundamentos associados à *Lógica* e à *Programação em Lógica*. Apresentamos uma breve introdução histórica, os vários tipos de lógicas, as suas abordagens matemáticas, os mecanismos de raciocínio e as extensões à programação em lógica. Conforme apresentamos, a *Lógica* e a *Programação em Lógica* tem vindo a ser aplicadas com sucesso na representação do conhecimento, assim como formalismo de raciocínio associado à resolução de problemas, em especial em cenários em que a informação é imprecisa ou omissa. Isto motiva a utilização da *Programação em Lógica* para resolver problemas nestes cenários. Neste sentido, no capítulo três iremos apresentar um breve estudo da utilização da lógica com os paradigmas evolucionário e conexionista enquanto que, no capítulo quatro e cinco iremos apresentar vários exemplos da utilização da *Programação em Lógica Estendida* juntamente com os paradigmas evolutivo e conexionista, de modo a criar sistemas inteligentes que optimizem o *universo do discurso*, dado aqui através das funções lógico matemáticas materializadas por teorias ou programas lógicos.

"One thing I have learned in a long life: that all our science, measured against reality, is primitive and childlike – and yet it is the most precious thing we have."
[Einstein in His own words, Anne Rooney,2006]

Capítulo 3

Inteligência Evolucionária e Programação em Lógica

A *Inteligência Evolucionária* corresponde à utilização de mecanismos computacionais baseados no comportamento humano e na teoria da evolução das espécies (*teoria de Darwin* [Dar1859]). Pela sua simplicidade e eficiência, estes métodos computacionais têm sido aplicados e reconhecidos como formas eficientes de resolver problemas em várias áreas do conhecimento. Por outro lado, vários investigadores tem se debruçado sobre a interligação destes sistemas com outros mecanismos computacionais orientados para fins específicos, como é o caso do paradigma simbólico para a representação do conhecimento e mecanismo de raciocínio na resolução de problemas.

Este capítulo centra-se na apresentação dos conceitos associados ao paradigma conexionista e ao paradigma evolutivo, dando especial atenção às características da *Programação Genética*, a qual é utilizada neste trabalho de doutoramento, sendo mais aprofundada no capítulo quatro. Iniciamos com uma breve introdução aos *Sistemas Inteligentes* orientados na resolução de problemas em cenários de informação imprecisa, assim como a combinação de técnicas de *Inteligência Artificial* na resolução de problemas. Na secção 3.2 apresentamos os conceitos associados à *Computação Evolucionária*, dando particular atenção à *Programação Genética*. Na secção 3.3 apresentamos uma breve descrição sobre os sistemas conexionistas, em particular as suas características, topologias e áreas de aplicabilidade. Também nesta secção apresentamos o contributo deste trabalho de doutoramento na área da utilização das redes em processos de classificação. Nas secções 3.4 e 3.5 apresentamos um breve

estudo relacionado com a integração do paradigma simbólico com o evolutivo e o conexionista, respectivamente.

3.1 Introdução

Os *Sistemas Inteligentes* requerem a capacidade de raciocinar com informação incompleta [Hal2005] pelo facto de que no mundo real a informação total é difícil de ser obtida, mesmo em situações bastante controladas. Por outro lado, o ser humano toma a maior parte das suas decisões, se não todas, baseada em informação incompleta, imprecisa e mesmo incerta. Um outro factor importante na flexibilidade do ser humano raciocinar acerca de sistemas complexos advém da capacidade natural de usar informação parcial e de combiná-la de acordo com a sua disponibilidade.

Numa outra vertente computacional, muitos dos sistemas de informação ignoram esta característica da informação acerca do mundo real e constroem modelos baseados em alguma idealização que elimina a herança da incerteza [NM$^+$1997]. O resultado é um sistema que nunca disponibiliza as respostas esperadas dada a sua incapacidade de modelar o mundo. Em vez disso, deve-se lidar com a incerteza nos próprios modelos. De facto, surge a necessidade de se implementar sistemas de informação úteis, nomeadamente com conhecimento incerto, tornando-se necessário representar e raciocinar em ambientes em que a informação é nebulosa, omissa, ou sensível ao erro. A combinação de técnicas de *Inteligência Artificial* como o paradigma simbólico e sistemas evolutivos permite aliar as vantagens de cada uma destas abordagens, em particular para a resolução de problemas complexos.

Os *Algoritmos Evolucionários* [Ang2000] correspondem a um modelo da evolução natural apresentada por *Charles Darwin* [Dar1859] tendo vindo a ser aplicados com sucesso na resolução de problemas de optimização e pesquisa de soluções. Estes algoritmos baseiam-se na adaptação colectiva e na capacidade de aprendizagem dos indivíduos. Através de um processo dinâmico aplicando operadores genéticos, os indivíduos vão sendo gerados e testados através de uma função de avaliação, tentando-se encontrar o melhor indivíduo, o qual corresponde à melhor solução para resolver o problema em causa.

As *Redes Neuronais Artificiais* [Hay1998] são um conhecido paradigma de processamento de informação inspirado na forma como o cérebro humano processa a

informação ao nível biológico. O comportamento de coleccionar a informação das redes demonstrou que, o seu comportamento de alto nível tem a capacidade de aprender a partir dos dados submetidos à rede.

3.2 Computação Evolucionária

3.2.1 Introdução

Desde os anos 40 que a ideia de aplicar os princípios da teoria de *Darwin* [Dar1859] no processo de resolução de problemas complexos tem vindo a ser seguida nos anos 60 por vários investigadores com o objectivo de criar formalismos matemáticos e computacionais para a resolução de problemas. Foram apresentadas várias técnicas na área da *Inteligência Artificial* [RN1995] com a intenção de resolver problemas complexos na área da optimização. Foi o caso de *Fogel*, *Owens* e *Walsh* [FOW1966] que apresentaram a *Programação Evolucionária*, *Holland* [Hol1962] os *Algoritmos Genéticos* e *Rechemberg* e *Schwefel* [RS1965] as *Estratégias Evolucionárias*. Durante os anos estes trabalhos seguiram direcções de investigação separadas, tendo nos anos 90, surgido por *Koza* [Koz1992] [Koz1994] a *Programação Genética*. Estas técnicas e estratégias de utilização de processos estocásticos iterativos baseados na teoria *Darwinista* para a resolução de problemas foram baptizadas de "*Computação Evolucionária*".

Por sua vez, um algoritmo baseado na Computação Evolucionária é denominado na literatura como "*Algoritmos Evolucionários*". Ao longo dos anos foi produzida bastante literatura associada à *Computação Evolucionária*. Neste sentido, a informação associada a cada uma das técnicas poderá ser complementada pela literatura corrente, como por exemplo em [Jon2006] [Ang2000] [BFM2000a] [BFM2000b] [ES2003].

Como referimos os métodos (ou algoritmos) mais importantes associados à Computação Evolucionária são, os *Algoritmos Genéticos*, as *Estratégias Evolucionária*, a *Programação Evolucionária* e a *Programação Genética*. No entanto, heurísticas de pesquisa similares a estes métodos estão relacionadas com a *Computação Evolucionária*, como *Simulated Annealing* [AK1989], *Hill Climbing* [OO1994], *Particle Swarm* [Ken2006], *Ant Systems* [DS2004] e *Tabu Search* [Glo1996]. Nos últimos anos a utilização dos *Algoritmos Evolucionários* tem sido aplicado com sucesso em vários processos de optimização de problemas complexos, nomeadamente em processos de:

optimização [GR1987], planeamento [JGB1992], desenho [Ben1999], escalonamento [CRF1994], classificação [Kei2002], só para enumerar alguns. Essencialmente, estes estudos diferem no modo de representação dos indivíduos e de alguns mecanismos associados ao processo evolutivo. Neste sentido e devido à diversidade dos métodos dos *Algoritmos Evolucionários* torna-se fácil seleccionar o método mais adequado para a resolução do problema em causa, tendo em consideração os tipos de dados que são processados, a representação da solução e a topologia do espaço de procura.

Neste subcapítulo e face ao exposto iremos apresentar uma breve abordagem aos conceitos associados à *Computação Evolucionária*, como os componentes e o ciclo evolutivo, dando especial atenção à *Programação Genética* sobre a qual se fundamenta parte deste trabalho de doutoramento.

3.2.2 Componentes de um Algoritmo Evolucionário

Os componentes mais importantes de um *Algoritmo Evolucionário* são [ES2003]: a representação (ou codificação), a função de avaliação, o mecanismo de selecção, os operadores genéticos e o mecanismo de selecção. A representação corresponde à definição dos indivíduos (vulgarmente designados de *cromossomas*) e tem como objectivo representar indivíduos na forma de soluções dentro do contexto do problema a resolver. Consoante o tipo de heurística a utilizar os tipos mais comuns são a representação por *strings* (de forma binária ou não binária), em árvore, com valores numéricos, valores reais ou símbolos [ES2003].

O mecanismo de selecção tem como objectivo distinguir os indivíduos consoante a sua qualidade, de modo a seleccionar os melhores indivíduos para serem os "parentes" nas próximas gerações. Os principais métodos de selecção [ES2003] são: *fitness proportional selection*, *ranking selection*, *probabilities selection* (i.e. *roulete* e *stochastic sampling*) e *tournment selection*.

O objectivo da utilização dos operadores genéticos é o de criar novos indivíduos a partir de outros. Os dois operadores geralmente aplicados no processo de computação evolutiva são a recombinação (ou cruzamento) e mutação. O cruzamento troca *genes* codificando os atributos da solução entre os parentes no sentido de gerar novos descendentes. Tal como para o operador de cruzamento, o operador de mutação difere da codificação dos indivíduos. Na representação binária a mutação é realizada num único ponto enquanto que, numa representação não binária a mutação é feita pela

substituição de um atributo. Por outro lado, neste operador e consoante o tipo de representação (números reais, valores discretos ou em árvore) a aplicação do operador poderá ser efectuada aleatoriamente. No caso da representação em árvore, com a aplicação do operador de mutação, será criado uma sub-árvore sendo "promovida" a nova árvore em detrimento da sub-árvore que foi removida.

O mecanismo de sobrevivência (*replacement*) é baseado na distinção entre a qualidade dos indivíduos. Este mecanismo é similar ao mecanismo de selecção dos parentes, no entanto, é usado num estado diferente do ciclo evolucionário.

Como referimos, o estudo de aplicabilidade dos *Algoritmos Evolucionários* é antiga tendo sido produzida bastante literatura ao longo dos últimos anos. Neste contexto, a breve informação apresentada nesta secção poderá ser complementada com outra literatura, como por exemplo [Ang2000] [ES2003] [Jon2006].

3.2.3 Ciclo Evolutivo

A ideia básica do mecanismo dinâmico das heurísticas evolucionárias segue a orientação básica da teoria da evolução *Darwinista* [Dar1859]. Baseiam-se na adaptação colectiva e capacidade de aprendizagem dos indivíduos em que, indivíduos de uma população representam uma possível solução para o problema. No inicio, os indivíduos numa população são inicializados aleatoriamente podendo ser modificados por operadores de selecção, mutação e recombinação (ou cruzamento). Neste sentido, a evolução é baseada e medida de acordo com uma métrica denominada vulgarmente por função de avaliação. Desta forma, em gerações posteriores irão aparecendo indivíduos representando as melhores soluções. Como referimos, dentro de cada paradigma existem vários algoritmos com diferentes características. No entanto, o processo evolutivo baseia-se no procedimento [Jon2006] descrito na figura 3.1.

O algoritmo (ou esquema) ilustrado na figura 3.1 apresenta a forma básica de um *Algoritmo Evolucionário*. Trata-se de um processo estocástico iterativo que opera sobre um conjunto de indivíduos (população). Cada indivíduo representa uma solução potencial, sendo a solução obtida através do mecanismo de codificação (ou descodificação). Inicialmente a população é gerada aleatoriamente, podendo ser realizada com várias estratégias. Depois, cada indivíduo da população é avaliado sendo, adicionados ao processo iterativo (dentro do ciclo de repetição, linha 3 a 9 do procedimento ilustrado ilustrado na figura 3.1). Dentro do processo evolutivo, um número

Algoritmo 1 – Forma básica do processo evolutivo de um Algoritmo Evolucionário

```
1.  Inicializar uma população;
2.  Avaliar cada indivíduo da população;
3.  Repetir
4.  .....Seleccionar parentes;
5.  .....Recombinar pares de parentes;
6.  .....Mutar o resultado dos descendentes;
7.  .....Avaliar descendentes;
8.  .....Inserir descendentes na população;
9.  Até (critério de paragem)
10. Extrair a solução da população.
```

Figura 3.1 - Forma básica do processo evolutivo de um Algoritmo Evolucionário.

de indivíduos são seleccionados a partir da população no sentido de se obter descendentes (passo 4 da figura 3.1). Estes descendentes são gerados através da aplicação dos operadores de cruzamento e mutação (passo 5 e 6 da figura 3.1). De seguida, os descendentes são avaliados e inseridos na população. Este processo é continuado até se chegar a um critério de paragem, como por exemplo um número máximo de iterações ou até se encontrar uma solução com uma determinada qualidade.

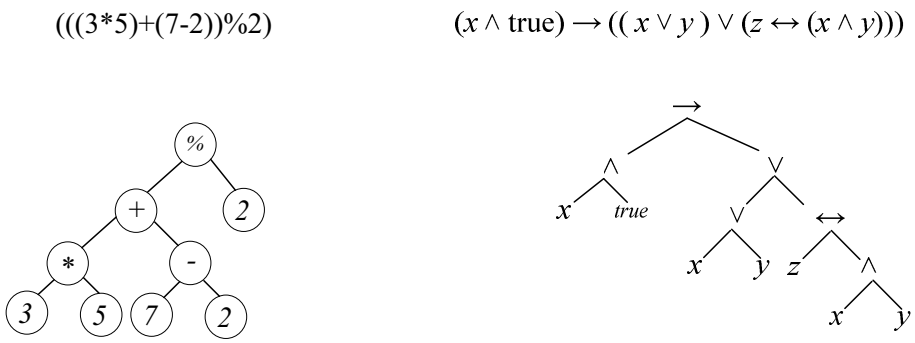
3.2.4 Programação Genética

Os *Algoritmos Genéticos* e a *Programação Genética* [Koz1992] [BN$^+$1998] [ES2003] são dois diferentes *Algoritmos Evolucionários*. Os *Algoritmos Genéticos* correspondem a uma *metaheuristica* baseada na pesquisa inspirada pelo princípio da evolução de *Darwin* pela teoria da genética [Dar1859]. Foram propostos por *John Holland* em 1975 como forma de explicar o processo de evolução através da criação de sistemas artificiais baseados nessa teoria.

A *Programação Genética* segue uma abordagem diferente dos *Algoritmos Genéticos*. Em vez de codificar *strings* que representam uma solução particular de um problema, a *Programação Genética* gera programas computacionais, representados na forma de árvore (figura 3.2). A característica geral da *Programação Genética* é a representação de programas ou conjuntos de instruções como atributos na característica de representação/codificação associada aos *Algoritmos Evolucionários*. Estas estruturas são

alteradas durante o processo de optimização em que a característica de representar a população de cromossomas como árvores dinâmicas, permite tornar o processo evolutivo mais flexível para a estrutura de dados.

Neste contexto, o campo da evolução artificial que disponibiliza o método mais conveniente para codificar funções matemáticas e fórmulas lógicas envolvendo-as no processo genético é a *Programação Genética.* Nestes métodos, as expressões matemáticas e/ou lógicas na forma de expressões sequênciais são representadas como estruturas [Ang2000], tal como o exemplo ilustrado na figura 3.2.



(a) Fórmula Aritmética (b) Fórmula Lógica

Figura 3.2 - Exemplos da representação da Programação Genética usando fórmulas aritméticas e fórmulas lógicas.

Nesta figura apresenta-se uma ilustração como exemplo da representação usando fórmulas aritméticas (figura 3.2(a)) e fórmulas lógicas (figura 3.2(b)). As árvores são lidas iniciando-se a partir do topo, resolvendo um nodo de cada vez de uma forma ascendente. Tipicamente, uma árvore da *Programação Genética* contém dois tipos de nodos: terminais e funções. Os terminais são os inputs da árvore. Por exemplo, na figura 3.2 (a), os terminais são os números 3, 5, 7, 2 e 2. Os terminais podem ser constantes ou variáveis. Os nodos de função são responsáveis por desencadear operações nos seus argumentos, neste caso da figura 3.2 (a), a multiplicação, subtracção, adição e a divisão protegida, que significa que a divisão por zero produz o valor 1 em vez de erro. Neste sentido, este tipo de representação torna-se simples e flexível para a representação de expressões (ou programas).

Tal como noutros *Algoritmos Evolucionários*, na *Programação Genética* os indivíduos são gerados inicialmente de uma forma aleatória, tendo em consideração

alguma limitação na sua geração, no sentido de não exceder um número máximo de níveis de profundidade da árvore dos descendentes (normalmente especificado pelo utilizador). No entanto, outras técnicas [Koz1994] [PLM2008] poderão ser usadas de modo a dar mais liberdade na criação de árvores com níveis de profundidade equitativos.

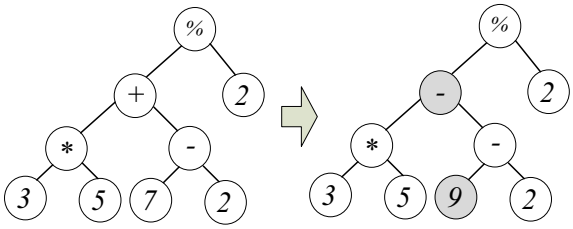


Figura 3.3 - Ilustração da operação de mutação usando a Programação Genética.

Em termos de implementação dos operadores genéticos, a *Programação em Genética* difere de outros algoritmos evolucionários. Nas árvores da *Programação Genética* existem dois tipos de operadores genéticos: mutações num ponto (*subtree mutation*) e operador de cruzamento (*subtree crossover*).

Como se ilustra na figura 3.3, a mutação num ponto altera simplesmente o valor aleatório de um indivíduo dentro da árvore, substituindo novos terminais no nodo terminal seleccionado e, gerar novas funções nos nodos seleccionados. No operador de cruzamento é criada uma alteração recíproca no material genético entre quaisquer dois genes do mesmo tipo, produzindo descendentes que são uma mistura genética dos seus parentes. Como se ilustra na figura 3.4, o operador de cruzamento trabalha seleccionando aleatoriamente um aglomerado em que dentro de cada árvore "parental" é trocada por outra.

Com a aplicação do operador de mutação, originará a modificação de exactamente uma sub-árvore, conforme se ilustra na figura 3.4. Tal como noutros *Algoritmos Evolucionários* podem ser aplicados também aqui na *Programação Genética* vários tipos de operadores [PLM2008], podendo ser por vezes aplicados a composição dos dois operadores, estando normalmente nestes casos associado a uma taxa dos operadores.

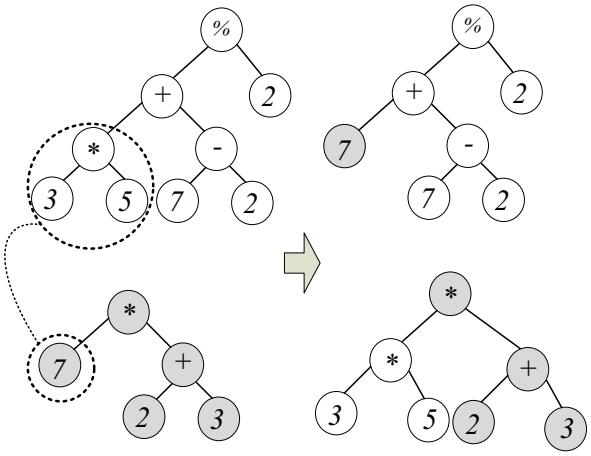


Figura 3.4 - Ilustração da operação de cruzamento usando a Programação Genética.

Nas últimas décadas, a *Programação Genética* têm sido aplicados com sucesso em várias áreas do conhecimento [ORW2005] [PLM2008], em especial como ferramenta automática de programação, ferramenta de aprendizagem automática e como motor de resolução automática de problemas. A sua aplicabilidade estende-se desde as áreas da matemática (ex. regressão simbólica), processamento de sinal e de imagem, previsão de séries temporais, processamento e controlo industrial, medicina, biologia, assim como em processos de toma à decisão.

3.3 Sistemas Conexionistas

3.3.1 Introdução

As *Redes Neuronais Artificiais* [WL1990] [Hay1998] são um paradigma computacional orientado para o processamento de informação inspirado na forma como o cérebro humano processa a informação. O estudo computacional das redes remonta à década de 1940, quando *McCulloch e Pitts* (1943) [MP1943] desenvolveram o primeiro modelo. Foi seguida nos anos 60 pelo modelo *perceptron* desenhado por *Rosenblatt* [MR1986], o qual gerou interesse da comunidade científica devido à sua capacidade de resolver problemas de classificação usando padrões de dados. Este interesse reduziu-se durante uns anos quando *Minsky e Papert* (1969) [MP1969] apresentou provas matemáticas

das limitações do *perceptron*[1], apontando a sua limitação computacional na incapacidade de resolver o problema clássico do "*ou exclusivo*" (conhecido como o problema *XOR*). Ao longo dos anos 80 vários investigadores desenvolveram modelos de rede mais sofisticados e melhores algoritmos de treino.

Com a evolução da capacidade dos sistemas computacionais, nas últimas décadas aumentou largamente o interesse do uso destas redes, tendo vindo a ser reconhecidas e aplicadas com sucesso num largo domínio de áreas científicas, como no reconhecimento de padrões, no reconhecimento de voz, no processamento de imagem, regressão, controlo e previsão, processos de inferência, na tomada à decisão, entre outros.

Existe uma grande variedade de modelos de redes neuronais, em particular sobre *Multi Layer Perceptron* [ZY2001], *Support Vector Machines* [LDM2009], Radial Basis Function [NY2002], self-organizing Maps [Koh2001], Adaptive Resonance Theory (ART) [Din2006], máquina de *Boltzmann* [AT1995], Simulated annealing [Ing1993], Time-Delay Neural Networks [CG$^+$1997], Recurrent Neural Networks [Bod2002], Spiking Neural Networks [GK2002], Reservoir Computing [LJ2009], Pulse-coupled Neural Networks [LK2005], Adaline/Madaline [WL1990]. As múltiplas aplicações das *Redes Neuronais Neuronais*, podendo a informação contida nesta secção ser complementada em [WRL1994] [NY2002] [Abra2004] [RD2005] e [Hay2008].

3.3.2 Características

As *Redes Neuronais Artificiais* são caracterizadas pela sua estrutura (topologia) e pelos seus parâmetros (os quais incluem os pesos das conexões) [Roj1996]. São estruturas extremamente interconectadas [Wei2005] por unidades computacionais (*neurónios*) que têm a capacidade de responder à informação dos *inputs*, aprendendo a adaptar-se a determinados ambientes. As *Redes Neuronais Artificiais* operam em duas fases: a primeira fase corresponde à fase de programação (ou de treino) durante a qual a força (peso) de cada conexão é determinada, seguida por uma fase de execução, durante a qual a rede está pronta para ser usada. Depois de terminada a fase de programação, o processo de aprendizagem termina e as capacidades de processamento são "*estagnadas*". Desta forma, na fase de execução a *Rede Neuronal Artificial* não irá aprender nada de novo. Assim, com a rede treinada por exemplo, para re-

[1] O *perceptron* é um neurónio de *McCullock-Pitts* (1943) [MP1943] de um nível simples de conexão. É capaz de classificar correctamente padrões de entrada linearmente separáveis.

conhecer padrões de imagem, a introdução de nova informação no conhecimento da rede irá obrigar a que a rede seja limpa e treinada desde o inicio.

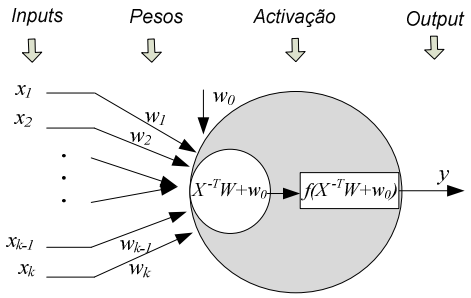


Figura 3.5 - Estrutura típica de um neurónio das Redes Neuronais Artificiais.

Na estrutura (linear) do neurónio (figura 3.5), o neurónio recebe um número de *inputs* de neurónios precedentes ponderando o peso. A soma dos *inputs* é formada e o sinal de activação é transferido através de uma função de activação [Has1992] para produzir o *output* do neurónio.

Contudo, há a salientar que devido à grande variedade de tipos de *Redes Neuronais Artificiais* é comum existirem outros tipos de neurónios variando consoante o seu tipo de activação. Por exemplo, nas *Radial Basis Function Neural Networks* [NY2002] que são um tipo especial de redes *Feedforward* existem dois tipos de neurónios, um é o *Radial Basis Function Neuron* em que a função de activação é a função *Gausiana* e o neurónio linear com funções de activação lineares, como a *step*, *sigmoid* ou *signal* [Wei2005].

Em termos de métodos de treino existem vários tipos para a geração dos pesos das conexões. Muitos deles são métodos que ajustam os parâmetros da rede a um dado problema, mas não à estrutura da rede. Dois dos tipos de treino das *Redes Neuronais Artificiais* mais conhecidos são a *aprendizagem supervisionada* e *aprendizagem não supervisionada*.

A ***aprendizagem supervisionada*** que incorpora uma entrada externa (denominada de "professor") e desta forma o treino contém exemplos dos *inputs* em conjunto com o seu correspondente *output*, em que a rede aprende a inferir os relacionamentos entre as entradas e as saídas. Neste tipo de aprendizagem a rede é treinada usando um dos algoritmos supervisionados [Roj1996] [Wei2005] [CY+2006] onde o mais

conhecido é o *backpropagation* [Roj1996, capitulo 7], no qual usa informação para ajustar os pesos da rede e *thresholds* para minimizar o erro nas previsões do conjunto de treino.

O segundo tipo de aprendizagem é a ***aprendizagem não supervisionada*** [CY+2006] que não usa um "professor" e é baseado apenas em informação local e critérios de similitude ou distância. Também se denomina por *Self-Organization*, no sentido que a rede se auto organiza, detectando determinadas propriedades ao longo do processo de aprendizagem.

3.3.3 Topologias

As arquitecturas das *Redes Neuronais Artificiais* podem ser divididas em vários tipos correspondendo à forma como os nodos se interligam. Existem inúmeros tipos de arquitecturas [Hay1998], como por exemplo a *MultiLayer Perceptron*, a *Radial Basis Function* e a *Self Organization Maps*, cada uma com as suas próprias potencialidades, podendo ser caracterizadas em duas categorias: unidireccionais (*feedforward*) e recorrentes. As primeiras são organizadas por camadas em que as conexões se propagam sempre numa única direcção, desde a camada de entrada (camada de nível inferior) até à camada de saída (camada de nível superior), passando por uma ou várias camadas intermédias. Na figura 3.6 apresenta-se uma ilustração das redes unidireccionais.

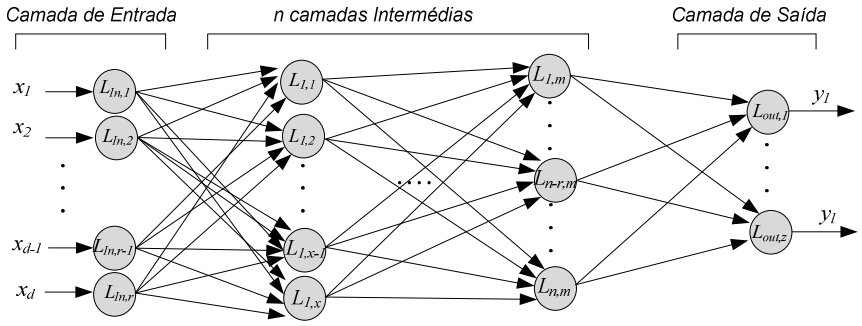


Figura 3.6 - Estrutura arquitectural das RNA do tipo Unidireccional (FeedForward).

As redes *Feedforward* tem como finalidade estabelecer o mapeamento dos *inputs*

com os *outputs*, adoptando os seus pesos aos exemplos de treino, de modo a analisar a sensibilidade dos *inputs* da rede ou à variação de pesos, no sentido de avaliar o seu desempenho, como a tolerância a erros ou a capacidade de generalização.

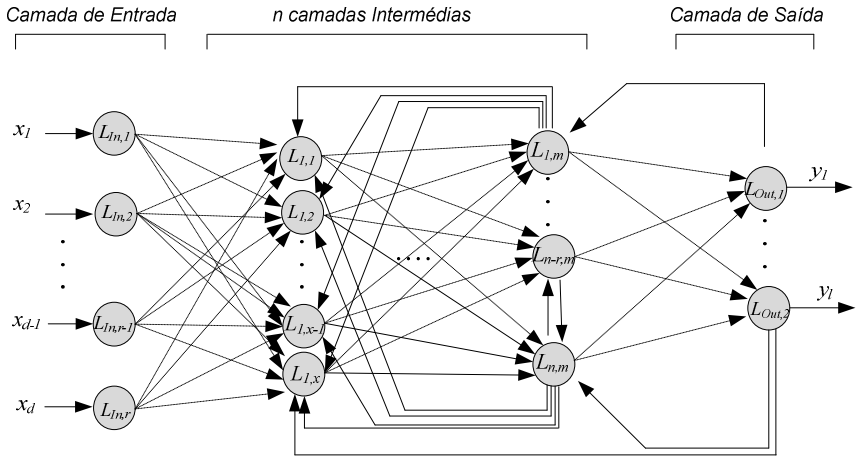


Figura 3.7 - Estrutura arquitectural das RNA do tipo Recorrente.

As redes recorrentes (figura 3.7) correspondem a redes em que existe a possibilidade dos neurónios de camadas superiores se ligarem a outras camadas de nível inferior. Ao conter ciclos, as saídas não são em função exclusivamente das conexões entre nodos, mas também de uma dimensão temporal, i.e. está-se na presença de um cálculo recursivo que obedecerá a uma certa condição de paragem.

Ao longo da história das *Redes Neuronais Artificiais*, várias topologias foram apresentadas com as suas diferentes particularidades, no entanto todas elas se baseiam nas redes *feedforward* e nas redes recorrentes. Nas últimas décadas surgiu uma nova área de investigação denominada de *Redes Neuronais Evolucionárias* [Yao1999] orientadas para o estudo da integração dos paradigmas conexionista e evolutivo. Estes estudos apresentam e exploram diferentes abordagens para a criação dinâmica de topologias de redes, como é o caso dos seguintes trabalhos: [SWE1992] [Bra1995] [B$^+$2001] [CA$^+$2002] [Abra2004] [Nit2007], [Nit2008], [FDM2008], [KC2008], [SM2010].

3.3.4 Contribuição

De acordo com *Einstein*, "*The... paper is only a rough draft right now, and is an electrodynamics of moving bodies which employs a modification of the theory of space and time.*" [Ann2006]. Durante este trabalho de doutoramento e conforme apresentado na metodologia adoptada (ver secção 1.5) surgiu a necessidade de efectuar um estudo detalhado sobre as características e fundamentos das *Redes Neuronais Artificiais*. Pretendeu-se estudar as várias topologias de rede, no sentido de avaliar a sua aplicabilidade na criação do sistema evolutivo proposto como objectivo deste trabalho. Durante esse estudo realizamos alguns trabalhos aplicando topologias de rede simples a problemas de classificação.

No artigo [FR$^+$2010a] apresentamos a "*Fast weight calculation for kernel-based perceptron in two-class classification problems*", onde propomos um método denominado *Direct Kernel Perceptron* para calcular directamente os pesos de um *perceptron* simples usando a expressão *closed-form* a qual não requer qualquer fase de treino. Os pesos minimizam o desempenho, tendo em consideração os erros de treino e a margem de classificação do *perceptron*. A capacidade de aprendizagem não linear separável dos problemas é fornecida por um mapeamento entre as entradas e o espaço intermédio (ou oculto). Usando mapeamentos *Gaussianos* (*Gaussian kernels*), o método *Direct Kernel Perceptron* alcança melhores resultados que o *standard Support Vector Machine* e o *Linear Discriminant Analysis* para uma variedade de conjuntos de dados de teste de duas classes. Por outro lado, o custo computacional do *Direct Kernel Perceptron* diminui linearmente com a dimensão do espaço de entrada, sendo muito pequeno do que o correspondente à *Support Vector Machine*.

Por sua vez, no artigo [FR$^+$2010b] intitulado"*A Parallel Perceptron network for classification with direct calculation of the weights optimizing error and margin*", realizamos um estudo com a rede *Parallel Perceptron* [ABM2008]. Esta rede neuronal simples que se demonstrou actuar como um aproximador universal de funções. O *Parallel Perceptron* pode ser treinado usando a regra *Parallel Delta* que tenta minimizar a distância entre as activações do *perceptron* e as decisões do *hyperplano*, de modo a aumentar a sua capacidade de generalização (seguindo os princípios da *Statistical Learning Theory* que se baseia na SVM). Neste estudo propomos uma expressão analítica *closed-form* no sentido de calcular sem iterações os pesos do *Parallel Perceptron* para tarefas de classificação. Os pesos calculados optimizam globalmente

uma função custo que toma simultaneamente em consideração os erros de treino e a margem do *perceptron*, similar à regra *P-Delta*.

Na nossa abordagem, denominada de *Direct Parallel Perceptron*, a sua complexidade computacional é linear ao número de entradas, sendo bastante interessante para problemas de elevada dimensionalidade. Por outro lado, o *Direct Parallel Perceptron* é competitivo face à *Support Vector Machine* e a outras abordagens (incluindo a regra *P-Delta*) para problemas de classificação com duas classes. No entanto, em oposição à maioria deles os parâmetros do *Direct Parallel Perceptron* não influencia os resultados. Além disso, a ausência de uma fase de treino iterativa dá ao *Direct Parallel Perceptron* a capacidade de ser utilizado de forma "*on-line*".

Com a definição formal do sistema evolutivo para a resolução de problemas é possível extrair a melhor informação para a resolução de problemas em ambientes em que a informação é incompleta. Neste sentido, o sistema permite quantificar a informação desconhecida, convertendo-a para informação conhecida possível de ser integrada em qualquer sistema computacional, como por exemplo as *Redes Neuronais Artificiais*. Neste contexto, para fins de classificação, o estudo [NS+2011] apresenta-se como exemplo desta contribuição.

3.4 Computação Evolucionária e Programação em Lógica

Nas últimas décadas vários investigadores têm estudado a integração entre o paradigma simbólico e o evolutivo [OC+1995] [Tve1997] [WN1994]. A *Inductive Logic Programming* [Mit1982] constitui um tópico central na área da aprendizagem automática. Tipicamente, neste tipo de aprendizagem o mecanismo de aprendizagem pode ser visto com um processo de pesquisa iniciado através de hipóteses iniciais e a pesquisa no espaço de hipóteses possíveis [Mit1982] para uma hipótese que se ajusta a um dado conjunto de exemplos.

Quando uma linguagem é usada para expressar exemplos, o conhecimento adquirido e as hipóteses da lógica de primeira ordem, a *Inductive Logic* é designada por *Inductive Logic Programming* [LD1995]. Vários estudos [WN1994] [BG1993] [WL1995] [TCM1998] centram-se na generalização e especialização de derivar a partir da *Inductive Logic Programming* para o desenvolvimento dos operadores genéticos na pesquisa

do espaço de procura de soluções.

Man Leung Wong et al. apresenta alguns estudos sobre a interligação da *Inductive Logic Programming* com o paradigma evolucionário [WL1995a] [WL1995b] [WL1997] [Won2001]. Criam uma *framework* para combinar a *Programação Genética* com a *Inductive Logic Programming* baseando-se em formalismos da gramática da lógica. Esta *framework* aplica gramáticas lógicas para controlar a evolução dos programas em várias linguagens de programação, representando informação contextual e conhecimento dependente do domínio.

Brian Ross [Ros2001] apresenta um estudo baseado na lógica e na *Programação Genética* com "*Definite Clause Translation Grammars*". Neste estudo apresenta um sistema da interligação dos dois paradigmas, no sentido de melhorar outros sistemas baseados em gramáticas quando usadas com a *Programação Genética*, permitindo caracterizar aspectos semânticos não triviais da linguagem, sendo definido com gramáticas neste seu sistema. Por outro lado, este sistema tem a particularidade de determinar a profundidade mínima e as características de término na criação das representações em árvore.

Um dos trabalhos com grande relevância na integração dos paradigmas simbólico e evolutivo é o de *Frederico Divina* [DM2001] [Div2004], o qual estuda e explora as potencialidades dos *Algoritmos Evolucionários* na geração de programas usando a lógica de primeira ordem, em particular usando a *Inductive Logic Programming*.

Por sua vez, no âmbito da integração dos sistemas simbólico e evolutivo, vários sistemas têm sido apresentados como: o *GELOG* – um sistema que combina os *Algoritmos Genéticos* com a *Inductive Logic Programming* [Kok2001]; o *REGAL* [BG1993] um sistema baseado nos *Algoritmos Genéticos* para a aprendizagem de conceitos da lógica de primeira ordem; *DOGMA* [Hek1998], *G-NET* [AB2002], entre outros. Estes e outros sistemas são avaliados e comparados no estudo de *Divida* [Div2001].

Em [HW2009] poderemos encontrar vários estudos relacionados com a integração da lógica com sistemas evolutivos, nomeadamente, experiências usando lógica na área da bioinformática, ferramentas e sistemas orientados para as linguagens declarativas, assim como estudos recentes da aplicabilidade da *Programação em Lógica* com outros paradigmas, como os sistemas multi-agente, lógica probabilística, raciocínio abdutivo, entre outros.

Ainda nesta área da integração de paradigmas, convém referir que ao longo das

últimas décadas, outros trabalhos têm sido apresentados na integração do modelo evolucionário e conexionista, sendo denominado por *Redes Neuronais Evolucionárias* [Yao1999] [CRN2001] [RCN2004] [PK2005] [G+2008] [KC2008], em particular para treinar as redes (e os pesos das conexões), desenhar a sua arquitectura [SWE1992] [Bra1995] [B+2001] [CA+2002] [Abra2004] [Nit2007] [Nit2008] [FDM2008] [AP+2010], como abordagem à resolução de problemas complexos [Spe1990] [QS+2009], assim como na criação de mecanismos de inferência usando os sistemas *Fuzzy* [Kas1998].

3.5 Sistemas Conexionistas e Programação em Lógica

As *Redes Neuronais Artificiais* [Hay1998] correspondem a um paradigma computacional orientado para o processamento de informação inspirado na forma como o cérebro humano processa a informação. Estes modelos conexionistas têm sido aplicados com sucesso em várias áreas do conhecimento [WRL1994] e na sua interligação com os sistemas evolutivos [SKS2007], em especial na parametrização dos dados de entrada e dos pesos das conexões da estrutura das redes.

Ao longo das últimas décadas, a interligação entre sistemas simbólicos e conexionistas tem vindo a ser explorados [Ahm1996] [RJJ1996] [BS2001] [HHS2004] [HP2007] [Nit2007] [KM2008] [Nit2008] [FDM2008] [QS+2009] [GB+2010], sendo estes estudos conhecidos como investigações na área da ciência *neuro-simbólica* [HP2007].

Pascal Hitzler [HHS2004] apresenta um estudo sobre a semântica dos operadores dos programas em lógica proposicional, em que esses operadores podem ser computados pelas redes conexionistas. Concluem que a mesma semântica dos operadores para a lógica de primeira ordem pode ser aproximada pelas redes conexionistas *feedforword* [Hay1998]. *Wan Abdullah* [Abd1991] propôs um método para criar programas lógicos nas redes *Hopfield* [Hop1985]. O seu objectivo centrou-se na optimização da consistência lógica obtida a partir da rede depois dos pesos das ligações serem definidas a partir de programas lógicos. *Hideya Kawahara et al.* [IM+1993] e posteriormente *Takeda et al.* [TK+2004] apresentam um estudo sobre a criação de um mecanismo de inferência para a *Programação em Lógica* usando as redes *Hopfield*. Propõe um algoritmo que converte a rede dos operadores lógicos em redes *Hopfield* aumentando o poder de optimização destas redes, no sentido de serem utilizadas nas inferências lógicas usando o *PROLOG*. *Jimmy Lee* e *W. Tom* [LT1994]

[LT1995] apresentam uma *framework* para integrar os paradigmas conexionistas e lógico (*constraint logic programming*) para a resolução das *constraints satisfaction problems*. *Alexandros Chortaras et al.* [CS+2006] apresentam o estudo de um modelo conexionista aplicado aos pesos dos programas *Fuzzy Logic*. Neste estudo estendem a programação *Fuzzy Logic* com pesos, no sentido de as aproximar à semântica das *Redes Neuronais Artificiais*, propondo uma implementação das redes capaz de computar o modelo mínimo de *Herbrand* [Bus1995] de um programa *Fuzzy logic* com pesos que compõem os átomos do corpo da regra lógica.

Kolman e Margaliot [KM2008] apresentam um estudo da integração dos dois paradigmas de modo a extrair conhecimento simbólico a partir das redes neuronais recorrentes usando a lógica *fuzzy* como formalismo lógico.

Artur d'Avila-Garcez apresenta também vários estudos sobre esta interligação [dA+2001] [dA+2002] [dA+2007]. Em [GB+2010] apresentam o sistema *PAN – Predicate Association Network* que tem como objectivo utilizar as *Redes Neuronais Artificiais* para aprender relações entre programas lógicos. O seu estudo é baseado nos fundamentos do raciocínio indutivo para extraír conhecimento nas relações dos programas em lógica de primeira ordem usando as redes. Adicionalmente, apresentam uma alternativa às técnicas de *Inductive Logic Programming* quando utilizadas com as *Redes Neuronais Artificiais*, no sentido de associarem as redes a uma lógica de primeira ordem mais expressiva.

Saratha Sathasivam [Sat2009] seguindo os seus trabalhos [Sat2007] [SA2008] apresentam estudos utilizando as redes *Hopfield* e a programação em lógica. Utilizam o conceito de "*sign constraint*" para melhorar o desempenho das redes ajustando o peso durante o processo de geração e a avaliação da energia da rede e, desta forma, podem melhorar o critério de convergência das redes *Hopfield* para se efectuar a integração dos paradigmas simbólico e conexionista.

3.6 Conclusão

Neste capítulo apresentamos de uma forma geral os conceitos associados à *Inteligência Evolucionária* nomeadamente à utilização do paradigma evolutivo e conexionista. Ao longo dos anos, cada uma das técnicas associadas a estes paradigmas foram melhoradas, testadas e aplicadas com sucesso a inúmeras áreas do conhecimento. Por

outro lado, as vantagens associadas a cada um destes paradigmas foram aproveitadas e interligadas com outros paradigmas computacionais, entre elas o paradigma lógico através da *Programação em Lógica*.

A maioria dos estudos de integração do paradigma evolutivo e conexionista tiveram como objectivo treinar as redes e os pesos das conexões, no desenho de arquitecturas [Yao1999] [Abra2004] [RCN2004] [Nit2007] [Nit2008] [FDM2008] [SM2010], na resolução de problemas complexos [QS$^+$2009] e na criação de mecanismos de inferência usando os sistemas *Fuzzy* [Kas1998].

Outros estudos centraram-se na utilização destes paradigmas para a geração de programas lógicos, assim como na validação da sua consistência em termos de execução, sendo a grande maioria dos estudos centrados na interligação do paradigma evolutivo com a *Inductive Logic Programming*.

Por sua vez, os estudos sobre a interligação dos paradigmas conexionista e simbólico centraram-se na criação de estruturas de rede para a criação de mecanismos de inferência. Tal como na integração dos sistemas evolutivos com o simbólico, a maioria dos estudos de interligação dos sistemas conexionistas e simbólico centraram-se na utilização das redes *Hopfield* e na exploração das *Redes Neuronais Artificiais* com a *Inductive Logic Programming* [Abd1991] [TK$^+$2004] [CS$^+$2006] [dA$^+$2007] [KM2008] [Sat2009] [GB$^+$2010].

Contudo, neste trabalho de doutoramento o âmbito de utilização e interligação dos três paradigmas é bastante diferente dos trabalhos efectuados na área. Não estamos interessados apenas na geração automática de programas lógicos, mas sim, na geração de programas lógicos baseados em contexto medidos por medidas de quantificação associadas a esses programas lógicos. Por outro lado, *as Redes Neuronais Artificiais* serão utilizadas como arquitectura geral num processo de natureza evolucionário (*evolutionary network*) [Abra2004] [Nit2008] [FDM2008], sendo criadas topologias de rede dinâmicas. Com este sistema evolutivo serão obtidos as melhores teorias lógico matemáticas para a resolução de um determinado problema em particular em cenários em que a informação é incompleta. Desta forma, o sistema disponibilizará e quantificará a melhor informação na resolução de problemas.

Adicionalmente, conforme referimos na secção 3.3.4, ao longo desta tese de doutoramento, realizámos três trabalhos de investigação na área das *Redes Neuronais Artificiais*, em particular para problemas de classificação [FRM$^+$2010a] [FRM$^+$2010b] [NS$^+$2011].

Com a elaboração do sistema evolutivo e como contribuição complementar, o sistema criado irá permitir seleccionar a melhor informação para alimentar o mecanismo computacional das *Redes Neuronais Artificiais*, apresentando à rede informação quantificada em cenários em que a informação é incompleta [NS+2011].

Nesta nossa abordagem, em vez de atribuir uma probabilidade à informação desconhecida é atribuído um valor de verdade associado a essa informação, permitindo desta forma submeter à rede informação quantificável.

"We have here no revolutionary act but the natural continuation of a line that can be traced through centuries."
[Einstein in His own words, Anne Rooney,2006]

Capítulo 4

Inteligência Evolucionária - Modelação Computacional

Neste capítulo descrevemos o sistema *híbrido* evolucionário que nos propusemos, no qual tiramos partido das vantagens dos paradigmas simbólico, conexionista e evolutivo. Este trabalho tem em vista o uso de metodologias de resolução de problemas e dos formalismos e técnicas associadas ao uso dos *Algoritmos Genéticos* (em especial a *Programação Genética*) e as *Redes Neuronais Artificiais* (como esquemático arquitectural) para a criação de sistemas evolutivos que materializem a *Representação do Conhecimento* e formas de *Raciocínio Simbólico*, o que nos é dado através da *Programação em Lógica Estendida* [NM$^+$1997] [NM$^+$2007].

As *Redes Neuronais Artificiais* (evolutivas) [FDM2008] terão como principal objectivo tirar partido da capacidade de esquematizar de uma forma arquitectural o intelecto evolutivo associado ao sistema. Este sistema é baseado nos formalismos associados à representação simbólica tratando e quantificando o conhecimento adquirido em ambientes em que a informação é incompleta, irá permitir usar as potencialidades das *Redes Neuronais Artificiais* no que se refere à preparação da informação no sentido de as treinar.

Iniciamos este capítulo com uma introdução aos conceitos e temáticas envolvidas (secção 4.1), sendo apresentado de seguida uma formalização do tipo de representação de conhecimento e mecanismo de raciocínio que cada entidade (neurónio da rede neuronal) irá incorporar (secção 4.2), assim como a função de avaliação que quantificará e avaliará o desempenho de cada indivíduo (secção 4.3). Na secção 4.4 abordare-

mos os formalismos associados à *Inteligência Evolucionária*, quer em termos da sua representação, quer em termos do processo de inferência e processo evolutivo. Posteriormente no capítulo cinco, iremo-nos debruçar sobre a implementação de todo o sistema descrito neste capítulo.

4.1 Motivação

Várias abordagens têm sido apresentadas para modelar o raciocínio e o comportamento humano através da simulação computacional. O ser humano tem vindo a usar computadores para melhorar o desempenho das suas tarefas, no entanto, existem actividades que requerem análise, planeamento, tácticas e estratégias e características sobre as quais correspondem a manifestações de inteligência, sendo tipicamente estas actividades de difícil transcrição para os sistemas computacionais. Cada vez mais, fruto da evolução tecnológica, os algoritmos computacionais tornaram-se mais complexos e mais rápidos em termos de processamento, no entanto, não tem trazido melhoramentos em termos de inteligência.

Inúmeros investigadores da área da *Inteligência Artificial* tem desenvolvido potentes sistemas inteligentes, contudo, muitos destes sistemas tem uma importante limitação: os seus procedimentos e objectivos têm de ser bem definidos e não podem ser alterados. Na realidade, os sistemas inteligentes devem ser capazes de se adaptarem a novos ambientes com caracteristicas imprevisíveis, assim como aprender por eles próprios como decidir o que fazer.

Neste sentido, estes estudos apenas têm sido capazes de criar computadores e sistemas que representam ou mapeiam alguns dos comportamentos inteligentes dos seres humanos. De modo a criar sistemas que poderão realmente entender o ser humano, torna-se necessário estudar formas de duplicação muito mais que apenas alguma coisa sobre a forma de como o cérebro humano trabalha, mas sim, duplicar a forma de como o fazem.

Alguns progressos têm vindo a ser feitos na análise da forma como o cérebro humano trabalha, assim como no seu mapeamento para mecanismos computacionais. Os sistemas das *Redes Neuronais Artificiais* tem tido uma grande aceitação e sucesso na aplicabilidade da correlação dos seus algoritmos, em especial, na forma como determinam os melhores mapeamentos entre diferentes conjuntos de dados. São vistos

como simuladores do cérebro humano mapeados em sistemas espertos, mas não são inteligentes mais do que isso. Tipicamente as *Redes Neuronais Artificiais* consistem num conjunto de elementos de processamento conectados e distribuídos por camadas, caracterizadas por incluir uma camada de entrada, uma ou mais camadas intermédias e uma de saída. Cada camada está interconectada com outras, sendo o processamento do sinal transferido entre camadas segundo uma determinada topologia (por exemplo, *feedforward*). Neste sentido, as *Redes Neuronais Artificiais* não apresentam mais complexidade do que isto. O principal obstáculo no desenvolvimento de sistemas artificiais com inteligência do cérebro é o facto de ninguém saber realmente o que torna o cérebro inteligente.

No último século, vários investigadores debruçaram-se sobre o estudo do comportamento do cérebro humano, em áreas que vão desde a filosofia, a psicologia, a biologia, a neurociência, entre outras, não tendo sido encontradas soluções na produção de várias teorias de sucesso de como realmente a mente humana emerge a partir da forma de trabalhar do cérebro humano, em especial na pesquisa de como o cérebro humano alcança a inteligência.

Vários cientistas têm desenvolvido métodos associados à evolução artificial baseados na evolução da teoria das espécies, nomeadamente a *Computação Evolucionária* [ES2003]. Estes algoritmos têm sido aplicados com sucesso numa larga variedade de domínios científicos, em especial, em processos de optimização aplicados a problemas complexos (ver capítulo 3).

Por outro lado, as decisões do ser humano assentam em informação que grande parte das vezes, se não toda, é imprecisa, incompleta ou mesmo omissa. O paradigma simbólico materializado pela *Programação em Lógica Estendida*, tem vindo a demonstrar-se como uma ferramenta poderosa para a representação do conhecimento e mecanismos de raciocínio em particular, quando se pretende resolver problemas em cenários em que a informação é imprecisa, nebulosa, ou imperfeita.

Face ao exposto, consideramos que deve ser dado algum salto qualitativo na área da *Inteligência Artificial*, no sentido de tirar partido das vantagens associadas a estes três paradigmas (simbólico, evolutivo e conexionista). Desta forma, os métodos clássicos tornar-se-ão mais capazes de serem combinados com outros processos de forma a dar alma à criatividade computacional do uso dos paradigmas e deste modo, criar sistemas mais inteligentes capazes de simular a inteligência do cérebro humano na resolução de problemas.

Neste contexto, a motivação deste trabalho de doutoramento centra-se em estudar a criação de um mundo dinâmico virtual de interacção entre genes, genomas, organismos e população que competem uns com os outros num regime rigoroso de selecção, em que a função de avaliação é baseada num único critério: a inteligência das entidades, medida através da qualidade da informação associada ao conhecimento que cada entidade possui num determinado momento do seu estado evolutivo. No entanto, temos de ter em consideração que um sistema evolucionário orientado para a tarefa da criatividade de inteligência apresenta vários desafios. Num típico sistema de *Computação Evolucionária* a estrutura básica do espaço de soluções é bem definido à partida, sendo que o processo evolucionário apenas necessita de fazer pequenos ajustamentos ao conjunto de parâmetros predefinidos. Por exemplo, a evolução do algoritmo genético de uma *Rede Neuronal Artificial* consiste apenas em encontrar valores praticáveis para os pesos das conexões e para o número de neurónios das camadas intermédias.

Contudo, um sistema evolucionário artificial orientado para o poder criativo da evolução biológica tem de ser capaz de incorporar novos tipos de parâmetros e estruturas. Neste sentido, a maior parte dos problemas da *Computação Evolucionária* são bem definidos, sendo as comparações de desempenho efectuadas através da avaliação da competitividade dos indivíduos da população. No entanto, a selecção dos indivíduos com características gerais e abstractas como a inteligência apresenta algumas dificuldades em termos de desempenho das métricas de avaliação desses indivíduos. Neste sentido, esses indivíduos tem de ser testados na sua capacidade de se adaptarem à alteração do ambiente, fazer deduções e desencadearem inferências, no sentido de seleccionar o melhor percurso de acção a partir de um intervalo de alternativas.

Neste trabalho de doutoramento a avaliação dos indivíduos será quantificada pela qualidade da informação associada ao conhecimento de uma entidade, materializada por teorias ou programas lógicos representados através da *Programação em Lógica Estendida*. O sistema evolutivo tirará partido das vantagens associadas aos fundamentos dos *Algoritmos Genéticos*, em particular a *Programação Genética*, sendo criadas redes evolucionárias dinâmicas que expressam as melhores teorias lógico matemáticas para a resolução de um determinado problema.

4.2 Representação do Conhecimento

4.2.1 Introdução

Uma importante característica de todos os sistemas naturais é a capacidade de adquirir conhecimento através de experiências e de se adaptar a novas soluções. Em geral, derivar conclusões gerais a partir de observações específicas é designado por indução. Neste contexto, a aprendizagem pode ser vista como um problema de pesquisa no espaço de todas as hipóteses possíveis [Mit1982]. Quando se pretende resolver um problema através de mecanismos computacionais, temos em primeiro lugar de transcrever o problema para termos computacionais (ou formais). Neste sentido, a escolha da linguagem para efectuar a representação de hipóteses varia de um fragmento do cálculo proposicional até à lógica de segunda ordem. Enquanto que, a primeira tem um reduzido poder de abstracção, a lógica de segunda ordem é mais complexa e por este facto é raramente usada.

As linguagens de representação terão de ser escolhidas de modo a representar conceitos, exemplos e conhecimento adquirido. No entanto, com uma linguagem de representação que tem um baixo poder de expressão, poderemos não ser capazes de representar certos domínios dos problemas, uma vez que se torna bastante complexo para a linguagem adoptada. Por outro lado, linguagens muito expressivas poderão nos dar a possibilidade de representar todos os domínios do problema. Contudo, esta solução poderá dar muita liberdade, no sentido de que poderemos construir conceitos complexos sobre várias formas, o que nos poderá conduzir para a impossibilidade de encontrar o conceito correcto.

Neste sentido, a *Representação do Conhecimento* como forma de descrever o mundo real baseada em formas mecânicas, lógicas ou outras, será sempre em função da capacidade dos sistemas, de modo a descrever o conhecimento existente e os seus associados mecanismos de raciocínio. De facto, na concepção de sistemas de representação do conhecimento, tem de ser tida em conta as diferentes instâncias do conhecimento [NA2000], nomeadamente:

- *Conhecimento existente*: não será conhecido em todas as extensões, uma vez que caracteriza todas as circunstâncias do *universo do discurso*, nomeadamente informação conhecida e desconhecida;

- *Conhecimento observado*: adquirido por experiência;
- *Conhecimento representado*: com respeito a um certo objectivo, poderá ser irrelevante de representar um dado conjunto de dados. Neste sentido, esta é a informação que deverá ser representada e entendida.

De um ponto de vista computacional, a ambiguidade semântica, juntamente com a complexidade gramatical e léxica das linguagens naturais (*e.g.*, Português), tornam difícil a sua utilização para a descrição formal de conhecimento. Por outro lado, para a representação de conhecimento em sistemas computacionais recorre-se não menos vezes à utilização de linguagens lógicas. De uma maneira geral uma *Linguagem Lógica* é composta por [Woo1992] [RN1995]:

- uma *sintaxe*, que define os objectos de discurso;
- uma *semântica*, que atribui um significado a cada objecto sintáctico; e
- uma *teoria de prova* [LN1994] [Kow1995], que define as manipulações sobre os objectos sintácticos.

Em suma, das várias linguagens lógicas existentes destacam-se a *Lógica Proposicional* e a *Lógica de Predicados de 1ª Ordem*, como mecanismos de representação de conhecimento acerca do mundo real. As linguagens lógicas permitem uma descrição formal e não ambígua de factos, que podem ser verificados e validados formalmente [Lig1997]. A lógica proposicional tem um poder de expressão relativamente simples, permitindo apenas trabalhar com factos do mundo real. Por outro lado, a lógica de predicados de 1ª ordem tem a particularidade de permitir trabalhar com factos e relações entre objectos do mundo real. A *Lógica Proposicional* formaliza a estrutura lógica mais elementar do discurso matemático, sendo uma linguagem que permite o tratamento de expressões lógicas simples sobre a sua forma abstracta.

4.2.2 Trabalho Relacionado

Várias técnicas não clássicas foram propostas para modelar o *universo do discurso* e procedimentos de raciocínio de sistemas inteligentes, como por exemplo [KK1998] [She1991] [NM+1997] [Sub2001] [BK2005] [PL2007], podendo várias delas serem ligadas à lógica com a teoria das probabilidades [LS1997], combinando o raciocínio

Bayesiano [DHN1976], lógicas *multivalor* [Loy2004], teoria da evidência (*Dempster-Shafer*) [Sha1992], lógica *fuzzy* [Zad2001], formalismos híbridos (i.e numéricos e não numéricos) e outras metodologias não *standards*. Nos últimos séculos, a *Lógica* e os *Programas em Lógica* emergiram como um formalismo atractivo para a representação do conhecimento e mecanismos de raciocínio, i.e., apresentou-se como um mecanismo eficiente para resolver problemas de pesquisa.

Por outro lado, a *Programação em Lógica Abdutiva* [KK1998] é um promissor paradigma computacional, tendo vindo a ser reconhecido como uma forma de resolver algumas limitações da programação em lógica, no que respeita à representação do conhecimento em alto nível, assim como para tarefas de raciocínio. Muitos trabalhos relacionados com a representação do conhecimento e metodologias de resolução de problemas utilizam os formalismos associados à lógica matemática, como a teoria dos modelos e a teoria da prova [LN1994].

Pereira et al. [PP2005] baseia o seu trabalho na teoria dos modelos e apresenta uma abordagem à *Programação em Lógica*, representando o conhecimento usando a semântica da teoria dos modelos (*stable models* [GL1998]). Estes investigadores, considerando que a *Programação em Lógica* é um formalismo robusto como linguagem de representação para conhecimento estático, detectaram algumas limitações na sua aplicação em ambientes dinâmicos, em especial quando se pretende integrar actualizações de conhecimento a partir de recursos externos. Os mesmos autores, apresentaram o *DLP – Dynamic Logic Programming* [ALP$^+$1998], no sentido de representar informação temporal e actualizações em ambientes dinâmicos, assim como representar informação negativa [PW1990]. Também apresentam o *Generalized Logic Programs* [AHP2000] de forma a lidar com a negação forte. No sentido de formalizar estes estudos foi criado o *LUPS – Language for Updating ProgramS* [AP$^+$1999] de modo a implementar o conceito DLP.

Neste contexto, e na aplicabilidade do *LUPS* em ambientes com a participação de agentes, os mesmos autores desenvolveram o sistema *MINERVA* [LAP2002], uma *framework* de agentes que, considerando a linguagem *LUPS*, permite a combinação de várias representações não monótonas e mecanismos de raciocínio. Apesar das vantagens do *LUPS*, algumas limitações foram encontradas em termos da alteração dos estados dos agentes durante o tempo. Neste sentido, os mesmos autores apresentaram o *EVOLP – Evolving Logic Programs* [AB$^+$2002] que generaliza a *Programação em Lógica*, permitindo especificar a evolução dos próprios programas. Baseado na

aplicação do *EVOLP* em ambientes de agentes, considerando a necessidade de activar um programa (*evolving program*) e, no sentido de "*look ahead prospectively*" é possível prever futuros estados de um agente empregando metodologias associadas à *Abduction in Logic Programming – ALP* [KK1998] [DK2002]. Com base neste paradigma, apresentam também o *Prospective Logic Programming* [LP2006], considerando preferências e restrições no sentido de conduzir os estados futuros de um agente, minimizando a explosão da combinação futura que um agente pode seguir. No mesmo estudo, apresentam a arquitectura *ACORDA* que corresponde a uma implementação da plataforma *EVOLP* acrescida pela utilização de precedências e restrições baseadas na abdução, aplicando estes conceitos à arquitectura *ACORDA* em ambientes de agentes [PL2007]. Em [AB$^+$2003] e [PS2007] os autores apresentam um estudo sobre a modelação da característica da moralidade usando a metodologia *PLP - Prospective Logic Programming.*

Os estudos recentes destes autores [PS2007] [PA2009] [PDL2009] centram-se no problema da decisão do estado dos agentes quando confrontados com percursos diferentes da evolução (ou decisões). Neles dão especial atenção à especificação de alguns níveis de comprometimentos (*commitments*) e preferências, no sentido de conduzir as decisões para um determinado objectivo sobre a *ALP*. Em [PA2009] actualizaram a arquitectura *ACORDA*, de modo a contemplar "*causation probabilistic*"(*Causal Bayes Nets*) [PR2009] para a quantificação e qualificação das preferências envolvidas na *PLP*. Por outro lado, tentam interligar a programação "*prospective*"e a probabilística, no sentido de potenciar a qualidade e quantidade das preferências.

Numa outra orientação da investigação, *Neves et. al.* [MA$^+$2006] baseiam o seu trabalho no mecanismo da *teoria da prova* [Nev1984], no sentido de representar o conhecimento (e.g. *universo do discurso*), assim como em criar mecanismos de inferir conhecimento em raciocínio não monótono [EST2005] com informação incompleta [NA$^+$2004]. Em comparação com a teoria dos modelos, utilizando o mecanismo baseado na prova de teoremas, um modelo em [PL2007] pode ser entendido na teoria da prova como uma composição de predicados que denotam os objectos e as suas relações que possam ser estabelecidas entre elas, no sentido em que essas relações modelam o *universo do discurso*. Esta metodologia (teoria da prova) é aplicada a ambientes dinâmicos de agentes [MA$^+$2006] [MA$^+$2008] [MM$^+$2009]. Os autores apresentam uma extensão aos seus estudos apresentando o conceito de *Qualidade da Informação* [AN$^+$2006] [MA$^+$2006a] associada a um programa em Lógica.

Este conceito foi aplicado com sucesso em vários dominios nomeadamente, [MA$^+$ 2006] [LC$^+$2008] [MM$^+$2009] [RN$^+$2009] [MA$^+$2010] [NA$^+$2010] [LN$^+$2009] [NS$^+$2010]. No trabalho [NM$^+$2007], os autores apresentam a ligação entre a *Programação Genética* e a *Programação em Lógica*. Esta interligação entre a *Lógica* e os *Algoritmos Evolucionários*, pode ser vista como uma abordagem aos programas ou teorias lógicas evolucionárias, sendo as soluções candidatas preservadas e testadas quando uma solução é óptima baseada na medida da qualidade da informação associada pelos programas ou teorias lógicas.

Comparando com o conceito de abdução, *Neves et al.* [NM$^+$2007] assume que um *abducible* pode ser visto como hipóteses que fornecem soluções possíveis ou explicações para determinadas questões, sendo representadas sobre a forma de excepções das extensões dos predicados que formam o programa lógico.

4.2.3 Representação da Informação e Mecanismo de Raciocínio

Cada base de conhecimento de cada entidade[1] é tida a partir da teoria ordenada *(T, <)*, onde T é um conjunto de regras (premissas) e "$<$" é uma ordem não circular sobre T. A necessidade para esta teoria de ordem circular prende-se por duas razões: a importância relativa das regras (uma regra é escolhida face a outra regra) e a usabilidade computacional (um programa lógico escrito numa linguagem como o *PROLOG* [CM1981] [Brat1990] [SS1994] necessita de alguma concreta ordem sobre o conjunto de cláusulas).

A teoria ordenada *(T,<)* pode ser estendida para incorporar a introdução, dentro da base de conhecimento de cada entidade, de regras que expressam prioridades. Esta teoria ordenada não circular é representada por $TN = (T, <, (S, \prec))$, onde T é um conjunto de regras que expressam o conhecimento (experiências prévias ou hipóteses), "$<$" é uma ordem não circular sobre T e $(S, \prec)$ representa um conjunto de regras de prioridade e a sua ordem relativa.

Tradicionalmente, os programas em *Programação em Lógica* são restritos na sua semântica de lógica de dois valores. Devido ao *Pressuposto do Mundo Fechado* [Hus1994], tudo o que não é considerado como verdadeiro é considerado falso. Desta

[1] No contexto deste trabalho de doutoramento, quando nos referimos a uma entidade, referimo-nos a um neurónio da rede arquitectural do sistema evolutivo.

forma, esses programas reagem apenas para a existência de informação positiva na base de conhecimento de uma entidade. Uma ênfase especial deve ser dada para a representação de informação desconhecida ou incompleta. A extensão à *Programação em Lógica* é extremamente flexível em tratar informação positiva, negativa e desconhecida (ou incompleta). Neste sentido, uma nova estrutura de cláusulas de conhecimento deve ser definida:

Definição 3 – negação por falha: *A fórmula not P_i, com P_i um literal (i.e., uma fórmula com a forma p ou $\neg p$, onde p é um átomo), representa a negação por falha* ***(negation-as-failure****); i.e., significa que não é possível de provar que P_i é verdadeiro.*

Definição 4 – Cláusulas de conhecimento: *O Conhecimento de cada entidade é expresso por um conjunto de cláusulas da forma r_k:$\forall_{x1}, \ldots, \forall_{xn}$ $^{\varphi}$, onde $^{\varphi}$ é da forma $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_{i-1} \vee \neg not\ P_i \vee \ldots \vee \neg not\ P_{i+1} \vee P_{i+j+1}$ excepto para a disjunção comutativa, onde se $i = j = 0$, $^{\varphi} = \perp$ e $P_1, \ldots, P_{i+j+1}$ são átomos.*

Notação 1: *Cláusulas da forma $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_{i-1} \vee \neg not\ P_i \vee \neg not\ P_{i+j} \vee \neg not\ P_{i+j+1}$, excepto para a comutatividade, são escritas como r_k: $P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge \ldots \wedge P_{i-1} \wedge not\ P_i \wedge \ldots \wedge not\ P_{i+j}$. Estas cláusulas são designadas por "regras", P_{i+j+1} é designado por "consequente" e $P_1 \wedge P_2 \wedge \ldots \wedge P_{i-1} \wedge not\ P_i \wedge \ldots \wedge not\ P_{i+j}$ é designado por "antecedente".*

Notação 2: *Cláusulas da forma r_i: $P_1 \leftarrow$ são representadas como $r_i : P_1$. Estas cláusulas são designadas por "****factos****".*

Definição 5 – (extensão às cláusulas de conhecimento): *As cláusulas de conhecimento da forma $r_k : P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge \ldots \wedge P_{i-1} \wedge not\ P_i \wedge \ldots \wedge not\ P_{i+j}$, onde $k \in N_0$ e $P_1, \ldots, P_{i+j+1}$ são átomos, tornando-se uma extensão às cláusulas de conhecimento da mesma forma quando $P_1, \ldots, P_{i+j+1}$ são literais, i.e., fórmulas da forma p ou $\neg p$, onde p é um átomo, e onde r_k, not, pi+j+1, e $P_1 \wedge P_2 \wedge \ldots P_{i-1} \wedge not\ P_{i+j}$ referem-se, respectivamente, ao identificador da cláusula, ao operador da negação por falha, o consequente da regra e o antecedente da regra. Se $i = j = 0$, a cláusula é designada por "****facto****" e é representado por r_k:P_1.*

Definição 6 – (Programa em Lógica Estendida) – Um *Programa em Lógica Estendida* é um conjunto (possivelmente infinito) de fórmulas segundo o modelo (9) [GL1990], onde cada um dos literais P_i é uma fórmula atómica de primeira ordem ou a sua negação explícita ou clássica (*i.e.*, A ou $\neg A$) e '*not*' representa a negação por falha [Bre1996] [NA2000]. Este programa pode ser representado da seguinte forma:

$$q \leftarrow p_1 \wedge ... \wedge p_m \wedge not\ p_{m+1} \wedge ... \wedge not\ p_{m+n} \quad (9)$$

$$?\, p_1 \wedge ... \wedge p_m \wedge not\ p_{m+1} \wedge ... \wedge not\ p_{m+n} \quad (10)$$

onde $?$ é o átomo do domínio denotando falsidade, e q e todo o p_i são literais i.e. fórmulas do tipo A ou $\neg A$, sendo A um átomo. Se q é uma fórmula vazia então, a fórmula (9) corresponde a um *invariante*, i.e., uma ***restrição de integridade***.

A *Programação em Lógica Estendida* introduz um outro tipo de negação: negação forte, representada pelo sinal de negação clássica "¬". Na maioria das situações, é útil representar $\neg A$ como um *literal* se é possível de provar $\neg A$. Na *Programação em Lógica Estendida* a expressão A e *not* A, sendo A um literal, são literais estendidos, enquanto que A ou $\neg A$ são literais simples. Intuitivamente *not* p é verdadeiro sempre que não houver razão de obter p, onde $\neg p$ requer a prova do literal negado.

A cada programa está associado um conjunto de *abducibles*. Os *abducibles* podem ser vistos como hipóteses que fornecem possíveis soluções ou explicações para determinadas questões, sendo dadas aqui sobe a forma de excepções às extensões dos predicados que constituem o programa. O objectivo é provar o poder expressivo da representação explícita da negação informativa, assim como descrever directamente o *Pressuposto do Mundo Fechado* para alguns predicados, também conhecido como predicados de circunscrição [Nev1984] [AR1994].

Neste sentido, três tipos de respostas para uma dada questão são possíveis: verdadeiro, falso e desconhecido. A representação de valores desconhecidos pode ser dividida em três tipos:

- *Tipo 1* – valores desconhecidos, não necessariamente a partir de um conjunto de valores; (11)

- *Tipo 2* – valores desconhecidos, mas possíveis de serem obtidos a partir de um conjunto de valores, podendo ser seleccionados em combinação uns com os outros; (12)

- *Tipo 3* – valores desconhecidos, mas possíveis de serem obtidos a partir de um conjunto de valores, pondendo ser escolhido apenas um. Neste caso, na representação da informação terá de haver um invariante para assegurar esta restrição. (13)

Um dos princípios básicos da *Programação em Lógica* passa pela definição do *Pressuposto do Mundo Fechado* para todos os predicados, ou seja, assumindo-se que *"tudo o que não se pode provar a partir de um programa é falso"*. Existem situações em que o que pretendemos representar pela ausência de um dado conhecimento, não é que ele seja desconhecido (como acontece na *Programação em Lógica Estendida*), mas que é exactamente que ele é falso. Assim, de forma a não perdermos nenhum poder de representação é possível tornar um programa em *Programação em Lógica Estendida* num programa em Lógica, definindo explicitamente o *Pressuposto do Mundo Fechado*, para todos os predicados $p(X)$, através da seguinte forma:

$$\neg \; p(X) \leftarrow not\ p(X)$$

A representação de atributos através de valores nulos permite-nos distinguir os casos em que o seu valor é conhecido (verdadeiro ou falso), daqueles em que essa informação não está disponível, pelo menos na sua totalidade. Um atributo representado por um ***nulo do tipo desconhecido***, quando o seu valor não é determinado e o conjunto dos seus valores possíveis poderá incluir objectos não presentes no *universo do discurso*. Assim, se o valor do atributo p para um dado objecto a é desconhecido, e não necessariamente contido num conjunto de valores, então esta informação pode representar-se da seguinte forma, utilizando o predicado *p(Objecto, Valor)* e a *Programação em Lógica Estendida*:

$$p(a,\underline{w})$$

Sendo w uma constante que representa um valor nulo do tipo desconhecido. De-

vemos em seguida, identificar de uma forma explícita todas as situações em que o conhecimento é determinado (verdadeiro ou falso) ou desconhecido. Neste sentido, criamos para o efeito um predicado designado *abducible* (ou *excepção*) que armazena os casos em que o predicado p tem o valor desconhecido. Neste caso, todas as questões à base de conhecimento inquirindo sobre o valor do predicado p para o objecto x, deverão ter resposta desconhecida (*valor desconhecido de tipo 1*):

$$abducible_p(X,\_) \leftarrow p(X,\underline{w})$$

Finalmente, resta especificar, para este predicado o *Pressuposto do Mundo Fechado*, ou seja, identificar os casos em que podemos afirmar que a informação é falsa:

$$\neg\ p(X,Y) \leftarrow not\ p(X,Y)\ \wedge \\ not\ abducible_p(X,Y)$$

Quando sabemos que o valor desconhecido que pretendemos representar, só pode tomar valores de um conjunto finito e determinado, a forma de representação em *Programação em Lógica Estendida* segue uma abordagem um pouco distinta. Assumindo que o atributo p tem para um objecto x, um conjunto de valores possíveis dados por $\{v_1,\ v_2,\ \ldots,\ v_n\}$, podemos representar essa informação da seguinte forma (i.e., para o *valor desconhecido de tipo 2)*:

$$p(x,\ \{\ v_1,\ v_2,\ \ldots,\ v_n,\})$$

Em relação à definição dos casos em que a informação é desconhecida, temos:

$$\{$$

$$abducible_p(x,\ v_1)\ \wedge$$

$$abducible_p(x,\ v_2)\ \wedge$$

$$.....$$

$$abducible_p(x,\ v_n)\ \wedge$$

$$\}$$

Neste caso, especifica-se que o valor de x para o predicado p é desconhecido, mas pode ser obtido através do conjunto de valores possíveis $\{v_1,\ v_2,\ \ldots,\ v_n\}$. Nesta situação, a selecção dos valores possíveis pode ser obtida através da combinação da

escolha dos valores, i.e., pode ser escolhido em simultâneo mais do que um valor. Os valores desconhecidos do *tipo 3*, são idênticos aos do *tipo 2*, no entanto, do conjunto de valores apenas pode ser seleccionado um valor e não mais do que um. Neste caso surge a necessidade de se especificar uma restrição (ou invariante), de modo a assegurar este propósito. Esta restrição pode ser definida da seguinte forma:

$$\begin{array}{l} \leftarrow \quad \neg \ ((abducible_p(X) \ \vee \ abducible_p \ (Y)) \ \wedge \\ \neg(\ abducible_p \ (X) \ \wedge \ abducible_p \ (Y)) \end{array} \qquad (14)$$

em que, a restrição (14) apresenta que para o predicado p o seu valor é desconhecido, mas, ou pode ser escolhido um ou outro valor e não os dois em simultâneo, i.e. assegurar o operador *XOR*. Neste sentido, a definição do *Pressuposto do Mundo Fechado* faz-se nos mesmos moldes apresentados no caso anterior.

Os ***valores nulos do tipo não permitido*** utilizam-se em casos em que não é permitido conhecer nem inserir um dado conhecimento na base de dados (ou base de conhecimento). O seu comportamento estático é idêntico aos dos valores nulos do tipo desconhecido enquanto que, o seu comportamento dinâmico implica a definição de ***invariantes***, i.e. de condições que devem manter-se verdadeiras na base de conhecimento, sempre que nova informação é inserida ou que algo é removido. Assim, se não for possível na base de conhecimento conheccer o valor de um dado atributo p, para o objecto a, podemos representar este facto da seguinte forma para a componente estática:

$$\begin{array}{l} \{ \\ p(a,np) \ \wedge \\ abducible_p(X,\_) \leftarrow \ p(X,np) \ \wedge \\ (\ \neg \ p(X,Y) \ \leftarrow \ not \ p(X,Y) \ \wedge \\ \qquad\qquad not \ abducible_p(X,Y)) \\ \} \end{array}$$

e para a componente dinâmica:

$$\leftarrow \ (p(X,np) \ \wedge \ p(X,Y) \ \wedge \ Y \neq np)$$

Face ao exposto, dado um programa P em *Programação em Lógica Estendida*, uma interpretação de uma questão Q deverá ser formalizada da seguinte forma (15):

- verdade, se é possível provar Q, no contexto da teoria subjacente ao programa P;

- falso, se é possível provar $\neg Q$, no contexto da teoria subjacente ao programa P;
- desconhecido, nos outros casos.

Ou seja, em termos práticos, considerando uma questão abstracta $q(X)$, em que X pode tomar a forma X_1, X_2, ..., X_n, definem-se as seguintes três respostas possíveis para essa questão:

- ***verdadeiro*** se $\exists X : q(X)$
- ***falso*** se $\exists X : \neg q(X)$
- ***desconhecido*** se $\neg\exists X : q(X) \vee \neg q(X)$

em que '*:*' é a notação para "*tal que*". Tendo em consideração o exposto, em termos da *Lógica Matemática*, a derivação formal de um novo teorema é denominado de prova ou demonstração. O nome do *axioma* é geralmente atribuído ao conjunto mínimo de teoremas que especificam a teoria. Utilizando um conjunto de regras de inferência é possível deduzir uma fórmula directamente a partir da técnica de redução ao absurdo [Kow1995].

Neste sentido, para raciocinar acerca de um conhecimento particular baseado nos formalismos apresentados anteriormente, consideremos um procedimento em termos da extensão à *Programação em Lógica*, baseado na teoria da prova (apresentada na secção 2.3.4). Este predicado (definição 7) permite disponibilizar mecanismos de raciocínio acerca do conhecimento apresentado num domínio em que dada uma questão é devolvida a solução baseada num conjunto de assumpções.

Definição 7 (meta-teorema *demo*): Um meta-teorema de resolução para informação incompleta representado pela assinatura: ***demo***$:T,V \rightarrow \{$*verdade, falso, desconhecido*$\}$ infere a avaliação V do teorema T em termos dos valores de *verdade*, verdadeiro (ou 1) ou *falso* (ou 0), ou desconhecido (com valores de verdade - "Truth values"entre 0 e 1) de acordo com o seguinte conjunto de produções:

demo(T, verdade) $\leftarrow$ T
demo(T, falso) $\leftarrow$ $\neg$ T

demo(T, desconhecido) $\leftarrow$ *not T, not* $\neg T$

em que a primeira cláusula estabelece que a resposta à questão recorre à base de conhecimento com informação positiva; a segunda cláusula determina que a questão revelasse falsa representando a informação negativa [PW1990] e o conhecimento representado a esse nível e, a terceira cláusula baseia-se no conceito de desconhecido ou informação incompleta. Estes elementos são átomos que representam conceitos abstractos sem qualquer definição, i.e., elementos que tem uma boa definição (*well-defined*) de intervalo de valores como opções válidas.

A utilização da *Programação em Lógica Estendida* com o meta-interpretador apresentado, permite a representação e manipulação de valores nulos, nomeadamente, nulos desconhecidos, nulos mutuamente exclusivos e nulos não permitidos. Um ***nulo do tipo desconhecido*** representa a falta de um ou mais itens de informação. Os *nulos do tipo desconhecido de um conjunto de valores* referem-se a informação para descrever situações para as quais se posicionam diversas alternativas (mutuamente exclusivas). Os **nulos do tipo não permitido** representam situações não permitidas na base de conhecimento de cada entidade.

Na secção seguinte iremos apresentar uma abordagem sobre a quantificação da informação em ambientes em que ela é incompleta, contemplando desta forma estes três tipos de nulos. Na secção 5.2 apresentamos a forma operacional de implementação usando a linguagem de programação *PROLOG*.

4.3 Qualidade da Informação

4.3.1 Trabalho Relacionado

No contexto da teoria qualitativa da incerteza (*Qualitative Uncertainty Theory*) [Hal2005] [KR2010], *Seridi and Akdag* [SA2000] apresentaram uma forma axiomática baseada na substituição de intervalos probabilísticos $[0 \ldots 1]$, com um conjunto ordenado de valores simbólicos para predicados lógicos multi-valor. Fazem uma aproximação ao problema da representação mas não à *Qualidade da Informação* desses predicados lógicos ou do seu conjunto. Noutro trabalho [Str2006], a teoria de *Dempster-Shafer* [Sha1992] é combinada com conjuntos *fuzzy*, de modo a representar separadamente informação incerta e imprecisa, no sentido de modelar por exemplo, processos

de diagnóstico médico. Os níveis de imprecisão permitidos são determinados por limites (*thresholds*) durante a formulação de regras na base de conhecimento, assim como em mecanismos de raciocínio. Estes limites são relacionados e o seu valor óptimo pode ser estimado enquanto processam os dados de treino disponíveis.

A teoria de *Dempster-Shafer* é mais uma vez usada em [HX2008] combinando com a *Analytic Hierarchy Process* [Saa2001], no sentido de construir um método de resolver problemas de tomada à decisão multi-atributo com informação incompleta. Este método resolve o problema directamente baseado no conceito da matriz de decisão incompleta. Identifica todos os elementos possíveis a partir da matriz de decisão incompleta, calculando depois a probabilidade básica atribuída a cada elemento local, sendo finalmente avaliada o intervalo de confiança para cada decisão.

Seguindo o trabalho de *Belnap* [Bel1977], *Ofer Arieli* e *Arnon Avron* [AA1998] sugerem que uma lógica de quatro valores é a melhor opção para lidar com informação inconsistente e incerta do que uma lógica de três valores para raciocínio computorizado. Concluem que a existência de elementos como a falta de informação ou "sem conhecimento" e inconsistência (ou "*over-knowledge*"), assim como a ideia de ordenar os dados de acordo com graus de conhecimento numa estrutura "*bilattice*" é particularmente apropriada para tratar raciocínio com incerteza.

Contudo, os modelos qualitativos e de raciocínio qualitativo tem vindo durante algum tempo a ser investigados na àrea da *Inteligência Artificial* [For1984] [WK1989] [Kui1994] [For1996], em particular na área da saúde e dado o aumento do número de diferentes opções de manuseamento dos pacientes, combinados com as expectativas das autoridades de saúde acerca do comportamento dos seus físicos quando efectuam o balanceamento dos benefícios para os seus pacientes com os custos financeiros Neste sentido, linhas médicas orientadoras são um exemplo do suporte à tomada à decisão em ambientes em que a informação é incompleta, assim como na verificação de quando uma linha orientadora entra em conformidade com aos requisitos médicos globais de qualidade é o principal objectivo.

Os trabalhos de *Lucas* [Luc2003] e *Hommersom* [HLB2004] são um bom exemplo da avaliação da qualidade usando lógica. Usam *abduction* [Poo1990] e lógica temporal para verificar a qualidade da orientação das linhas médicas, propondo um método para diagnosticar potenciais problemas numa determinada linha orientadora, tendo em consideração o cumprimento dos critérios médicos gerais num meta nível de caracterização. Adicionalmente, exploram uma aproximação que usa uma translação

relacional de um mapa de fórmulas lógicas temporais para lógica de primeira ordem, assim como usar um teorema de prova baseado na resolução [HLB2008].

Numa outra linha de investigação, o conceito da *Qualidade da Informação (QoI)* [AN+2006] demonstrou a sua aplicabilidade em vários ambientes dinâmicos de tomada à decisão, nomeadamente na área médica [MM+2009] [MA+2010], lei civil [NA+2010], Sistemas Multi-Agente [MM+2009], entidades virtuais [MA+2006], *Ambient Assisted Living* [LC+2008] e em ambientes de tomada à decisão [RN+2009] [LN+2009] [NS+2010]. Baseado nos princípios de *Neves et al.* [AN+2006], em termos da representação do conhecimento e mecanismo de raciocínio, *Machado et al.* [MA+2006] apresentam um estudo na área das entidades virtuais sobre a criação de modelos formais em serviços de contratualização baseados na *Web*. Neste estudo utilizam o conceito da *QoI* para avaliar a qualidade dos parâmetros como a veracidade e a reputação, que deve ser tida em consideração para a avaliar, certificar e justificar as decisões.

Marreiros et al. [MS+2007] apresentam um estudo sobre a tomada à decisão com base na argumentação em ambientes inteligentes, os quais utilizam a *QoI* na avaliação da informação associada à selecção dos oponentes através dos aspectos associados a cada membro do grupo, nomeadamente, o estado emocional, gratitude, credibilidade, entre outros.

Costa et al. [CN+2008] apresenta um estudo sobre a qualidade do serviço na prestação de cuidados médicos. Apresentam o *VirtualECare*, um sistema centrado na sustentabilidade online dos serviços prestados, onde utilizam a *QoI* nas características da tomada à decisão em grupo e no processo de argumentação subjacente a este tipo de ambientes.

Lima et al. [LC+2008] realizam um estudo sobre a criação de um modelo computacional para a tomada de decisão em grupo baseada na avaliação da *QoI* ao longo dos vários estados de tomada à decisão, no contexto dos sistemas de suporte à decisão em grupo. *Lima et al.* [LN+2009] utiliza novamente a *QoI* na modelação do processo de tomada à decisão em grupo em especial para a avaliação das decisões em ambientes na área da saúde, onde as terríveis consequências das más decisões ou a falta delas poderá ter consequências desastrosas. Este estudo é estendido em [NS+2010] aplicando a *QoI* na tomada à decisão mas centrada na multiplicidade de cenários. No contexto desta tese de doutoramento, e nesta área da tomada à decisão em ambientes industriais, *Ribeiro et al.* [RN+2009] apresentam um estudo sobre a representação formal da informação da *QoI* na escolha dos melhores cenários durante

o processo de produção de vinhos, empregando distintas técnicas (*árvores de decisão, Redes Neuronais Artificiais e Regressão Linear*) para classificar vinhos segundo o seu processo de vinificação.

Na área da lei civil, *Novais et al.* [NA+2010] utilizam o conceito da *QoI* na avaliação das condições contratuais e de confiança no processo de negociação contratual. Na área da saúde e área médica, vários trabalhos foram apresentados utilizando o conceito da *QoI*. *Machado et al.* [MA+2008] [MM+2009] [MA+2010] utilizam a *Programação em Lógica Estendida* para a representação do conhecimento e dos mecanismos de raciocínio na modelação do processo de tomada à decisão na área médica. Utilizam a *QoI* em [MA+2010] para quantificar a qualidade do serviço em unidades de saúde. Em [MM+2009] apresentam um trabalho relacionado com a problemática dos dilemas médicos na tomada à decisão, através da modelação do conhecimento usando a programação em lógica estendida. Utilizam a *QoI* para quantificar os parâmetros associados à importância do estado emocional e a experiência do técnico de saúde.

Na área do *Ambient Assisted Living* e *Ambient Intelligence,* a *QoI* é utilizada nos trabalhos de *Lima et al.* [LC+2008] e *Gomes et al.* [GM+2010] para medir a qualidade do serviço de saúde prestada em unidades hospitalares nas suas várias vertentes de gestão.

4.3.2 Medida da Qualidade da Informação

O objectivo da medida da *Qualidade da Informação* (*QoI*) é definir um processo de quantificação da informação associada a uma teoria ou programa em lógica. A definição da *QoI* pode ser formalizada da seguinte forma:

Definição 8 – (*Qualidade-da-Informação – QoI*): Seja k ($k \in \{1,\ldots,m\}$) denotar os predicados cujas extensões tornam o programa em lógica estendida ou teoria que modelam o *universo do discurso* e j ($j \in \{1,\ldots,n\}$) os atributos desses predicados. Tal como apresentamos na secção 4.2.3, a cada *Programa em Lógica Estendida* está associado um conjunto de *abducibles*. Estes *abducibles* são representados aqui através de valores nulos. Neste sentido, a *QoI* com respeito a um predicado genérico k, pode ser analisado em quatro situações e pode ser medido com valores do intervalo $[0\ldots1]$: 1) quando a informação é conhecida (positiva ou negativa); 2) quando a informação é desconhecida; 3) quando a informação é desconhecida mas

pode ser seleccionada a partir de um ou mais valores e, 4) quando a informação é desconhecida, podendo ser derivada a partir de um conjunto de valores mas apenas um pode ser seleccionado. Se a informação for conhecida (positiva) ou falsa (negativa) (caso 1), a qualidade da informação para o predicado é "1" (16) que é o seu máximo valor. Para situações em que o valor é desconhecido (caso 2), a fórmula da qualidade da informação é dada por:

$$QI_k = \lim_{N\to\infty} \frac{1}{N} = 0(N >> 0) \tag{17}$$

Para a terceira situação (informação conhecida mas derivável a partir de um conjunto de um ou mais valores), a qualidade da informação do predicado é dada pela fórmula (18).

$$QI_k = 1/Card \tag{18}$$

onde *Card* denota a cardinalidade do conjunto de excepções (ou abducibles) do predicado k, se o conjunto de excepções não for disjunto. Se o conjunto de excepções for disjunto, ou seja, um valor pode ser seleccionado individualmente ou através da combinação dos restantes elementos do conjunto (quarta situação), a qualidade da informação é dada pela seguinte fórmula:

$$QI_k = \frac{1}{C_1^{card} + \ldots + C_{card}^{card}} \tag{19}$$

onde C_i^{Card} $i=1\ldots$ *card* é o subconjunto *card-combination*, com *card* elementos. Seja $x_j \in [min_j, max_j]$ o valor do atributo j, para cada predicado k está associada uma função de quantificação (*scoring*) $V^k{}_i : [min_i,\ max_i[\rightarrow\ 0\ldots 1]$, que dá a pontuação (*score*) ao predicado k atribuindo-lhe um valor de atributo i no intervalo [0,1]. Esta característica de aplicar pontuações, pode ser usada para dar importância aos predicados de um programa em lógica estendida. A importância relativa que um predicado possui para cada um dos seus atributos em observação; $w^k{}_j$ estabelece a relevância do atributo i para o predicado k. É também assumido que os pesos para todos os predicados são normalizados, i.e.,

$$\sum_{1\le i\le n} w_i^k = 1 \quad \forall i \tag{20}$$

É também possível definir funções de pontuação de predicados, i.e. definido por:

$$V^k(t) = \sum_{1\le i\le n} w^k{}_i \ QI_k(t) \tag{21}$$

onde k é um predicado, i os atributos do predicado k, e t é o tempo. Desta forma, é possível medir a *Qualidade da Informação* de um programa lógico, pondo os valores QI_k num espaço multidimensional, em que a área entre os eixos denotam o programa lógico ou teoria, com um número a partir de 0 (ao centro) a 1.

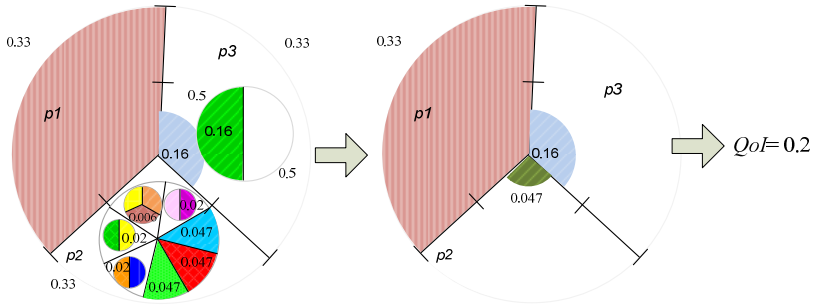


Figura 4.1 - Exemplo da representação gráfica da medida da Qualidade da Informação associada a uma teoria ou programa lógico.

Na figura 4.1 ilustra-se um exemplo da QoI associada a uma teoria ou programa lógico. Tendo em consideração a informação em termos das extensões dos predicados que modelam o *universo do discurso*, a representação e o valor total da QoI pode variar. Na figura está ilustrado um exemplo para três predicados (p_1, p_2 *e* p_3), em que se pretende determinar em simultâneo todos os possíveis valores para $p_1(X)$, $p_2(X)$ e $p_3(X)$, onde a informação do predicado p_1 é conhecida, a do predicado p_2 é do tipo 3 (apresentado na secção 4.2.3) e a do predicado p_3 é do tipo 2. Em virtude da questão colocada ao sistema contemplar os três predicados, o gráfico é dividido em três partes correspondendo a "0.33"do valor total da informação (que seria 1 se a informação para os três predicados fosse conhecida). No entanto, o semicírculo para o predicado p_3 representa a situação em que existem duas extensões ao predicado com um invariante que restringe a selecção de um dos possíveis valores. Neste caso, como existem duas possibilidades este semi-circulo é dividido em dois, correspondendo a "0.5"do valor total desse semi-circulo. Contudo, o valor do semi-circulo corresponde a "0.33"sendo o valor da QoI para cada uma das possíveis soluções do predicado p_3 de "0.16". Em relação ao predicado p_2, representa a situação em que existem três possíveis soluções para obter o seu valor. Neste caso, a combinação de todos os

possíveis valores corresponde a sete possibilidades (conforme a fórmula (19)). Neste sentido, o semicírculo do predicado p_2 é dividido em sete partes correspondendo a cada uma das possíveis soluções. No entanto, por exemplo numa dessas soluções, o valor é "0.02"correspondeo a um cenário com duas possibilidades de selecção de um valor para o predicado. No sentido de clarificar melhor este processo, no *anexo A* apresentamos um exemplo mais ilustrativo do cálculo da *QoI*.

A área delimitada pelos arcos corresponde apresenta a medida da *QoI* obtida para cada solução do problema em consideração (figura 4.1), usando QI_k e as extensões dos predicados p_1, p_2 e p_3, o mapeamento num espaço multidimensional e a área delimitada pelos arcos apresenta a *qualidade da informação* do *Programa em Lógica Estendida*. Com este mecanismo de quantificação é possível atribuir um valor de verdade ao conhecimento de uma entidade representada através da *Programação em Lógica Estendida*. Neste sentido, não se trata de atribuir uma probabilidade à informação desconhecida, mas sim, um valor de verdade desse conhecimento.

4.4 Inteligência Evolucionária

4.4.1 Esquemático Arquitectural

A arquitectura do modelo computacional é designada por ***intelecto virtual*** *(ou arquétipo)*, sendo estruturado por um conjunto de entidades denominadas por neurónios (figura 4.2). Nesta abordagem, existem dois arquétipos: um correspondente à estrutura processual das entidades envolvidas na interpretação do problema e o segundo intelecto correspondente ao processo de instanciação das soluções durante o processo de inferência para a interpretar ou resolver uma questão (ou lista de problemas a interpretar). A cada neurónio está associado o contexto que modela o *universo do discurso* representado pelas extensões do programa em lógica. O contexto corresponde à representação do conhecimento através dos predicados, das suas extensões, dos invariantes e das dependências estruturais e de relacionamento entre predicados, em que esta representação é efectuada recorrendo à *Programação em Lógica Estendida.*

O *input* do sistema corresponde à descrição inicial do *universo do discurso* juntamente com uma questão ou lista de problemas a interpretar. Por outro lado, na entrada de cada neurónio é apresentada uma lista de problemas correspondendo ao

problema a interpretar, sendo o contexto dado em termos da realidade associada a um dado sub-problema. Por exemplo para resolver o problema inicial "$?\ (p_1(X),\ p_2(Y),\ p_3(Z))$", a lista de problemas submetida a um neurónio de uma camada intermédia, poderá ser interpretar o valor de p_1 *e* p_3 (ou seja, provar o teorema "$?\ \ (p_1(X),\ p_3(Z))$"), noutro "$?\ \ (p_2(Y),\ p_1(X))$", e assim por diante ao longo do processo de inferência até interpretar (ou resolver) o problema inicial.

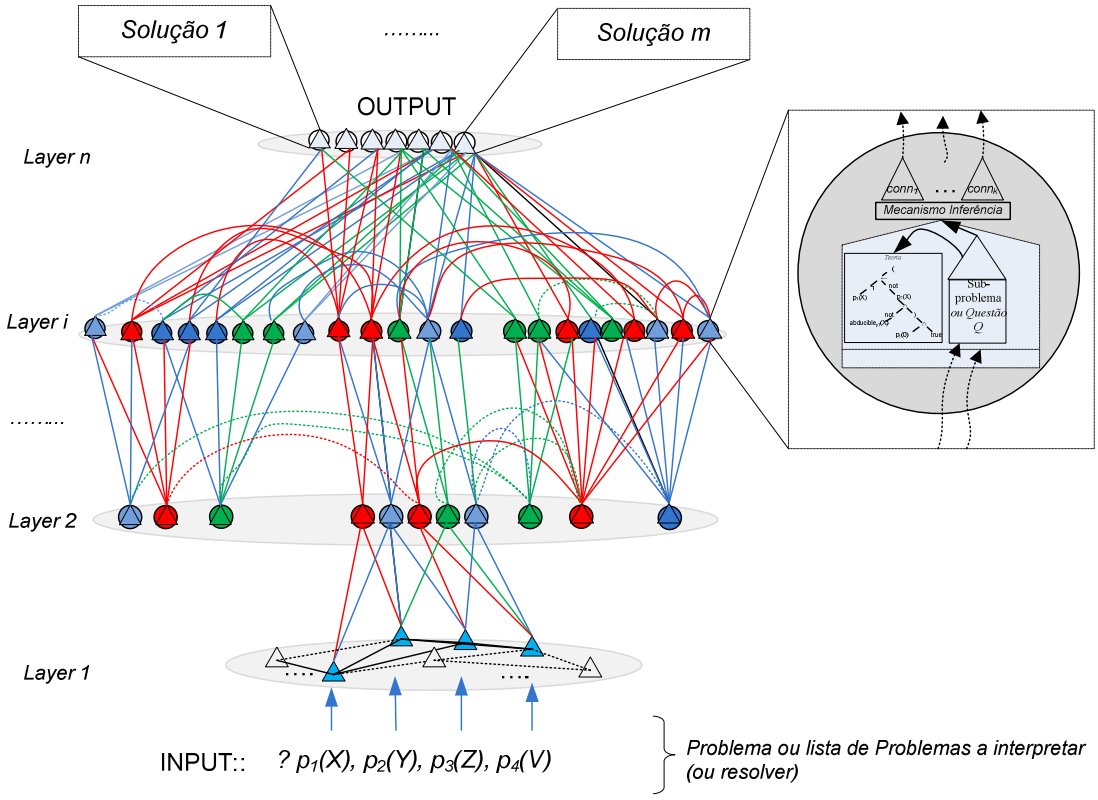


Figura 4.2 - Esquemático arquitectural do sistema.

No entanto, nenhuma quantidade de optimização evolucionária poderá criar uma rede de neurónios perfeitamente ajustada para todos os tipos de tarefas intelectuais. Neste sentido, os neurónios artificiais terão a capacidade de modificar os seus próprios circuitos sempre que necessário, de modo a escolher o melhor caminho para resolver uma questão (ou problema). Assim, o processo evolucionário inicia-se com uma representação inicial do *universo do discurso* e progride procurando a optimização

baseando-se na medida da *Qualidade da Informação* [NA+2006] [NM+2007] associada a uma teoria ou programa lógico até não haver melhoramentos, sendo condicionada através de uma condição/*threshold* previamente definida. Esta condição pode ser clarificada do seguinte modo: para a interpretação de uma questão inicial ou resolução de uma lista de problemas (ex. "*? $(p_1(X), p_2(Y), p_3(Z))$*"), será criado um intelecto virtual arquitectural, assim como criados os neurónios que o compõem interligados entre si, correspondendo a todos os possíveis relacionamentos entre as entidades envolvidas na interpretação da questão inicial.

Aquando da instanciação das variáveis, esse intelecto estrutural ser instanciado no contexto associado a cada um dos seus neurónios. Consoante o tipo de extensões existentes em cada contexto de cada neurónio, será avaliada a *QoI* associada ao programa lógico que interpreta ou resolve a questão (ou lista de problemas) submetidas a esse neurónio. Se o valor da *QoI* estiver abaixo de um *threshold*, o processo de inferência pára, não evoluindo no intelecto na tentativa da resolução do problema, pois a qualidade que transporta não oferece um valor suficiente para o grau de confiança esperado.

Em suma, o *input* do sistema corresponde a uma especificação de uma questão a interpretar ou (a resolver), podendo ser também designada por lista de problemas a resolver. Por exemplo, interpretar todos os valores para os predicados p_1, p_2 *e* p_3, ou resolver a lista de problemas para os predicados "p_1, p_2 *e* p_3".

Tendo em consideração os relacionamentos entre as entidades envolvidas na interpretação de uma questão (neste caso predicados e as suas extensões), a instanciação do intelecto virtual arquitectural irá autoorganizar-se de uma forma paralela e massiva, atendendo à reacção de cada neurónio, sendo criadas novas ligações, novos neurónios e assim por diante. Estas ligações entre tipos de neurónios variam consoante o problema em causa, sendo que, neurónios do mesmo tipo têm uma ligação "formal" em comum[2], por exemplo, num modelo relacional de base de dados, uma ligação formal corresponde às ligações de relacionamento entre entidades.

Na figura 4.9 da secção 4.4.5 iremos apresentar um exemplo com quatro entidades, em que existe por exemplo uma ligação formal entre a entidade "suppliers"e "companies". Desta forma durante o processo de instanciação cada nova ligação irá seleccionar um intervalo aleatório dentro do intervalo de ligações disponíveis a partir

[2] sendo estas relações baseadas nas dependências de relacionamentos dos predicados representados nos programas lógicos (ou predicados lógicos envolvidos na resolução da questão).

de um neurónio. Este intervalo de ligações, pode ser clarificado da seguinte forma: se considerarmos um cenário de aplicação (ex. numa base de dados relacional) com inúmeras entidades e relacionamentos entre elas, há medida que as questões são colocadas ao sistema, são criados intelectos virtuais (ou arquétipos) com a estrutura do mecanismo de inferência envolvendo as entidades necessárias para a resolução das questões. Para cada uma das questões o seu intelecto estrutural é instanciado, sendo a questão inicial sub-dividida em várias que irão ser submetidas aos vários neurónios do intelecto. Durante o processo de instanciação poderão haver vários caminhos entre o "*input*"do intelecto até à saída no sentido de obter as soluções. Por este facto, tendo em consideração a avaliação da *QoI* em cada neurónio, o caminho da instanciação poderá seguir "rotas"diferentes até se obter por um lado uma ou várias soluções que interpretam a questão inicial, permitindo criar na instanciação do intelecto arquitectural um dinamismo de evolução durante o processo de pesquisa de soluções para a interpretação do problema inicial.

Neste sentido, aquando de uma nova questão a interpretar, várias entidades poderão estar envolvidas, as quais já foram utilizadas na interpretação de questões anteriores. Por este facto, vários caminhos entre essas entidades poderão ser utilizados e, por este facto, a passagem para o neurónio seguinte a partir de um neurónio poderá ter várias ligações. Neste contexto, a seleção da ligação a seguir poderá ser efectuada de forma aleatória entre as ligações possíveis a partir de um neurónio. Na aplicação do paradigma evolutivo no sentido de encontrar possíveis caminhos no processo de inferência, se o processo de inferência encontrar uma possível ligação para um neurónio do mesmo tipo, então irá conectar-se a esse neurónio sendo criado um novo neurónio que fará a união dos seus antecedentes. Se encontrar um neurónio de tipo diferente então, irá seleccionar um outro ponto na aplicação dos operadores genéticos e tentar de novo o processo de pesquisa de novos caminhos até ao nodo de saída. Caso não encontre um qualquer neurónio, irá criar um novo neurónio do mesmo tipo e ligar-se a ele, em que esta situação corresponderá à criação do neurónio da camada de saída e à ligação a ele dos neurónios da camada anterior.

Na instanciação do esquemático arquitectural (figura 4.2), a qualquer momento um novo neurónio é criado, sendo geradas novas conexões, em que algumas dessas conexões poderão conectar-se a neurónios já existentes. Esta reacção dos neurónios e da criação das ligações irá continuar até que se atinja um limite dado por um número máximo de iterações, ou desde que não haja melhoramentos na *Qualidade*

da Informação associada a cada neurónio (*threshold* mínimo). Quando o intelecto virtual estiver criado, o último grupo de neurónios é designado por neurónios de saída, correspondendo às melhores teorias lógico matemáticas encontradas para a resolução de um problema inicial. Caso na camada de saída existam vários neurónios e consequentemente várias teorias para a resolução do problema, será aplicada a relação de ordem da medida da *Qualidade da Informação* associada a essas teorias.

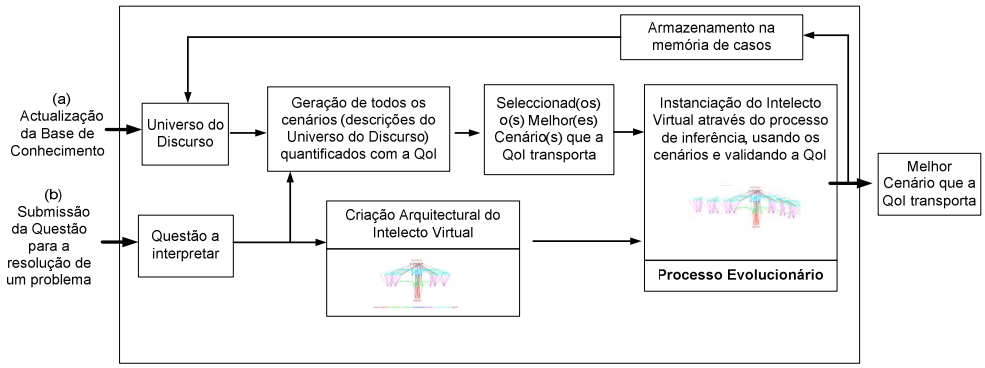


Figura 4.3 – Arquitectura processual da criação do Intelecto Virtual.

Na figura 4.3 ilustra-se o esquemático processual da criação do *intelecto virtual*. O processo inicia-se com a informação associada ao *universo do discurso*, podendo ser actualizada ao longo do tempo (figura 4.3 (a)). Nestes ambientes e, no seguimento do exposto ao longo deste documento, o *universo do discurso* é composto pelas extensões dos predicados em termos de informação positiva, negativa e desconhecida, assim como dependências estruturais (a invocar no upgrade das extensões dos predicados) e dependências de relacionamento (a invocar aquando da remoção da informação nas extensões dos predicados e na invocação de interrogações que envolvam as extensões dos predicados). Tendo em consideração esta informação associada ao *universo do discurso*, são gerados e quantificados todos os cenários possíveis com base nas extensões dos predicados que modelam o *universo do discurso*. Face a esta quantificação são seleccionados os melhores cenários que serão utilizados para a interpretação e resolução (ver secção 2.3.2) da questão colocada ao sistema. Aquando da colocação de uma questão é criado o primeiro intelecto virtual que corresponde a uma representação em grafo da estrutura das entidades envolvidas na resolução da questão e que irão alimentar o processo computacional para a resolução do problema.

Depois de criado o intelecto arquitectural, este corresponde a uma abstracção das entidades (ou predicados) envolvidas que modelam o *universo do discurso* e que serão utilizadas na interpretação da questão. De seguida é criado um segundo intelecto que corresponde à invocação do processo de inferência através da sua instanciação usando os cenários e validando a *QoI* ao longo do processo evolutivo. Em cada uma das camadas do arquétipo, os neurónios irão executar o processo de inferência em relação ao sub-problema que lhe é colocado, sendo testada a qualidade da informação com base num limite mínimo (*threshold*) previamente definido.

Em termos práticos, no final da criação do modelo (intelecto virtual instanciado) iremos obter um valor da *QoI* (e uma teoria ou programa lógico) que corresponde à melhor quantificação do *universo do discurso* optimizando-a ao longo do tempo, em que esta optimização só será possível com a colocação de novas questões ao sistema, sendo criados novos intelectos em que, num determinado momento, a resposta à interpretação de uma questão submetida anteriormente poderá ser respondida com outros cenários (programas lógicos) a partir da estrutura evolutiva do intelecto que foi actualizado com os "sub-intelectos"para as questões que foram sendo colocadas ao sistema. Sabendo esse valor iremos obter a melhor teoria ou programa lógico) e consequentemente a melhor modelação do problema a resolver.

4.4.2 Genoma

Conforme referimos, os *Algoritmos Evolucionários* baseiam-se nos conceitos de *cromossomas*, *genoma e phenotype*. Nestes sistemas durante o curso de evolução o *phenotype* corresponde à base para a avaliação de um indivíduo, tendo em consideração o seu ambiente contextual atendendo à sua integridade e funcionalidade, i.e, neste trabalho a integridade e funcionalidade refere-se à estrutura e relacionamentos entre predicados que modelam o *universo do discurso*. O *genotype* corresponde à representação real de um indivíduo sendo por exemplo, na *Programação Genética* armazenada num grafo, podendo o *phenotype* ser deduzido e avaliado a partir daí. Neste sentido, determinar a estrutura do *genotype* corresponde ao primeiro desafio no desenho dos sistemas evolucionários. Contudo, deve-se ter particular atenção que os blocos que constituem o ambiente genético devem ser versáteis o suficiente para construir um número de elementos de processamento e iterações dentro dos neurónios artificiais, tornando possível a criação de um sistema inteligente dinâmico. Por este

facto, os *genomas* irão ser constituidos por um conjunto de genes que codificam diferentes tipos de neurónios, codificando também diferentes tipos de iterações entre eles. Neste sentido, o esquemático do *genoma* é apresentado na figura 4.4.

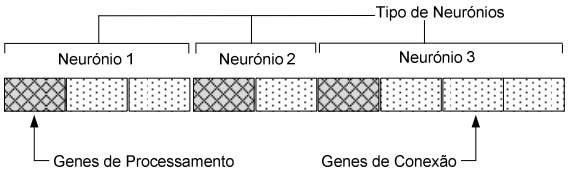


Figura 4.4 - Exemplo de um genoma com três tipos de neurónios, cada um com um gene de processamento simples, dois, um ou três genes de conexão.

Neste sistema, cada neurónio é codificado por dois tipos de genes: um gene de processamento que especifica como cada neurónio irá calcular a sua saída e um conjunto de genes de conexão. Este gene de conexão especifica as potenciais ligações entre outros predicados que modelam todo o *universo do discurso*. Por este facto, cada neurónio poderá conter vários genes de conexão, consoante as dependências de relacionamento entre os predicados envolvidos na questão submetida ao neurónio. A arquitectura de toda a rede e o *layout* físico em relação a um ou a outro neurónio irá emergir a partir da informação contida nos genes de conexão.

Por sua vez, os genes de processamento determinam como cada neurónio converte o seu *input* num *output*, isto é, recebe uma questão a interpretar e, mediante o contexto que lhe está associado, procede ao processo de inferência e ao cálculo da *QoI* na avaliação da(s) solução(ões) para a questão que recebeu como *input*. O *output* corresponderá aos cenários ou soluções obtidas juntamente com a sua *QoI*. Desta forma, a entrada para um neurónio corresponde ao *output* de todos os neurónios que são conectados a ele, ponderada pelos valores da *QoI* transportada pelas ligações para esse neurónio.

Neste contexto, a representação interna de um *Algoritmo Evolucionário* permite especificar os indivíduos de uma população. Neste sentido, seja $P(t)$ a representação de um conjunto de indivíduos de uma população num tempo t, a qual pode ser descrita da seguinte forma:

$$P(t) = <\{gp_1(t),[gc_1(t),...,gc_k(t)]\},...\{gp_n(t),[gc_1(t),...,gc_k(t)]\}> \quad (23)$$

em que, o indivíduo é composto por um conjunto de um gene de processamento ($gp_n(t)$) e um ou mais genes de conexão ($gc_k(t)$). Sobre cada um deles iremos nas secções seguintes apresentar com detalhe a sua estrutura e funcionamento.

4.4.3 Genes de Processamento

O gene de processamento é constituído por um conjunto de entradas com o valor da *Qualidade da Informação* do neurónio que o precede juntamente com a informação do conhecimento desse neurónio, a qual corresponde ao contexto (programa lógico) associado ao *universo do discurso* do neurónio, i.e. predicados, extensões dos predicados, invariantes, dependências estruturais e de relacionamento. O nível interno de

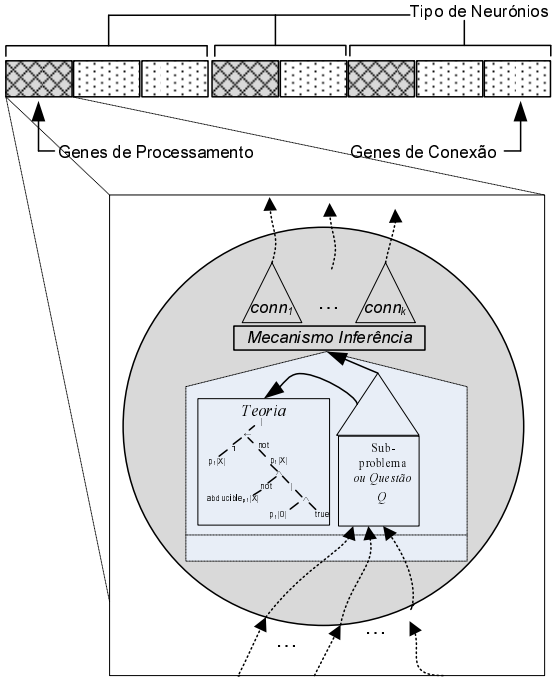


Figura 4.5 - Esquemático dos genes de processamento.

cada gene de processamento é também constituído pelo mecanismo de inferência, que tem como objectivo interpretar uma questão (sub-problema ou lista de problemas) a

qual é dada como entrada no neurónio.

O gene de processamento usa uma linguagem de representação de alto nível baseada nas cláusulas de *Horn* [CH1985]. Segundo os fundamentos apresentados na secção 4.2.3 referente à representação da informação e mecanismo de raciocínio, o gene de processamento representado por $gp_n(t)$ é tido em conta a partir de uma teoria ordenada $OT=(T,<,(S, \prec))$, onde T, $>$, S e $\prec$ correspondem respectivamente à base de conhecimento do gene na forma clausal, uma relação de ordem não circular sobre essas cláusulas, um conjunto de regras de prioridade e uma relação de ordem não circular sobre essas regras.

Neste sentido, o gene de processamento poderá ser descrito da seguinte forma:

$$gp_n(t) = < T, M,Q,C,QoI >$$

em que T corresponde à *Teoria (ou Universo do Discurso)*, M ao mecanismo de raciocínio (ou intelecto virtual), Q a questão (ou sub-problema) a resolver (ou a interpretar), C o(s) cenário(s) passíveis de resolver ou a interpretar o problema, a QoI que maximiza o melhor cenário. A teoria T é expressa através da *Programação em Lógica Estendida* (definição 6) em que o conhecimento de cada gene é expressa usando a extensão às cláusulas de conhecimento descritas na definição 5. O processo de inferência (ou mecanismo de raciocínio) segue a definição de prova através do meta-teorema de resolução de problemas apresentado na definição 7.

4.4.4 Genes de Conexão

O gene de conexão determina como cada gene pode ser conectado com outro, sendo quantificada pela QoI que a ligação transporta. Isto é, aquando de uma questão submetida ao sistema é criado um arquétipo geral com todos os relacionamentos entre os predicados envolvidos na interpretação da questão, sendo criados na estrutura do arquétipo, entidades (ou neurónios) associados aos relacionamentos entre alguns predicados que interpretam parte da questão inicial. Considerando um ambiente em que existem vários relacionamentos entre predicados, um neurónio poderá estar ligado a um ou a vários outros neurónios, correspondendo ao percurso do processo de inferência. Aquando do processo instanciação dos termos dos predicados para a interpretação de uma questão submetida ao sistema, vários caminhos podem ser tomados

até se obter a(s) solução(ões)s que interpretam a questão. Por este facto, os genes de conexão são estruturados de modo a assegurar a possibilidade de se efectuar a ligação a outros neurónios. Neste contexto, estes genes são esquematizados da seguinte forma:

$$gc_n(t)=\{[connect(p,q),estado,QoI],contagem,\ direcção,\ alcance\}$$

em que *connect(p,q)* refere-se á indicação da ligação entre o predicado p e o predicado q, correspondendo a uma dependência de relacionamento entre predicados, *estado* corresponde ao estado da ligação (i.e. se foi activada ou não na geração do actual indivíduo) e, a *QoI* corresponde à qualidade da informação determinada no neurónio após o processo de inferência na interpretação da questão apresentada na entrada desse neurónio. A este primeiro componente do gene de conexão dámos o nome de "força do gene", uma vez que apresenta perante outros neurónios a sua "força"em termos da *QoI* que transporta na sua participação na interpretação da questão submetida ao sistema. A partir do momento em que há a necessidade de especificar superficialmente mais informação que os genes de processamento, os genes de conexão são mais complexos. Por este facto, os três últimos argumentos do gene de conexão tem como objectivo direccionar o caminho para o próximo neurónio do arquétipo. São codificados em quatro genes: contagem, direcção e alcance. Tendo em consideração a especificação apresentada, na figura 4.6 ilustra-se um esquemático de um gene de conexão.

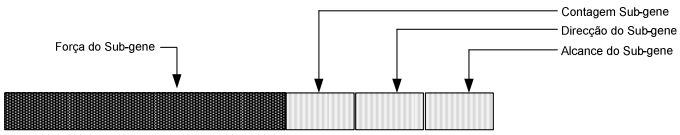


Figura 4.6 - Esquemático de um gene de conexão.

Estes três últimos parâmetros do gene de conexão determinam como as várias ligações entre neurónios são realizadas, onde as ligações vão, a sua direcção, que neurónios vão alcançar e que tipo de neurónios irá conectar. A genética de cada subgene consiste simplesmente num valor inteiro. O gene de contagem especifica o número de ligações realizado por cada neurónio. O número total de ligações realizadas

por cada neurónio corresponderá à soma da contagem do subgene "contagem" de cada um dos genes de conexão. O subgene de direcção determina a orientação de cada uma das ligações, especificando a direcção da ligações, para trás (neurónio de uma camada inferior), para o lado (neurónio de uma mesma camada), para a frente (neurónio de uma camada superior), ou de forma aleatória, podendo ser parametrizados respectivamente pelos valores {1,2,3,4}, em que o valor "1" corresponde à projecção para trás, o valor "2" para o lado, e assim por diante. O gene "alcance" determina quão longe dentro de uma camada simples a ligação poderá alcançar outros neurónios. Por exemplo, o valor "1" neste gene indica que deverá ser seleccionado o neurónio da camada imediatamente superior (neste caso "para a frente" se o gene direcção especificar o valor "3") ou por exemplo "2" indicando que deverá ser seleccionado o neurónio de duas camadas imediatamente acima (neste caso "para a frente" se o gene direcção especificar o valor "3").

Operadores Genéticos

A aplicação do operador de cruzamento (ilustrado na figura 4.7) separa em dois um par de genomas de forma aleatória, resultando dois novos genomas. O operador de mutação é aplicado no sentido de inserir ou remover um ou mais genes adjacentes de um genoma e unir noutro. No entanto, nenhuma das acções do operador de mutação altera por si próprio qualquer gene, ou seja, os genes são sempre mantidos intactos.

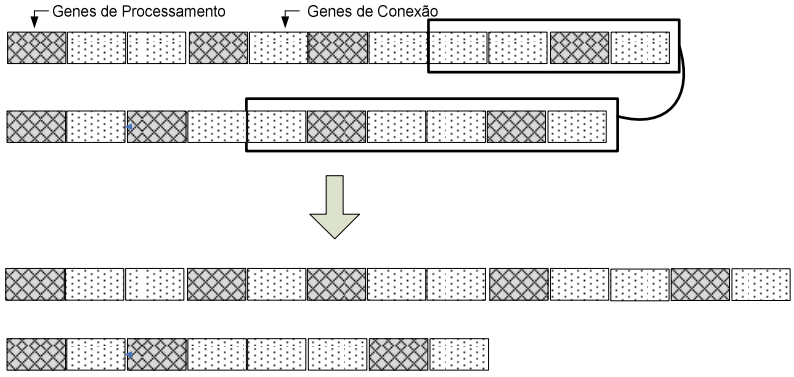


Figura 4.7 - Esquemático da aplicação do operador de cruzamento.

No sentido de assegurar que a mutação não "perturbe" a semântica da criação do intelecto virtual, ela nunca pode deixar um gene de processamento sem pelo menos conter um gene de conexão.

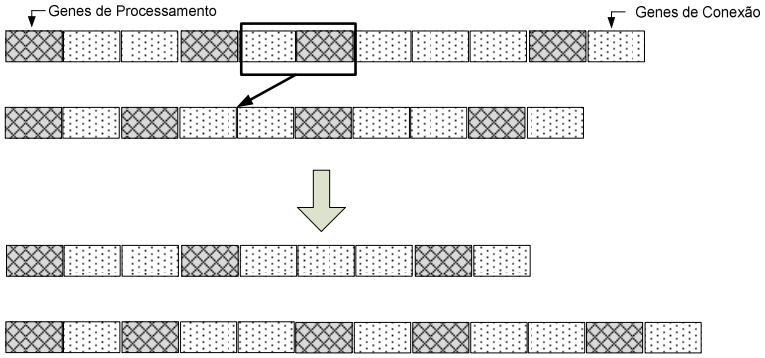


Figura 4.8 - Esquemático da aplicação do operador de mutação.

A mutação e cruzamento (ilustrado na figura 4.8) apresentam a componente estocástica do processo evolucionário. No entanto a orientação determinística da evolução corresponde à selecção da teoria da evolução, a qual permite selecção os melhores indivíduos com base numa função de avaliação.

Tradicionalmente, na aplicabilidade dos *Algoritmos Evolucionários*, o processo de selecção tem como objectivo seleccionar individuos aplicando-se um valor de aptidão, de modo a seleccionar para uma fase seguinte os individuos mais promissores. Entre outros, os métodos de conversão de valores de aptidão e técnicas de amostragem (deterministica e estocástica), são amplamente utilizadas para escolher os individuos com melhor aptidão. Por outro lado, o processo orientado para a avaliação dos tipos de genes (representados na Programação Genética na forma de grafo, são transformados em *phenotypes* e avaliados pelo interpretador (neste caso o *PROLOG*). O critério para a avaliação de um indivíduo é a medida da *Qualidade da Informação* apresentada na secção 4.3.2.

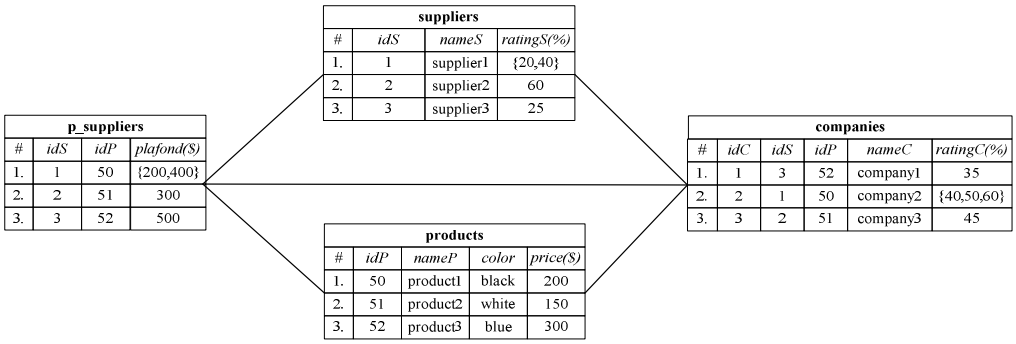
4.4.5 Processo de Inferência

Em cada neurónio ou entidade do sistema (figura 4.2) é executado o processo de inferência, no sentido de proceder à interpretação e (tentativa de) resolução do prob-

lema. Em cada entidade, o contexto (expresso em forma de programas lógicos) é representado segundo a *Programação em Lógica Estendida*.

A representação de informação negativa explícita [PW1990] contornando o *Pressuposto do Mundo Fechado*, permite distinguir entre o que se sabe, o que não se sabe e o que se desconhece. No entanto, existem outras situações interessantes a representar na base de conhecimento de forma a melhor aproximar essa base de conhecimento da realidade, nomeadamente no que toca à informação incompleta.

No sentido de apresentar o processo de inferência, considere-se o seguinte caso de estudo onde se contempla a utilização dos valores nulos usados para representar situações desconhecidas. Considere-se um exemplo típico de um sistema de gestão de bases de dados [MUW2002] [RM$^+$2010a] em que o modelo relacional é composto por quatro relações como se ilustra na figura 4.9.



suppliers

#	*idS*	*nameS*	*ratingS(%)*
1.	1	supplier1	{20,40}
2.	2	supplier2	60
3.	3	supplier3	25

p_suppliers

#	*idS*	*idP*	*plafond($)*
1.	1	50	{200,400}
2.	2	51	300
3.	3	52	500

companies

#	*idC*	*idS*	*idP*	*nameC*	*ratingC(%)*
1.	1	3	52	company1	35
2.	2	1	50	company2	{40,50,60}
3.	3	2	51	company3	45

products

#	*idP*	*nameP*	*color*	*price($)*
1.	50	product1	black	200
2.	51	product2	white	150
3.	52	product3	blue	300

Figura 4.9 - Exemplificação de um simples modelo de base de dados relacional.

O objectivo deste exemplo é o de representar um modelo simples para armazenar e gerir informação de fornecedores ("suppliers") e empresas ("companies") que fornecem produtos a clientes. Tipicamente a estrutura de representação nestes ambientes é designada por "relação" que corresponde a uma forma bidimensional de representar os atributos da relação, assim como os seus registos.

Neste contexto, considere-se o seguinte esquema do modelo de dados através do nome da relação e um conjunto de atributos:

1. *suppliers(#idS, nameS,ratingS)*
2. *p_suppliers(#idS,#idP, plafond)*
3. *companies(#idC,#idS,#idP,nameC,ratingC)*
4. *products(#idP,nameP,Color,Price)*

onde na primeira linha a relação "suppliers" está estruturada por três atributos, correspondendo à identificação do fornecedor (*idS*), o nome do fornecedor (*nameS*) e a cotação do fornecedor no mercado (*ratingS*). A segunda linha ("p_suppliers") apresenta o *plafond* do fornecedor com os atributos, identificação do fornecedor (*idS*), identificação do produto (*idP*) e o *plafond* fornecido pela empresa. Na terceira linha representa-se a relação das empresas com os atributos, identificação da empresa (*idC*), identificação do fornecedor (*idS*), a identificação do produto (*idP*), o nome da empresa (*nameC*) e a cotação da empresa no mercado (*ratingC*). Na última linha é apresentada a relação "produtos" com os seguintes atributos, identificação do produto (*idP*), nome do produto (*nameP*), cor do produto (*Color*) e o preço do produto (*Price*). Neste modelo relacional existem algumas relações que asseguram as dependências estruturais das suas relações.

Temos no entanto de salientar, que neste exemplo não estamos preocupados em assegurar as características associadas à normalização [MUW2002] da base de dados, mas antes em apresentar um exemplo ilustrativo com várias relações interligadas, no sentido de exemplificar o âmbito de aplicabilidade do nosso estudo. Contudo, neste exemplo a informação associada é contemplada com a presença de informação incompleta:

(a) Para o fornecedor com a identificação "1" a cotação (ou *ratingS*) é desconhecida mas pode ser seleccionada por um dos seguintes valores "*20*" ou "40".

(b) O *plafond* do fornecedor (ex. em *dollars*) com a identificação "1" é desconhecida mas pode ser seleccionada apenas a partir dos valores "*200*" ou "*400*".

(c) A cotação da empresa com o identificador "2" para o fornecedor com a identificação "1" e produto "50" é desconhecida, mas pode ser seleccionada a partir do

conjunto de valores "40", "50" ou "60".

Neste contexto, seguindo o modelo relacional ilustrado na figura 4.9 e as características da informação incompleta apresentada, podemos usar o formalismo da *Representação do Conhecimento* através da *Programação em Lógica Estendida* (definição 3):

Programa 1 – Representação do Conhecimento em termos das extensões do predicado suppliers {

$$
\begin{aligned}
&(\neg\ suppliers(X,Y,Z) \leftarrow\ not\ suppliers(X,Y,Z) \wedge \\
&\qquad\qquad not\ abducible_{suppliers}(X,Y,Z)) \wedge \\
&\quad abducible_{suppliers}(1,\ supplier1,20) \wedge \\
&\quad abducible_{suppliers}(1,\ supplier1,40) \wedge \\
&\quad ?((abducible_{suppliers}(X_1,Y_1,Z_1) \vee abducible_{suppliers}(X_2,Y_2,Z_2)) \\
&\wedge \neg(abducible_{suppliers}(X_1,Y_1,Z_1) \wedge abducible_{suppliers}(X_2,Y_2,Z_2))) \wedge \\
&\quad suppliers(2,supplier2,60) \wedge \\
&\quad suppliers(3,supplier3,25)
\end{aligned}
$$

}

No programa 1 o símbolo "¬" representa a negação forte, denotando que a informação deverá ser interpretada como falsa, e o termo "*not*" designa a negação por falha na prova (*negation-by-failure*). A primeira cláusula representa o *Pressuposto do Mundo Fechado* do predicado "suppliers".

A segunda e terceira cláusulas representam o facto de que o valor da cotação para o predicado "suppliers" é desconhecido mas pode ser obtido a partir do conjunto de valores "20" ou "40".

A quarta cláusula apresenta o invariante que implementa o operador *XOR*, i.e., estabelece que o predicado "suppliers" ou é um ou outro e não os dois em simultâneo.

Programa 2 – Representação do Conhecimento em termos das extensões do predicado p_suppliers {

$$
\begin{aligned}
&(\neg p\_suppliers(X,Y,Z) \leftarrow \ not\ p\_suppliers(X,Y,Z) \wedge \\
&\qquad\qquad not\ p\_suppliers(X,Y,Z)) \wedge \\
&abducible_{p\_suppliers}(1,50,200) \wedge \\
&abducible_{p\_suppliers}(1,50,400) \wedge \\
&?((abducible_{p\_supplier}(X_1,Y_1,Z_1) \vee abducible_{p\_suppliers}(X_2,Y_2,Z_2)) \\
&\wedge \neg(abducible_{p\_suppliers}(X_1,Y_1,Z_1) \wedge abducible_{p\_suppliers}(X_2,Y_2,Z_2))) \wedge \\
&p\_suppliers(2,51,300) \wedge \\
&p\_suppliers(3,52,500)
\end{aligned}
$$

}

Programa 3 – Representação do Conhecimento em termos das extensões do predicado companies {

$$
\begin{aligned}
&(\neg companies(X,Y,Z,W,H) \leftarrow \ not\ companies(X,Y,Z,W,H) \wedge \\
&\qquad\qquad not\ companies(X,Y,Z,W,H)) \wedge \\
&abducible_{companies}(2,1,50,company2,40) \wedge \\
&abducible_{companies}(2,1,50,company2,50) \wedge \\
&abducible_{companies}(2,1,50,company2,60) \wedge \\
&companies(1,3,50,company1,35) \wedge \\
&companies(3,2,51,company3,45)
\end{aligned}
$$

}

No programa 3 a segunda, terceira e quarta cláusula apresenta o caso em que o valor do atributo cotação do predicado "companies" é desconhecido mas pode ser obtido a partir do conjunto de valores "40", "50" ou "60".

Programa 4 –Representação do Conhecimento em termos das extensões do predicado products{

$$(\neg\ products(X,Y,Z,W) \leftarrow\ not\ products(X,Y,Z,W) \wedge$$
$$not\ abducible_{products}(X,Y,Z,W)) \wedge$$

$$products(50,product1,black,200) \wedge$$

$$products(51,product2,white,150) \wedge$$

$$products(52,product3,blue,300)$$

}

Seguindo o descrito na figura 4.3 sobre o circuito processual para a resolução de um problema ou interpretação de uma questão, será criada a arquitectura do intelecto virtual. Por outro lado, a partir do *universo do discurso* são gerados todos os cenários possíveis de serem utilizados na interpretação da questão.

Com base nesses cenários, poderemos utilizar a medida da Qualidade da Informação associada a um programa em lógica estendida, no sentido de quantificar esse cenário.

Neste sentido, consideremos que pretemos interpretar a questão: *"Quais os fornecedores que no presente estão em condições de fornecer produtos de cor preta?"*. As entidades envolvidas na interpretação desta questão são: "suppliers", "p_suppliers"e "products".

Tendo em consideração a informação do *universo do discurso*, em termos de informação conhecida e desconhecida com os invariantes associados (neste caso dois invariantes do tipo "XOR"associados às entidades "suppliers"e "p_suppliers"), no anexo A apresentamos os quatro cenários possíveis obtidos através do processo de inferência.

Dois desses cenários correspondem às seguintes extensões ao programa lógico, teoria ou cenário 1: {

$$(\neg suppliers(X,Y,Z) \leftarrow\ not\ suppliers(X,Y,Z)\ \wedge$$

$$not\ abducible_{suppliers}(X,Y,Z)) \wedge$$

$$abducible_{suppliers}(1,\ supplier1,20) \wedge$$

$$(\neg\ p\_suppliers(X,Y,Z) \leftarrow\ not\ p\_suppliers\ (X,Y,Z)) \wedge$$

$$not\ abducible_{p\_suppliers}(X,Y,Z) \wedge$$

$$abducible_{p\_suppliers}(1,50,200) \wedge$$

$$(\neg\ products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$$

$$not\ abducible_{products}(X,Y,Z,W)) \wedge$$

$$not\ products(50,product1,black,200)$$

}

A extensão ao programa lógico, teoria ou cenário 2 pode ser representado na seguinte forma: {

$$(\neg suppliers(X,Y,Z) \leftarrow\ not\ suppliers(X,Y,Z) \wedge$$

$$not\ abducible_{suppliers}(X,Y,Z)) \wedge$$

$$abducible_{suppliers}(1,\ supplier1,20) \wedge$$

$$(\neg\ p\_suppliers(X,Y,Z) \leftarrow\ not\ p\_suppliers\ (X,Y,Z) \wedge$$

$$not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$$

$$abducible_{p\_suppliers}(1,50,400) \wedge$$

$$(\neg\ products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$$

$$not\ abducible_{products}(X,Y,Z,W)) \wedge$$

$$not\ products(50,product1,black,200)$$

}

Através da geração destes cenários a partir da informação incompleta contida

no *universo do discurso* que interpreta a questão colocada e, considerando o grau de relevância (ou importância $w^k{}_j$ apresentada na secção 4.3.2) dos predicados da seguinte forma: 0.25, 0.35 e 0.40 para os predicados "suppliers", "p_suppliers"e "products", é possvel (face o exposto na secção 4.3.2) quantificar e representar a quantificação do conhecimento associado a cada um dos cenários (Anexo A) em que $QoI(Q_{N,M})$, $N \in \{1,2,3\}$ correspondente ao número das questões e $M \in \{1,..R\}$, onde R corresponde ao número máximo de cenários possíveis de interpretar a questão em observação. A quantificação de cada um dos cenários é apresentada no anexo A, sendo o gráfico da qualidade da informação de todos os cenários ilustrado na figura 4.10.

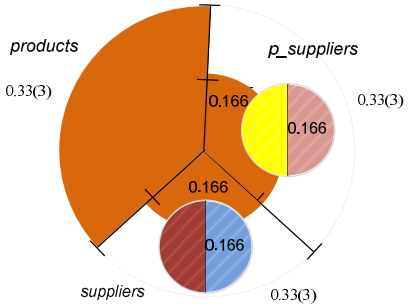


Figura 4.10 – Ilustração da quantificação de todas as teorias (ou programas lógicos) que interpretam a questão 1.

Tendo em consideração a medida da qualidade da informação associada a cada cenário, assim como a relevância de cada um dos predicados envolvidos na interpretação da questão, poderemos representar as possíveis soluções da seguinte forma:

$$Nome\_da\_Relação(a_1,\ldots,\ a_n,\ relevance,\ truth\_value)$$

onde a_i corresponde aos atributos do predicado, *Nome_da_Relação* corresponde à identificação do predicado (ou o nome do predicado), a *relevance* corresponde à relevância atribuída ao predicado e o *truth_value* corresponde ao valor de verdade associado ao registo (i.e. a quantificação da QoI). Seguindo esta representação, as possíveis soluções e o seu grau de confiança poderão ser apresentados no programa através da seguinte sintaxe:

1. *suppliers(#idS, nameS,ratingS,relevance,truth_Value)*
2. *p_suppliers(#idS,#idP, plafond,relevance,truth_Value)*
3. *companies(#idC,#idS,#idP,nameC,ratingC,relevance,truth_Value)*
4. *products(#idP,nameP,color,price,relevance,truth_Value)*

Tendo em consideração esta nova representação, estamos em condições de representar o *universo do discurso* através da *Programação em Lógica Estendida* na seguinte forma:

Nova representação das teorias através da Quantificação da Qualidade da Informação associada ao Programa Lógico ou cenário 1 {

$$\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers(X,Y,Z,R,T) \wedge$$

$$suppliers(1,\ supplier1,\ 20,\ 0.25,\ 0.166) \wedge$$

$$\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers(X,Y,Z,R,T) \wedge$$

$$p\_suppliers(1,50,200,0.15,0.166) \wedge$$

$$\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$$

$$products(50,product1,black,200,0.4,0.33)$$

}

A representação de todos os cenários é apresentado no anexo A. Adicionalmente, tendo em consideração esta representação e as teorias (ou cenários) obtidas através do processo de inferência (apresentadas no Anexo A), poderemos transformar a informação desconhecida em informação quantificável através da representação das relações das entidades da estrutura típica dos sistemas de gestão de bases de dados [MUW2002]:

suppliers	*IdS*	*nameS*	*ratingS*	*Rel*	*TV*
	1	supplier1	20	0.25	0.166
	2	supplier2	60	0.3	0.166

p_suppliers	*IdS*	*idP*	*plafond*	*Rel*	*TV*
	1	50	200	0.15	0.166
	2	51	200	0.2	0.166
	1	50	400	0.25	0.33

companies	*IdC*	*idS*	*idP*	*nameC*	*Rat*	*Rel*	*TV*
	2	1	50	company1	200	0.7	0.66

products	*idP*	*nameP*	*Color*	*Price*	*Rel*	*TV*
	50	product1	black	200	0.5	0.33
	51	product2	white	250	0.6	0.33

Com base nesta nova representação da informação incompleta em informação quantificável, consideremos que pretendemos listar quais os fornecedores que no presente estão em condições de fornecer produtos de cor preta.

No sentido de interpretar esta questão e obter as respostas com melhor qualidade de informação, consideremos o operador Ψ definido da seguinte forma: Sendo A e B duas relações de uma bases de dados, tendo A os seguintes atributos $(K, X_1 .. X_n, Rel, TV)$, tais que n $\geq$ 0 e B os atributos $(K, Y_1 .. Y_m, Rel, TV)$, tais que m $\geq$ 0 e

$$A\ \Psi_{(K)}\ B = \Pi_{K,X1..Xn,Y1..Ym,((A.Rel+B.Rel)/2,(B.TV+B.TV))}\ (A \infty_K B)$$

onde Π e ∞ denotam, respectivamente, os operadores de projecção e junção da álgebra relacional [MUW2002]. Desta forma, é possível obter a resposta à questão colocada ao sistema através da seguinte operação:

$$answer = (suppliers \Psi_{(\#idS)} ((\sigma_{plafond>0}(p\_suppliers)) \Psi_{(\#idP)} (\sigma_{Color=black}(products)))$$

onde σ corresponde ao operador de selecção na ágebra relacional [MUW2002], e a *answer* é dada em termos dos seguintes atributos:

$$answer(\#idS, \#idP, name, plafond, nameP, Color, Price, newTruthValue)$$

sendo a extensão da relação *answer* ordenada de acordo com o grau de confiança (i.e. em termos do seu valor de verdade "TruthValue" apresentado da seguinte forma:

answer(1,50,supplier1,200,product1,black, 200,0.3,0.66)
answer(2,50,supplier2,400,product1,black,200,0.35,0.826)

Através desta nova representação e da sua adaptação à estrutura típica dos sistemas de gestão de bases de dados, torna-se possível tirar partido de todas as potencialidades dos operadores de álgebra relacional, assim como do potencial da *Structured Query Languages (SQL)* [MUW2002] na invocação do mecanismo de inferência.

Tradicionalmente, nos Sistemas de Gestão de Bases de Dados, o SQL permite incluir atributos que contemplem, além do valor de verdade ("Truth value") verdadeiro ou falso, também o valor "NULL"(designado também por "null value"). No entanto, existem várias interpretações que podem ser adoptadas nos valores nulos [MUW2002, secção 6.1.5] sobe os quais podem ser efectuadas comparações contemplando um terceiro "truth value"denominado "unknown"[MUW2002, secção 6.1.6].

Neste sentido, usando a sintaxe do *PROLOG*, poderemos questionar a base de conhecimento de modo a identificar "*Quais os fornecedores que no presente estão em condições de fornecer produtos de cor preta*". A questão a colocar ao sistema que deve ser interpretada pelo mecanismo de inferência é a seguinte (24):

$\forall(A, B, C, D, F, N, P, L))$ $\neg$ *interpretar([suppliers(A,B,C),(p_suppliers(A,D,F), F>0), products(D,N,black,P)],L).*

onde o predicado "interpretar" (ver o conceito de "interpretação descrito na secção 2.3.2) tem como objectivo "caminhar" no processo de inferência através das dependências de relacionamento envolvendo os predicados utilizados na tentativa de resolver esta questão. Neste sentido, no final do processo de inferência a variável *"L"* conterá uma lista de todas as soluções possíveis que satisfazem o problema, ou por outras palavras, corresponderá ao "*Output*" do *intelecto virtual*. No anexo *A* são apresentadas as soluções para a questão a interpretar.

Seguindo o descrito na figura 4.3 sobre o circuito processual para a resolução do problema e a criação do *intelecto virtual* arquitectural, a questão (24) é submetida ao sistema. O passo seguinte corresponde à identificação de todas as entidades envolvidas

na interpretação da questão colocada, tendo em consideração o *universo do discurso* e as dependências de relacionamento envolvidas por essas entidades.

Neste exemplo, as entidades envolvidas na interpretação da questão são os "suppliers", os "p_suppliers" e os "products". Por este facto, o primeiro nível do intelecto virtual arquitectural corresponde à representação em grafo da estrutura das entidades envolvidas na resolução da questão.

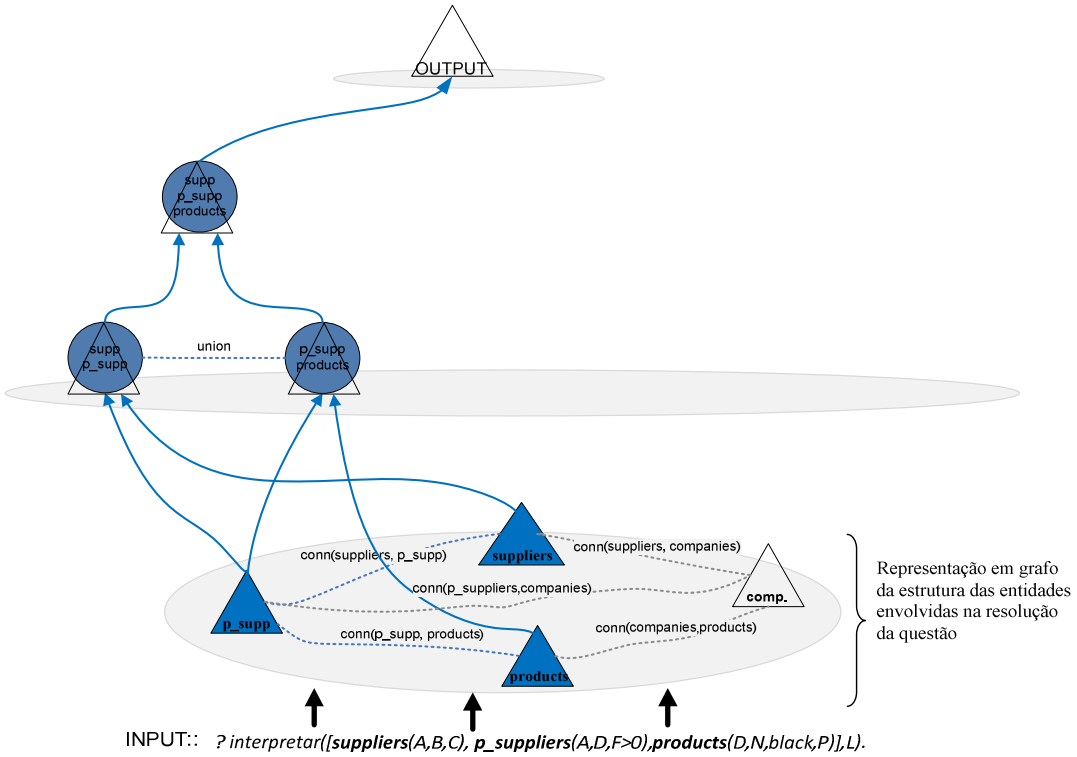


Figura 4.11 – Exemplificação da criação do Intelecto Virtual para uma Questão num momento t=1.

Na figura 4.11 ilustra-se o esquemático arquitectural do intelecto que permite responder à questão colocada, o qual corresponde a uma visão abstracta da solução para um determinado problema. Temos no entanto de salientar que a questão (24) submetida ao sistema implementado recorrendo ao *PROLOG*, o símbolo "¬"é substituído por "?"que corresponde ao símbolo de falsidade. Tal como apresentamos na secção 2.3.4, utilizamos o mecanismo de prova ou demonstração recorrendo à técnica de redução ao absurdo, em que se nega o que pretendemos provar juntamente com o *universo do discurso* e chegamos a uma inconsistência, provando o teorema que

pretendemos demonstrar.

Neste sentido, no primeiro nível são activados apenas os "neurónios" envolvidos na interpretação da questão, neste caso aquelas associadas às entidades "suppliers", "p_suppliers" e "products". No segundo nível, as entidades envolvidas correspondem à junção das ligações entre as entidades envolvidas na questão. Neste caso foram criadas duas entidades correspondendo às duas ligações existentes entre "suppliers" e "p_suppliers" e, entre "p_suppliers" e "products", correspondendo ao contexto que modela o *universo do discurso* nesse neurónio.

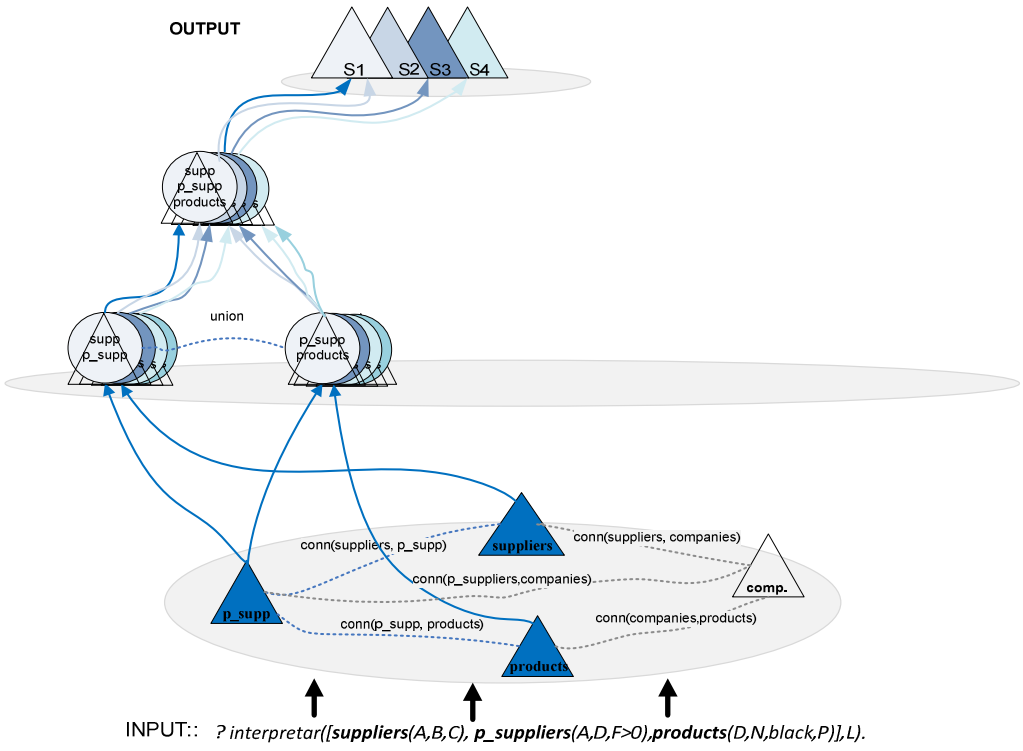


Figura 4.12 – Exemplificação da instanciação do intelecto virtual para uma questão num momento t=1.

Tendo em consideração a união destes dois últimos neurónios em virtude das dependências de relacionamento existentes, é realizada a união, sendo criado um novo neurónio correspondendo ao *universo do discurso* associado às extensões dos predicados que interpretam a questão, neste caso "suppliers", "p_suppliers" e produtos. Na camada de saída (designada também por *output*) irá ser obtida a lista de soluções possíveis para a interpretação da questão em causa, sendo o contexto quantificado

durante o processo evolutivo pela *QoI* associada a cada programa lógico (anexo A). Neste sentido, todas as soluções poderão ser extraídas, sendo seleccionadas apenas aquelas que maximizam o *universo do discurso* para a resolução do problema submetido ao sistema. Essas soluções são esquematizadas na figura 4.12 na camada de "output" através da identificação *S1, S2, S3 e S4*, correspondendo aos cenários (teorias ou programas lógicos) que interpretam e resolvem as questões colocadas ao sistema (descritas com mais detalhe no anexo A).

Desta forma, é definido inicialmente o intelecto virtual (ou arquétipo) com todas as soluções possíveis e, há medida que se "caminha" no processo de inferência a estrutura do intelecto é instanciada. Seguindo as características associadas à computação evolucionária e tendo em consideração a estrutura do genoma apresentada na secção 4.4.2 e os genes de processamento e de conexão, o genoma arquitectural da solução para o problema (24) pode ser representado da seguinte forma:

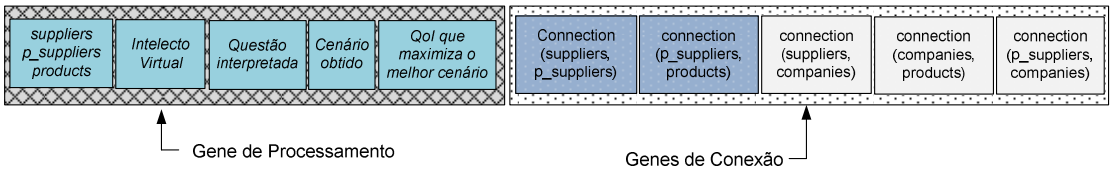


Figura 4.13 – Exemplificação do genoma arquitectural de uma solução para a interpretação de uma questão no momento t=1.

Tal como referimos, este genoma é constituído por um gene de processamento e vários genes de conexão. O gene de processamento é composto pelo contexto associado ao problema, ou seja: as extensões dos predicados que modelam o *universo do discurso* (no exemplo da figura as as extensões dos predicados "suppliers", "p_suppliers"e "products"); o intelecto virtual que corresponde ao esquema arquitectural do processo de inferência para a interpretação da questão em observação; o cenário obtido e o valor da *Qualidade da Informação* que maximiza o melhor cenário na resolução da questão em causa. Neste genoma, os genes de conexão correspondem às dependências de relacionamento existentes entre as entidades envolvidas, assim como as conexões subjacentes à estrutura de todas as entidades. Por exemplo, na figura 4.13, as conexões utilizadas são "connection(suppliers, p_suppliers)" e "connection(p_suppliers, products)" enquanto que, as outras correspondem às ligações entre entidades, no sentido

de assegurar a estrutura dos relacionamentos do *universo do discurso*. Será com base na aplicabilidade dos operadores genéticos nestes genes de conexão que se irão criar novos indivíduos (i.e. novas potenciais soluções).

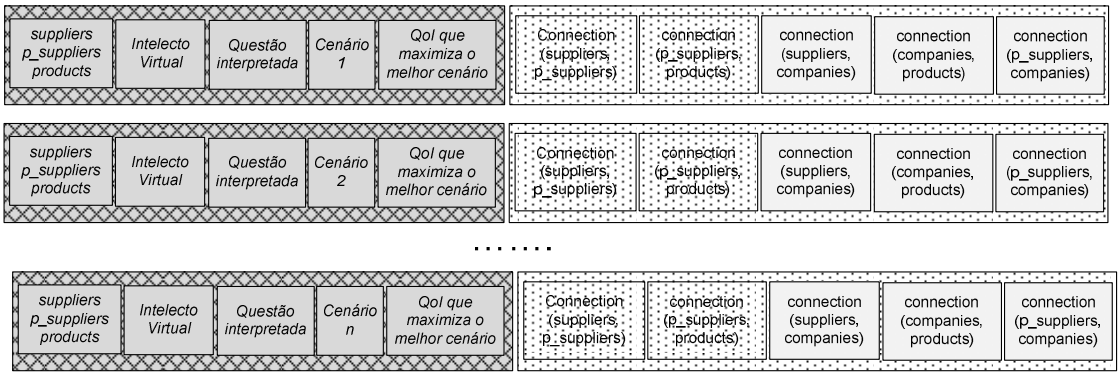


Figura 4.14 – Exemplificação do genoma das soluções para a interpretação de uma questão.

Tendo em consideração esta representação para o genoma geral, após a instanciação do intelecto virtual (figura 4.12), os indivíduos que formam a "primeira" população de soluções é esquematizado seguindo a representação da figura 4.13, sendo ilustrada na figura 4.14.

4.4.6 Processo Evolutivo

No sentido de alcançar o processo evolutivo e encontrar o melhor programa lógico ou teoria que maximiza o *universo do discurso*, ou por outras palavras, para modelar o *universo do discurso* num ambiente dinâmico, a estrutura de dois ou mais predicados é projectada numa fusão do espaço, que herda uma estrutura parcial dos seus *inputs*, apresentando uma nova estrutura por si só.

Esta secção apresenta o modelo conceptual e a quantificação do *universo do discurso* ao longo do processo evolucionário. Em problemas mais complexos o relacionamento entre predicados é mais óbvio em que a selecção de uma representação abstracta e lógica origina a que o seu desempenho seja claramente mais complexo.

Por outro lado, os indivíduos (neste caso programas lógicos) têm de ser testados pela sua capacidade de se adaptarem a ambientes dinâmicos, fazer deduções, efectuar

inferências e seleccionar o melhor percurso de acção a partir de um conjunto de alternativas. Apesar de tudo, os indivíduos terão de aprender como fazer estas tarefas por si só, não implementando nenhuma instrução específica por parte do utilizador (ou programador), mas pela resposta contínua da informação positiva e negativa dada pelo ambiente.

As *Redes Neuronais Evolucionárias* [Yao1999] [FSM2008] têm a particularidade de introduzir na sua arquitectura vários níveis, criando dinamismo na estrutura da rede. É comum nas camadas de baixo nível da rede, a introdução do treino dos pesos afectos aos pesos das conexões da rede. O pseudo-código de um *Algoritmo Evolucionário* orientado para a geração de redes evolucionárias no contexto deste trabalho de doutoramento é o apresentado na figura 4.15.

Algoritmo 2 – Geração_da_Rede_Evolucionária (*Teoria, Questão, QoI_minima*)

```
1.  Obter teorias (ou cenários) iniciais:
        Teorias ← Geração_de_Teorias(Teoria,Questão);
        conjunto_Teorias ← ProcessoDeInferência(Teoria,Questão);
2.  Gerar aleatoriamente a população inicial P(i) no momento i = 0 e invocar o interpretador
    PROLOG para validar o individuo:
        P(i) ← Gerar_População(conjuntoTeorias,i);
3.  Avaliar cada indivíduo (no momento i=0):
        NovaPopulação ← aplicar_Medida_QoI(P(i), QoI_minima);
4.  Gerar Ligações:
        Gerar_Ligações(NovaPopulação,i) ;
5.  Repetir
6.      Seleccionar indivíduos "parentes"baseado na sua avaliação;
        Aplicar o operador de reprodução aos "parentes"e produzir descendentes e invocar o
7.      interpretador PROLOG para validar o individuo. A próxima geração P(i+1) é obtida a
        partir dos descendentes e dos possíveis parentes.
8.      Avaliar cada indivíduo;
        NovaPopulação ← aplicar_Medida_QoI(P(i), QoI_minima);
9.      Gerar Ligações:
        Gerar_Ligações(NovaPopulação,i) ;
    até que o número máximo de iterações ou que a população tenha convergido para um valor
10. mínimo de qualidade;
    Extraír os índividuos resultantes, sendo aplicada a relação de ordem da quantificação da
11. qualidade da informação associada a cada um deles.
```

Figura 4.15 –Pseudo código da criação de uma Rede Neuronal Evolucionária.

Tipicamente neste tipo de redes em que o objectivo é a criação dinâmica da sua

estrutura, a evolução da topologia da rede é introduzida ao longo das várias camadas (ou níveis) através de mecanismos de treino/aprendizagem [Abra2002]. O passo seguinte na criação das *Redes Evolucionárias* corresponde à pesquisa dos pesos das conexões. Nestes ambientes, o processo de treino pode ser formulado com uma pesquisa global de pesos das conexões dentro de um conjunto óptimo definido pelo *Algoritmo Evolucionário*. Neste sentido, os pesos óptimos das conexões (*QoI* transportada ao longo do arquétipo) podem ser formulados como um problema de pesquisa global onde dentro da arquitectura da rede é predefinido e fixado um limite durante a evolução da arquitectura do intelecto aquando da sua instanciação. O algoritmo típico para a pesquisa dos pesos das conexões utilizando os *Algoritmos Evolucionários* é ilustrado na figura 4.16.

Algoritmo 3 – Geração das ligações

1. Gerar aleatoriamente a população inicial e seleccionar dois indivíduos com base nos predicados envolvidos na resolução da questão;
2. Seleccionar em cada indivíduo uma conexão utilizada na resolução da questão colocada;
3. Seleccionar em cada indivíduo uma ligação não utilizada na resolução da questão e que tenha as mesmas dependências de relacionamento entre os predicados associados ao contexto do indivíduo;
4. Verificar se as conexões resultantes são válidas na estrutura de ligação das dependências de relacionamento e avaliar a QoI associada por esses indivíduos, no sentido de verificar se a nova ligação irá originar num aumento da *QoI* face ao *Threshold* previamente definido. Caso contrário, executar novamente o passo 1 (até explorar todas as combinações de geração de indivíduos. Caso sejam válidas executar o passo 5.
5. Constituir o gene de conexão com as conexões seleccionadas;

Figura 4.16 –Pseudo código para a pesquisa dos pesos das conexões de uma Rede Neuronal Evolucionária.

De modo a avaliar os indivíduos, para cada sub-problema o sistema submete a questão a ser testada pelo programa lógico formado pela união das cláusulas codificadas pelo indivíduo. Tendo em consideração o exemplo apresentado na secção 4.4.5, consideremos que após a invocação ao sistema da questão (24) pretendemos saber "*Quais os fornecedores que no passado forneceram produtos de cor preta*", ou seja

provar o seguinte teorema (25):

$\forall(A, B, C, D, N, P, X, L, W, Z)) \neg$ *interpretar([suppliers(A,B,C),*

companies(A,D,P,W,Z),products(P,N,black,X)],L).

Seguindo o mesmo processo apresentado para a questão (24), a questão a interpretar pelo sistema irá criar um novo arquétipo que corresponde à representação em grafo da estrutura que irá alimentar o processo computacional para a resolução da questão (ilustrada a azul na figura 4.17).

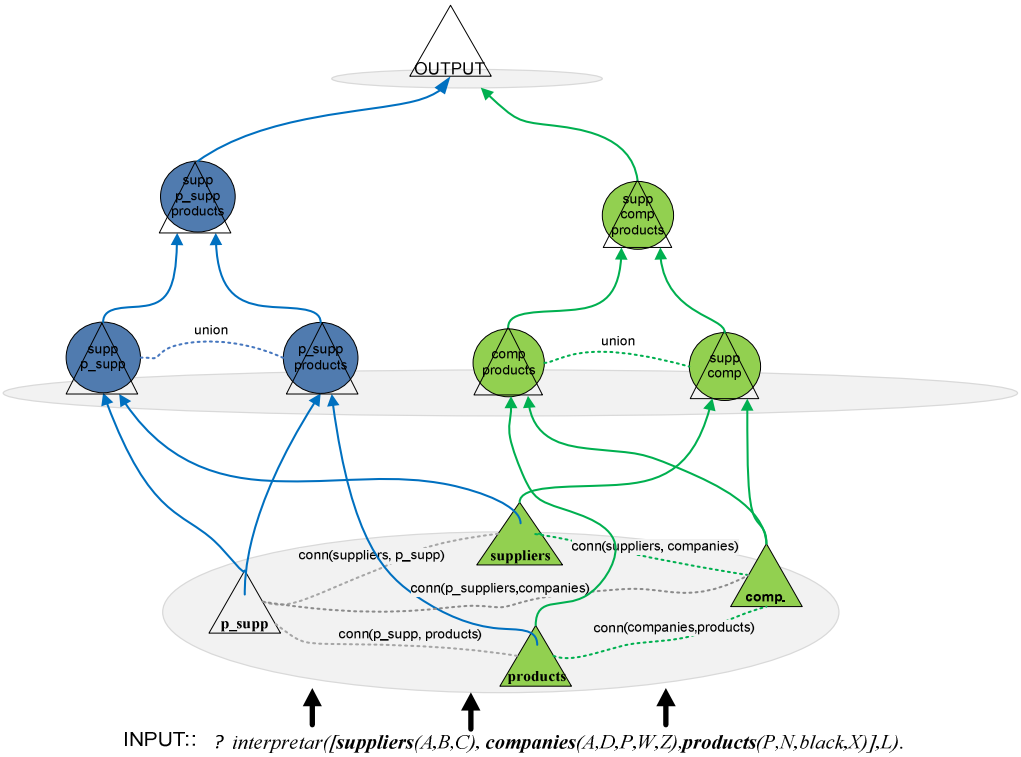


Figura 4.17 - Exemplificação da criação do intelecto virtual para uma questão num momento *t=2*.

Na figura 4.17 ilustra-se a criação do arquétipo que permite interpretar a questão (25). Para a resolução da questão são activados três entidades, nomeadamente, "sup-

pliers", "companies" e "products". Tendo em consideração as dependências de relacionamento entre estas entidades, são criados num segundo nível os neurónios correspondentes à sua junção. Num terceiro nível é criado um novo neurónio referente à união dos antecedentes.

Neste sentido, o anterior intelecto (figura 4.11) irá actualizar-se com este novo, isto é, o intelecto criado para interpretar a questão (24) é armazenado (a azul na figura 4.17) e no final da interpretação da questão (25) é actualizado com o arquétipo da interpretação da questão (25) (a verde na figura 4.17). Tal como se apresentou na figura 4.14, também as soluções (ou cenários) obtidas para a resolução desta questão são representados em forma de genoma (figura 4.18).

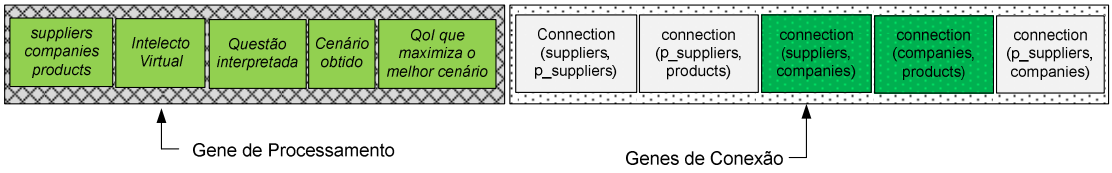


Figura 4.18 - Exemplificação do genoma arquitectural de uma solução para a interpretação de uma questão no momento t=2.

Após a obtenção das soluções para a questão colocada, o intelecto virtual irá autoorganizar-se e aprender com a informação obtida, no sentido de criar novas questões e respectivas soluções, onde há partida não eram descobertas com a invocação particular de cada uma das questões anteriores. Se analisarmos a figura 4.19, até à invocação das duas questões anteriores foram seleccionadas as quatro entidades capazes de interpretar e resolver os problemas colocados (24) e (25). No entanto, existe uma dependência de relacionamento entre a entidade "suppliers" e "companies" que está subjacente à estrutura das entidades que modelam o *universo do discurso*, mas que por força da invocação particular das questões colocadas não foi utilizada.

Neste exemplo do processo evolutivo, os genes de conexão são parametrizados da forma ilustrada na figura 4.20. Conforme referimos na secção 4.4.4 a parametrização do gene de conexão permitirá determinar como cada gene pode ser conectado com outro. Por este facto, a pesquisa de soluções durante o processo de inferência poderá

seguir vários caminhos com o aumento do número de relacionamentos entre entidades.

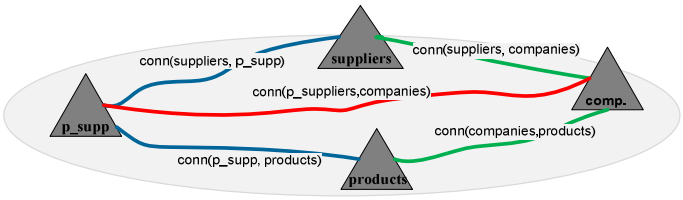


Figura 4.19 - Dependências de relacionamento na interpretação das questões colocadas ao sistema.

A parametrização foi efectuada da seguinte forma: o gene de contagem especifica que a solução contém três conexões. O gene de direcção determina a orientação das conexões deste neurónio, neste caso o valor "3" indica que na pesquisa de soluções deverá ser seleccionado o neurónio de uma camada superior. O gene "alcance" determina quão longe dentro de uma camada simples a conexão poderá alcançar outros neurónios. Neste caso pretendemos que seja seleccionado o neurónio da camada do nível imediatamente superior, por este facto foi parametrizado com o valor "1". Neste sentido, através de um processo evolutivo, o intelecto irá descobrir esta ligação e criar uma nova estrutura arquitectural permitindo a obtenção de novas soluções.

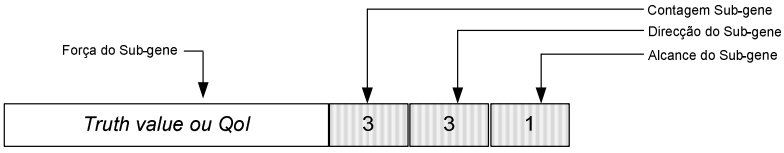


Figura 4.20 - Exemplificação da parametrização do gene de conexão.

Conforme apresentamos na secção 3.2.4 e na secção 4.4.4, a aplicabilidade dos operadores genéticos tem como objectivo aplicar operadores baseados na teoria da evolução das espécies, no sentido de gerar novos individuos a partir da população já existente. Em termos operacionais, o operador de cruzamento tem como entradas

dois elementos da população e um conjunto de parâmetros de controlo e tem como saída dois individuos resultantes da aplicação do operador de cruzamento num ou dois pontos de dois individuos.

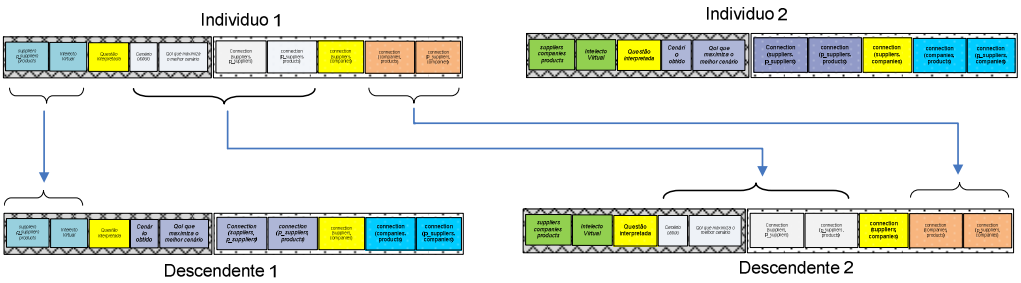


Figura 4.21 - Ilustração do processo de operacionalização do operador de cruzamento.

Na figura 4.21 ilustra-se a aplicação do operador de cruzamento em dois pontos. Neste caso são seleccionados os pontos referentes às questões interpretadas e as ligações entre as entidades envolvidas na interpretação da questão. Depois de gerados os novos dois indivíduos, estes serão interpretados pelo interpretador PROLOG no sentido de assegurar a sua coerência com o universo do discurso. Por outro lado, em ambos os indivíduos progenitores são contempladas todas as conexões existentes em termos das dependências de relacionamento entre entidades, no entanto, em cada um deles, apenas estão activadas (ou utilizadas) aquelas que envolvem o universo do discurso e a questão de cada um desses indivíduos. Por este facto, nos descendentes serão activadas novos relacionamentos no sentido de gerar novas possíveis soluções. A troca de informação entre cenários dos progenitores, serão transferidos para os novos descendentes onde será possível aliar o conhecimento de cada progenitor com os predicados e conexões envolvidas numa questão similar de outro progenitor. Por outro lado, em cada individuo o gene de conexão contem a informação das conexões assim como os parâmetros de controlo (figura 4.20) que permitirão agilizar o processo de pesquisa de novas soluções.

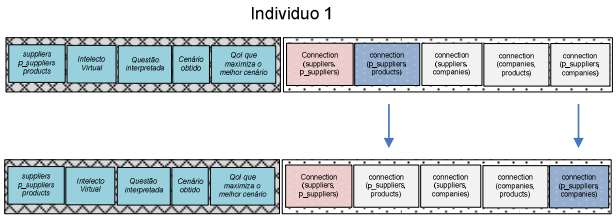


Figura 4.22 - Ilustração do processo de operacionalização do operador de mutação.

Por outro lado, no processo de mutação, as entradas correspondem a um individuo com um conjunto de parametros de controlo, tendo como sada um individuo. Neste trabalho, a aplciabilidade dos operadores semelhante, no entanto depois da geração dos individuos executado o interpretador *PROLOG*, no sentido de validar o individuo resultante (teoria ou programa lógico). Por outro lado, no sentido de assegurar que a mutaçã não perturbe a semântica da criação do intelecto virtual, ela nunca pode deixar um gene de processamento sem pelo menos conter um gene de conexão. Na figura 4.22 ilustra-se um exemplo da aplicação do operador de mutação num único ponto, sendo no individuo descendente seleccionado (ou activado) um outro gene de conexão, no sentido de tentar encontrar novas ligações entre as entidades envolvidas na questão a interpretar. De salientar, que tanto no operador de cruzamento de mutação, a invocação do interpretador *PROLOG* permitirá avaliar a coerência e semântica associada à informação do seu *universo do discurso*.

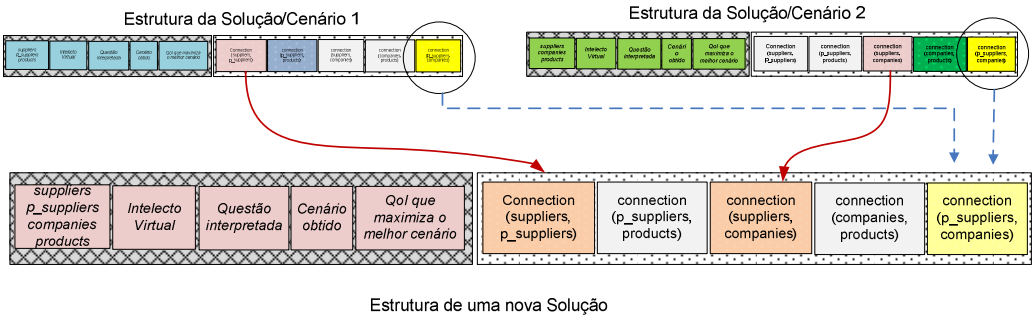


Figura 4.23 - Exemplificação da criação de novo conhecimento a partir das duas questões colocadas ao sistema.

Na figura 4.23 ilustra-se o processo de criação de novo conhecimento a partir das duas soluções (ou cenários) obtidos para a interpretação e resolução das duas questões invocadas ao sistema. O processo de criação de uma nova solução baseou-se na identificação aleatória de um gene de conexão em comum entre as soluções e que tivesse sido activado para a resolução da questão colocada ao sistema. Através da selecção aleatória de um gene de conexão ainda não usado nas questões colocadas de forma particular, foi possível identificar uma nova ligação entre as quatro entidades envolvidas obtendo-se assim uma nova solução que permite que o intelecto aprenda com novas situações que vão aparecendo sem esquecer a informação acumulada ao longo do tempo.

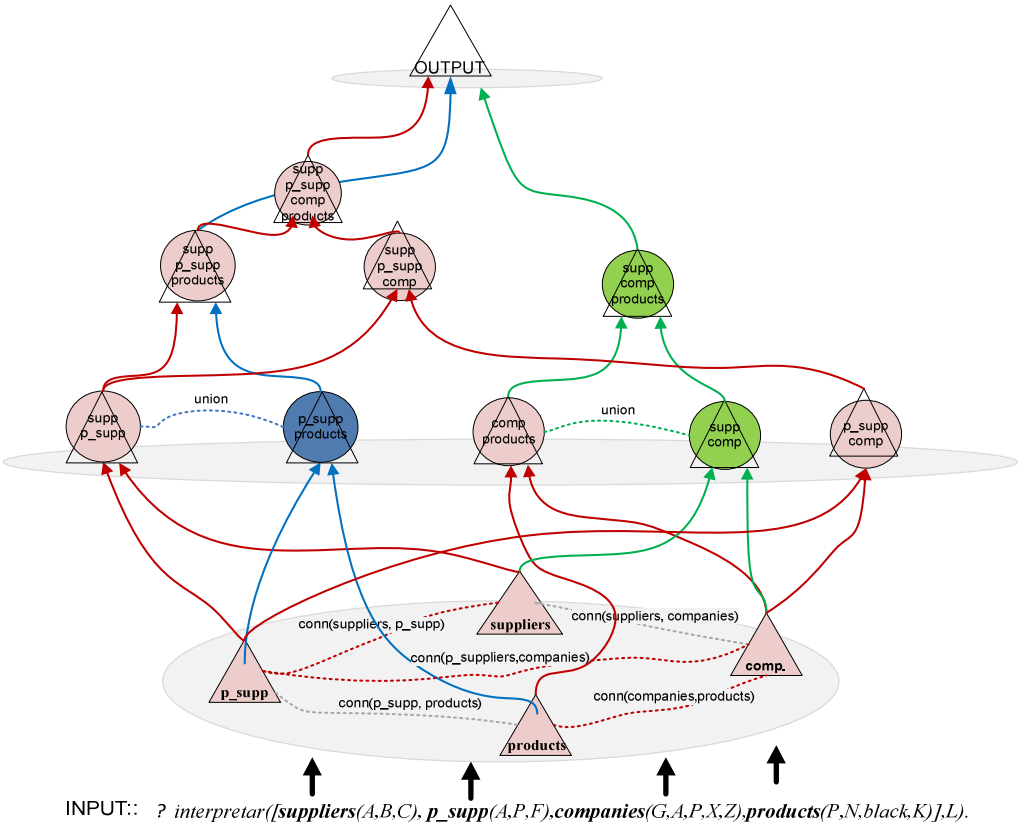


Figura 4.24 - Exemplificação da autoorganização do Intelecto Virtual ***após*** *o momento t=2.*

Tendo em consideração o exposto, o relacionamento deste trabalho com a Programação Genética, em termos da sua topologia (figura 3.2) e aplicação dos operadores genéticos (figuras 3.3 e 3.4), a representação de cada *universo do discurso* em cada individuo (primeira componente do gene de processamento) e a informação dos genes de conexão é representada em forma de árvore (através de fórmulas lógicas) [NM$_+$2007], sendo aplicados os operadores conforme referido.

Desta forma, há medida que o *universo do discurso* se torna mais complexo, novos relacionamentos vão aparecendo, permitindo descobrir novas questões que possam ser colocadas ao sistema as quais não seriam possíveis por si só de serem descobertas através da invocação particular de questões colocadas ao sistema.

Através da análise dos invariantes ou dependências de relacionamento), identifica-se que há um relacionamento entre as entidades envolvidas nas questões. Na figura 4.23 ilustra-se o exemplo da actualização do arquétipo. Desta forma o sistema é capaz de descobrir a questão *"Quais os fornecedores que no passado forneceram produtos de cor preta e que estão no presente em condições de os continuar a fornecer?"* que poderá ser invocada através do seguinte teorema a demonstrar (26):

$\forall(A, B, C, F, G, K, L, N, P, X, R, Z))$
*interpretar([**suppliers**(A,B,C),**p_supp**(A,P,F),**companies**(G,A,P,X,Z),*

***products**(P,N,black,K)],L).*

No seguimento das questões invocadas ao sistema e, tendo em consideração todos os possíveis cenários que poderão satisfazer a questão (26), o intelecto é optimizado na resolução de problemas, procedendo-se à optimização do cenário que consideramos para a resolução de problemas. Neste sentido, na camada de saída iremos obter o arquétipo e a lista de soluções que resolvem o problema (26), cujas soluções são detalhadas no *anexo A*.

Desta forma, a partir de uma associação de dois ou mais casos (criação do intelecto para a interpretação das questões (24) e (25)) advém um novo potencial de inferência que é condicionado pelas regras de associação (dependências de relacionamento entre predicados). Este processo leva à criação de novas extensões dos predicados, criando novas ligações que concretizam o processo de inferência por outros caminhos entre os predicados que até ao momento não tinham sido explorados (intelecto a vermelho na

figura 4.24).

Usando a medida da *QoI* sobre as extensões dos predicados associados ao contexto de cada neurónio, durante o processo de inferência (instanciação dos termos ou variáveis dos predicados) a medida transportada pela *QoI* em cada ligação é avaliada e ponderada. Caso esteja abaixo de um valor desejável (ao qual chamamos *threshold*), o processo de inferência pára uma vez que o grau de confiança para a interpretação da questão colocada ao sistema não é possível de atingir face a todo o processo que ainda falta desenvolver para interpretar a questão. Por este facto, o caminho usado no processo não é seguido (ou seja a instanciaçã do intelecto é "truncada"), procedo-se à exploração de novos caminhos dentro do intelecto até encontrar a(s) solução(ões) com um grau de confiança aceitável.

4.5 Conclusão

O objectivo principal deste trabalho de doutoramento centra-se na definição de um sistema computacional (ou modelo computacional) denominado por intelecto virtual (or arquétipo), que combina as vantagens associadas aos paradigmas, simbólico, conexionista e evolutivo, no sentido de optimizar o *universo do discurso*.

Na nossa abordagem o processo de aprendizagem é baseado na evolução dos indivíduos, sendo construído através do processo de quantificação da *Qualidade da Informação* associada a um programa lógico. Estes programas lógicos representam uma população de soluções candidatas que modelam o *universo do discurso*. Na nossa abordagem, a *Programação Genética* e a *Programação em Lógica Estendida* não irá obter uma solução para um problema particular, mas antes uma representação lógica (ou programa) do *universo do discurso* a ser optimizado.

Com este modelo computacional para tratar informação incompleta num ambiente evolucionário, algumas vantagens podem ser apresentadas, nomeadamente:

1. Avaliar e qualificar em tempo real o conhecimento de cada entidade e seleccionar a "óptima"(programa lógico ou teoria) que modela o *universo do discurso* e maximiza a *Qualidade da Informação* associada ao conhecimento de cada entidade em cada momento do processo evolutivo;

2. A selecção dos caminhos mais promissores baseados na *Qualidade da Informação* que cada entidade transporta;

3. Melhorar a tarefa computacional num ambiente evolucionário, no sentido de resolver problemas através da selecção dos melhores predicados baseados na *Qualidade da Informação*;

4. Através de uma nova representação, é possível aplicar os procedimentos da álgebra relacional [MUW2002] para as soluções possíveis, seleccionado apenas os melhores registos com um maior grau de confiança;

5. Usando os operadores de combinação associados ao paradigma genético ou evolucionário, é possível encontrar novas soluções para os problemas a resolver. Neste caso procuramos descobrir as extensões dos predicados do tipo já referido.

Em termos práticos, no final da criação do modelo iremos alcançar um valor (e uma teoria) que corresponderà quantificação do *universo do discurso* optimizado ao longo do tempo. Sabendo esse valor, o qual não é uma probabilidade, mas sim o valor da qualidade do conhecimento disponível, é possível obter a melhor teoria lógicomatemática (representada através de programas lógicos) e consequentemente, a melhor modelação do problema em observação.

"Let every man be respected as an individual and no man be idolized"
[Einstein in His own words, Anne Rooney,2006]

Capítulo 5

Inteligência Evolucionária - Operacionalização do Sistema

5.1 Introdução

Através da utilização do paradigma da *Programação em Lógica Estendida* para a representação do conhecimento e disponibilização de mecanismos de raciocínio em ambientes em que a informação é incompleta, o principal objectivo deste trabalho é descobrir quais as teorias (ou programas lógicos) capazes de resolver (ou interpretar) um determinado problema. Por outro lado, pretendemos criar um sistema capaz de avaliar perante esse conjunto de teorias, quais as melhores para resolver um problema específico. Neste sentido, baseados nos formalismos apresentados no capítulo anterior, pretendemos operacionalizar um sistema tirando partido do paradigma conexionista e evolutivo, de modo a avaliar e seleccionar as melhores teorias lógico matemáticas expressas através do paradigma simbólico. Através da apresentação de um exemplo simples, iremos descrever a forma de representar o conhecimento (em cenários em que a informação é incompleta), assim como a aplicabilidade do mecanismo de inferência e da medida da *Qualidade da Informação* associada aos programas lógicos (materializados aqui através da *Programação em Lógica Estendida*). Na secção 5.3 apresentamos o *VirtualInspector*, um sistema que materializa o modelo computacional especificado no capítulo quatro.

Iniciamos este capítulo através de um exemplo simples na área dos sistemas de gestão de bases de dados [MUW2002], de modo a apresentar a forma de imple-

mentação da representação do conhecimento e mecanismo de raciocínio em ambientes em que a informação apresenta características de imperfeição. No final deste capítulo apresentamos as conclusões e trabalho futuro a realizar, no sentido de melhorar as funcionalidades do *Virtual Inspector*.

5.2 Representação da Informação e Mecanismo de Raciocínio

5.2.1 Representação da Informação

Neste trabalho de doutoramento utilizamos uma representação muito próxima da linguagem de programação *PROLOG* [CM1981] [Brat1990] [SS1994]. Neste sentido, podemos escrever uma cláusula na sintaxe *PROLOG*, onde o símbolo " $\leftarrow$ " é substituído por ":-" e " $\wedge$ " por ",". Em termos de implementação usando o *PROLOG* e perante as definições apresentadas no capítulo quatro, torna-se necessário proceder à definição de alguns predicados que possam estender a linguagem de *Programação em Lógica*. Em primeiro lugar, é necessário definir uma sintaxe para a representação da negação forte. Para o efeito utilizamos o símbolo "*-(A)*", para significar a negação explícita de A, ou seja, para representar que A é falso. O operador ***not*** **(ou "nao")** é, por vezes, utilizado representando a negação por falha. Contudo, em termos de implementação recorrendo ao *PROLOG* já existe uma rotina com o mesmo nome "*not*". Por este facto, neste capítulo iremos utilizar a numenclatura "*nao*"para nos referimos a "*not*", sendo aqui representado da seguinte forma:

```
nao( Questao ) :- Questao, !, fail.
nao( Questao ).
```

A representação do *Pressuposto do Mundo Fechado* para os predicados, pode ser implementado da seguinte forma:

```
-(X) :- nao(X),
        nao(abducible(X)).
```

Neste sentido, considerando o exemplo apresentado na secção 4.4.5 a representação do *Pressuposto do Mundo Fechado*, por exemplo para o predicado "suppliers"pode ser descrito da seguinte forma:

-suppliers(X,Y,Z) :- nao(suppliers(X,Y,Z)),
nao(abducible(suppliers(X,Y,Z))).

Seguindo a terminologia apresentada no capítulo quatro em termos da representação em *Programação em Lógica Estendida*, poderemos representar a informação positiva e negativa da seguinte forma:

suppliers(2,supplier2,60).
suppliers(3,supplier3,25).
-suppliers(4,supplier4,80).

em que os dois primeiros predicados representam a informação positiva (conhecida e verdadeira) e o último representa a informação negativa (conhecida, mas falsa). Conforme apresentamos no capítulo quatro, para a representação de informação incompleta temos três tipos: informação desconhecida, informação desconhecida mas possível de ser seleccionada a partir de um conjunto de valores e, informação desconhecida mas possível de ser seleccionada um valor a partir de um conjunto de valores. Para o primeiro caso, poderemos a título de exemplo, considerar que *"o rating do fornecedor com o identificador '5' é desconhecido e não pode ser conhecido"*, ou seja, *"não se pode acrescentar informação acerca do rating deste fornecedor"*. Neste caso, poderemos representar a informação do seguinte modo:

suppliers(5,supplier5,not_permitted_to_known).

No entanto, neste caso temos de especificar um invariante que indique que esta informação é desconhecida. Isto pode ser realizado da seguinte forma:

abducible(suppliers(X,Y,Z)):-suppliers(X,Y,not_permitted_to_known).
null_value(not_permitted_to_known).

Por outro lado, torna-se necessário especificar um invariante que restrinja a inserção de informação associada a este "supplier". Este invariante, pode ser especificado da

seguinte forma:

+suppliers(X,Y,Z)::(solucoes((X,Y,Z),(suppliers(5,Y,Z),
nao(null_value(Z))),S), comprimento(S,N), N == 0).

em que o predicado "soluções"tem como objectivo simplicar a designação do predicado de pesquisa de todas as soluções existentes na base de conhecimento. Este predicado recorre ao predicado *"findall"* que é predefinido nos sistemas baseados em *PROLOG*. A representação de informação desconhecida mas possível de ser seleccionada a partir de um conjunto de dados, por exemplo, *"o rating do fornecedor com o identificador 1 é desconhecido, mas apenas pode ser seleccionado um valor a partir do conjunto de possíveis valores 20 ou 40"*. Esta situação e o invariante que restringe a escolha dos possíveis valores, pode ser representada da seguinte forma:

abducible(suppliers(1,supplier1,20)).
abducible(suppliers(1,supplier1,40)).
?((abducible(suppliers(X1,Y1,Z1)),abducible(suppliers(X2,Y2,Z2))),
- (abducible(suppliers(X1,Y1,Z1)),abducible(suppliers(X2,Y2,Z2)))).

Para a representação da informação desconhecida, mas possível de ser escolhida a partir de um conjunto de valores, sem qualquer restrição, ou seja, podendo ser seleccionado um valor individual ou em combinação com outros valores do conjunto, segue a mesma representação da situação anterior, mas sem a existência do invariante acima representado.

Neste exemplo, torna-se também necessário definir as dependências de relacionamento que permitem especificar a estrutura apresentada na secção 4.9. Neste sentido, o predicado *"connection"* tem como objectivo definir as ligações entre duas entidades, podendo ser representado da seguinte forma:

connection(suppliers(A,B,C),companies(D,A,P,N,R)).
connection(suppliers(A,B,C),p_suppliers(A,P,R)).
connection(p_suppliers(A,P,R),companies(D,S,P,N,Z)).
connection(p_suppliers(A,P,R),products(P,N,C,Z)).
connection(companies(D,S,P,N,R),products(P,NP,C,PR)).

Em termos da actualização (ou evolução) da informação existente na base de conhecimento, a sua inserção não pode ser directa, ou seja, a actualização da base de

conhecimento pode não ser possível caso existam restrições que o impessam.

Neste sentido, um invariante pode ser definido em *PROLOG*, a partir de um predicado lógico que deve ser cumprido no acto de inserir a informação. Assim, a inserção de informação na base de conhecimento poderá ser implementada da seguinte forma:

```
insert(P) : - assert(P),
              inv(P).
insert(P) : - retract (P).
```

O predicado *inv* deverá ser definido de forma a que seja verdadeiro quando todas as restrições impostas à introdução de nova informação sejam cumpridas. Como exemplo, implementemos o invariante de que não é possível na base de conhecimento o valor de um dado atributo p, para o objecto a, a sua implementação pode ser realizada da seguinte forma:

```
inv(p(X,Y)) :: soluções(_,p(X,np), p(X,Y), Y\==np), []).
```

Em que o predicado *soluções* pode ser implementado pelos predicados de recolha de todas as soluções para uma dada questão (e.g. por exemplo *findall*). Por exemplo, para a indicação de um invariante que reestringue a inserção de informação numa determinada condição, por exemplo não permitir que seja inserido o *rating* para o fornecedor com o identificador '5', poderemos representar esta situação desta forma:

```
+suppliers(X,Y,Z) :: (solucoes((X,Y,Z),(suppliers(5,Y,Z),nao(null_value(Z))),S),
comprimento( S,N ), N == 0 ).
```

Por outro lado, poderemos especificar um invariante estrutural no sentido de não permitir a inserção de conhecimento repetido. Poderemos representar esta situação da seguinte forma:

```
+suppliers(X,Y,Z) :: (solucoes( (X,Y,Z),(suppliers(X,Y,Z)),S ),
comprimento( S,N ), N == 1 ).
```

Adicionalmente, torna-se por vezes necessário, especificar a situação de determinado invariante que assegure a integridade referencial, por exemplo não admitir que mais do que dois "suppliers"existem para um mesmo identificador. Esta situação, pode

ser representada da seguinte forma:

+suppliers(X,Y,Z) :: (solucoes((X),(suppliers(X,Y,Z)),S), comprimento(S,N), N =< 2).

No sentido de efectuar o cálculo da medida da *Qualidade da Informação* descrita na secção 4.3.2, em especial para o cálculo da *card-combination* de n elementos em conjutos de k elementos. A fórmula do cálculo é dada por:

$$\mathbf{C}(n,k) = \mathbf{C}_n^k = \mathbf{C}_k^n = {}_nC_k = \binom{n}{k} = \frac{n!}{k!(n-k)!},$$

Uma forma de representar esta fórmula poderá ser através da especificação do seguinte predicado:

```
cardCombination(N,K,R) :- factorial(N,N1),
                          factorial(K,K1), R1 is N-K,
                          factorial(R1,N2),
                          R is (N1 / (K1*N2)).
```

em que o predicado *factorial*, corresponde ao cálculo do *factorial* de N elementos. Este e os restantes predicados que complementam a representação do conhecimento é descrita com mais detalhe no anexo B.

Para efectuar uma prova de uma questão Q, deve ser utilizado o meta predicado *demo* apresentado na definição 7. A implementação deste meta-interpretador pode ser expressa da seguinte forma:

```
demo(Questao, verdadeiro) : - Questao.
demo(Questao, falso) : - nao(Questao).
demo(Q, desconhecido) : - nao(Questao),
                          nao(-Questao).
```

Tendo em consideração a representação do conhecimento acima descrito, assim como o demonstrador de teoremas, estamos em condições de colocar ao sistema a interpretação da questão *"Quais os fornecedores que esto em condições de fornecer produtos de cor preta"*. Neste sentido, pretendemos que o sistema interprete a seguinte questão (27):

```
? interpretar([suppliers(X,Y,Z),(p_suppliers(X,P,R),R>0),
               products(P,NP,black,Pr)],L).
```

Depois de submetida a questão, o conjunto de soluções é o seguinte:

L=[Cenário 1, Cenário 2, ..., Cenário n]

Neste exemplo, cada um dos cenários (ou soluções) é descrito da seguinte forma:

Cenário 1 = {(X=1, Y=supplier1, Z=20, P=50, R=200, NP=product1,Pr=200)} :: 0.2316

Cenário 2 = {(X=1, Y=supplier1, Z=20, P=50, R=400, NP=product1, Pr=200)} ::0.2316

Cenário 3 = {(X=1, Y=supplier1, Z=40, P=50, R=200, NP=product1, Pr=200)} :: 0.2316

Cenário 4 = {X=1, Y=supplier1, Z=20, P=50, R=400, NP=product1, Pr=200)} :: 0.2316

em que, para cada cenário é calculada a qualidade da informação associada, sendo apresentado anteriormente na forma *Cenário n = {"Instanciação do universo do discurso que modela o problema em observação":: QoI}*. Por outro lado, se pretendermos colocar a questão *"Quais os fornecedores que no passado forneceram produtos de cor preta"*. A questão a colocar ao sistema é (28):

interpretar([suppliers(X,Y,Z),companies(A,X,P,N,R),
products(P,NP,black,Pr)], L).

Depois de submetida a questão, o conjunto de soluções e a sua quantificação (qualidade da informação associada) é o seguinte:

Cenário 1 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[40], NP=product1, Pr=200) :: 0.189}

Cenário 2 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[50], NP=product1, Pr=200) :: 0.189}

Cenário 3 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[60], NP=product1, Pr=200) :: 0.189}

Cenário 4 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[40,50], NP=product1, Pr=200) :: 0.181}

Cenário 5 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[40, 60], NP=product1, Pr=200) :: 0.181}

Cenário 6 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[50, 60], NP=product1, Pr=200) :: 0.181}

Cenário 7 = {(X=1, Y=supplier1, Z=20, A=2, P=50, N=company2, R=[40, 50, 60], NP=product1, Pr=200) :: 0.179}

Cenário 8 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[40], NP=product1, Pr=200) :: 0.189}

Cenário 9 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[50], NP=product1, Pr=200) :: 0.189}

Cenário 10 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[60], NP=product1, Pr=200) :: 0.189}

Cenário 11 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[40, 50], NP=product1, Pr=200):: 0.181}

Cenário 12 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[40, 60], NP=product1, Pr=200):: 0.181}

Cenário 13 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[50, 60], NP=product1, Pr=200):: 0.181}

Cenário 14 = {(X=1, Y=supplier1, Z=40, A=2, P=50, N=company2, R=[40, 50, 60], NP=product1, Pr=200):: 0.179}

Neste exemplo, tendo em consideração os resultados obtidos para cada uma das questões, poderemos seleccionar a melhor teoria que modela o *universo do discurso* em termos da interpretação das questões colocadas ao sistema. Por exemplo, para a segunda questão, a relação de ordem da *qualidade da informação* associada a cada cenário C_i que interpreta a questão a seguinte:

$$C_1 < C_2 < C_3 < C_8 < C_9 < C_{10} < C_7 < C_{14} < C_4 < C_5 < C_6 < C_{11} < C_{12} < C_{13} < C_7$$

em que $C_a < C_b < C_c .... < C_k$, significa que o valor de C_a é superior a C_b e assim por diante.

Neste sentido, tendo em consideração a relação de ordem da *qualidade da informação* associada a cada cenário capaz de interpretar (ou resolver) a questão, poderemos seleccionar o cenário que maximiza a *qualidade da informação*. Se considerarmos inúmeras dependências de relacionamento entre as entidades, assim como considerar que as questões colocadas envolvam várias entidades, o valor da *qualidade da informação* vai evoluindo ao longo do processo de inferência e, seguindo a selecção da(s) melhore(s) teorias (através de um *threshold* da *QoI* minimo), no final do processo evolutivo para a resolução de um determinado problema iremos obter a melhor teoria lógico matemática que maximiza o *universo do discurso*.

Por outro lado, a medida da *QoI* pode ser também aplicada a situações em vários atributos dos registos de uma relação de uma base de dados possa conter informaccão incompleta. Este problema de generalização da informação incompleta aplicada a um

ou a vários atributos, é aqui solucionada através da escolha dos melhores registos de uma relação e só depois escolhendo o melhor cenário para a extensão da relação.

Na secção seguinte iremos apresentar o *VirtualInspector*, um sistema protótipo que operacionaliza os conceitos apresentados no capítulo quatro no que se refere á aplicabilidade da Inteligência Evolucionária. Tendo em consideração as duas questões colocadas ao sistema apresentadas nesta secção, nomeadamente a (27) e (28), através da conjugação dos paradigmas simbólico, evolutivo e conexionista, o sistema irá aprender novo conhecimento, sendo exemplificada a descoberta de uma nova questão (26) e, por conseguinte novo conhecimento a disponibilizar ao utilizador do sistema.

5.2.2 Tratamento da Evolução Temporal da Informação

Tradicionalmente, as bases de dados acumulam informação ao longo do tempo, podendo em derminadas stuações a informação de uma entidade (ex. "supplier") ser alterada (ou negada) consoante os vários estados de evolução da informação armazenada nessas base de dados. Por este facto, torna-se necessário modelar as alterações da informação ao longo do tempo.

Seguindo a abordagem deste estudo, uma solução para proceder a esta modelação passa pela utilização da negação aplicada a um tuplo ou registo da base de dados. Em termos da dedução em bases de dados, uma actualização de informação negativa pretende-se que seja implicita no presente se a contraparte positiva não esteja explicita no presente, isto é, a falha na pesquisa de uma instância do tuplo ou registo de uma relação significa que a negação do tuplo é verdadeira. Desta forma, poderemos assumir [Nev1984] que:

"se uma cláusula 'p' não é uma consequência lógica dos dados na base de dados então infere 'not p'"

Por sua vez, numa base de dados lógica é desejável assegurar que a acumulação de informação adicional não altera a consistência da informação existente na base de dados. Por este facto, a base de dados deverá conter todos os registos de todas as transacções que ocorreram até ao actual momento, devendo o sistema ser responsável por permitir a dedução da informação em qualquer altura.

Considerando o exemplo exposto neste trabalho (secção 4.4.5), cada relação do

modelo de dados é estendida com um atributo adicional denotando o número do estado (ou etiqueta temporal), sendo a actualização da informação ao longo do tempo realizada através da atribuição do operador "nao"(aplicação ao registo da negação). Por exemplo, as relações apresentadas na figura 4.9 passarão a conter mais um atributo referente à etiqueta temporal da informação:

1. *suppliers(time,#idS, nameS,ratingS)*
2. *p_suppliers(time,#idS,#idP, plafond)*
3. *companies(time,#idC,#idS,#idP,nameC,ratingC)*
4. *products(time,#idP,nameP,Color,Price)*

em que o atributo *time*, corresponde à representação do tempo em que o registo foi criado ou actualizado.

Tendo em consideração esta representação, quando o tuplo ou registo $p(t,a_1,\ a_2,\ \ldots,\ a_n)$ é removido, a base de dados é actualizada com o tuplo $nao\ p(t,a_1,a_2,\ \ldots,\ a_n)$ indicando que este foi removido num momento t. Por exemplo, se considerarmos que a informação contida num momento $t=0$ da base de conhecimento for a apresentada no *programa 1*, poderemos actualizar a sua estrutura no sentido de ***Representar o Conhecimento em termos das extensões do predicado suppliers com evolução temporal da informação*** {

$$(\neg\ suppliers(T,X,Y,Z) \leftarrow\ nao\ suppliers(T,X,Y,Z) \wedge$$
$$nao\ abducible_{suppliers}(T,X,Y,Z)) \wedge$$
$$abducible_{suppliers}(0,1,\ supplier1,20) \wedge$$
$$abducible_{suppliers}(0,1,\ supplier1,40) \wedge$$

$$
\begin{aligned}
&?((abducible_{suppliers}(T,X_1,Y_1,Z_1) \vee abducible_{suppliers}(T,X_2,Y_2,Z_2)) \\
&\wedge \neg(abducible_{suppliers}(T,X_1,Y_1,Z_1) \wedge abducible_{suppliers}(T,X_2,Y_2,Z_2))) \wedge \\
&suppliers(0,2,supplier2,60) \wedge \\
&suppliers(0,3,supplier3,25)
\end{aligned}
$$

}

Se considerarmos que a informação referente à cotação do *supplier* com o identificador "1" passou a ser conhecida num momento *t=1*, a base de conhecimento será actualizada sendo inserido o novo tuplo e aplicado o operador "nao"ao mesmo tuplo com o mesmo identificador num momento anterior. No exemplo seguinte, a informação do "*supplier1*"no momento *t=0* era desconhecido, passando a ser conhecido no momento *t=1*. Neste caso a informação no momento *t=0* é actualizada com o operador "nao"e o registo do momento *t=1* é inserido na base de dados:

{

$$
\begin{aligned}
&(\neg\, suppliers(T,X,Y,Z) \leftarrow\ nao\ suppliers(T,X,Y,Z) \wedge \\
&\qquad\qquad nao\ abducible_{suppliers}(T,X,Y,Z)) \wedge \\
&nao\ abducible_{suppliers}(0,1,\ supplier1,20) \wedge \\
&nao\ abducible_{suppliers}(0,1,\ supplier1,40) \wedge \\
&nao\ ?((abducible_{suppliers}(0,X_1,Y_1,Z_1) \vee abducible_{suppliers}(0,X_2,Y_2,Z_2)) \\
&\wedge \neg(abducible_{suppliers}(0,X_1,Y_1,Z_1) \wedge abducible_{suppliers}(0,X_2,Y_2,Z_2))) \wedge \\
&suppliers(1,1,supplier1,55) \wedge \\
&suppliers(0,2,supplier2,60) \wedge \\
&suppliers(0,3,supplier3,25)
\end{aligned}
$$

}

Em termos de pesquisa de informação, torna-se necessário definir um mecanismo

de inferência no sentido de aceder a um registo de uma base de dados num determinado momento *t*. Este mecanismo geral de pesquisa poderá seguir uma estratégia da seguinte forma:

pesquisa-tuplo(t, a_1*, ...,* a_n*) :- p(t,* a_1*, ...,* a_n*),*

nao p(t, a_1*, ...,* a_n*), !.*

pesquisa-tuplo(t-1, a_1*, ...,* a_n*).*

em que a pesquisa de um tuplo numa entidade num momento "*t*", corresponde ao tuplo encontrado nesse momento com informação positiva ou negativa. Caso contrário a pesquisa irá procurar o tuplo num tempo anterior. Em termos de implementação deste mecanismo através do *PROLOG*, utilizamos o símbolo "*!*"que corresponde a um operador denominado por "*cut*", que permite "parar"o processo recursivo de inferência.

Seguindo esta representação, por exemplo para a relação "suppliers", o mecanismo de pesquisa de informação poderá ser representado da seguinte forma:

pesquisa-suppliers(t, a_1*, ...,* a_n*) :- suppliers(t,* a_1*, ...,* a_n*),*

nao suppliers(t, a_1*, ...,* a_n*), !.*

pesquisa-suppliers(t-1, a_1*, ...,* a_n*).*

Deste modo, este predicado irá percorrer a base de dados temporal iniciando a pesquisa de soluções a partir do tempo t até ao estado inicial, isto é, pesquisa a existência do tuplo "supplier1"durante um intervalo t' até t (em que $t'<t$). A pesquisa é bem sucedida se o registo do "supplier1"for encontrado num momento "t"inferior a "t"e não foi removido (ou actualizado) antes do momento "t".

Por sua vez, também se torna necessário contemplar a pesquisa de informação negativa (ver secção 2.4.3). Este mecanismo neste tipo de pesquisa poderá ser representado da seguinte forma:

pesquisa-Not_tuplo(t, a_1, ..., a_n) :- - p(t, a_1, ...,a_n),

nao - p(t, a_1, ..., a_n), !.

pesquisa-Not_tuplo(t-1, a_1, ..., a_n).

em que o símbolo "-" denota falsidade, tal como o já apresentado neste capítulo. Neste sentido e de modo a implementar esta estratégia, no *anexo C* apresentamos o meta-interpretador lógico para tratar a evolução temporal e a negação explícita.

Em termos de consulta de informação relacionada com um predicado "*p*"deverá ser traduzida como uma pesquisa da questão usando o predicado "*pesquisa-tuplo*", enquanto que qualquer questão a contemplar a negação de um tuplo deverá ser traduzida usando o predicado "*pesquisa-Not_tuplo*". Neste sentido, as regras de inferência serão sistematicamente orientadas na pesquisa pela cláusula mais recente na base de dados enquanto que a questão não for provada ou refutada. Assim, é possível representar a resposta de uma questão colocada por um utilizador fazendo a distinção entre uma proposição que é sempre verdadeira no domínio do discurso e de uma questão que é verdadeira num determinado momento do tempo. Desta forma é contemplado o raciocínio por defeito do tipo "*Se alguém se torna um 'supplier' numa companhia num determinado tempo 't' e não o foi num momento anterior, ele continuará a ser um 'supplier' neste momento*". Assim é assegurada a persistência dos dados ao longo do tempo. Como exemplo, poderemos considerar os seguintes três tipos de questões colocadas à base de conhecimento:

1 – *"Há actualmente fornecedores que fornecem produtos de cor preta?"*.

2 – *"Houve algum fornecedor que forneceu produtos de cor preta?"*.

3 – *Há mais fornecedores que forneceram produtos de cor preta no passado do que actualmente?"*.

A primeira questão poderá ser respondida pela consulta da informação contida na base de conhecimento no actual estado. A segunda pela consulta da informação nos estados anteriores, enquanto que a resposta à terceira questão poderá ser realizada pela avaliação das duas pesquisas sobre o estado actual e os estados anteriores da informação existente na base de dados.

Em suma, apresentamos aqui que o nosso trabalho contempla as questões associadas

ao tempo e à negação explícita em relação à informação armazenada numa base de dados. Com base no exposto nesta secção, as alterações efectuadas a uma base de dados são explicitamente representadas por uma etiqueta temporal nas cláusulas (ou registos) de uma base de conhecimento (ou base de dados). O tempo é representado como uma série de informação descrevendo o *universo do discurso* ao longo dos vários estados de tempo, em que os componentes das relações são definidos por predicados da lógica de primeira ordem aplicado a bases de dados com registo temporal da informação, podendo ser contemplado em cada neurónio do intelecto virtual na resposta a questões colocadas ao longo do tempo. Neste sentido, e tendo em consideração a nova representação da informação (ver secção 4.4.5) é possível estender a linguagem *SQL* (apresentada na secção 4.4.5) aplicada a bases de dados dedutivas contemplando a questão temporal em termos de registo da evolução da informação.

5.3 VirtualInspector

5.3.1 Motivação

Ao longo deste documento foram apresentados vários conceitos teóricos associados ao modelo computacional evolutivo para a criação de intelectos virtuais, que tem como objectivo a pesquisa de soluções para um determinado problema e consequentemente, encontrar a melhor formalização lógico matemática para resolver (ou interpretar) um determinado problema colocado ao sistema.

Contudo, apesar do sucesso da aplicabilidade das técnicas ou abordagens associadas aos paradigmas evolutivo e conexionista, neste trabalho consideramos relevante a criação de um sistema protótipo que permitisse de uma forma visual, ilustrar todo o processo evolutivo na resolução de problemas segundo os formalismos abordados neste trabalho. Na secção seguinte iremos apresentar o *VirtualInspector*, um sistema (ou Framework) que permite visualizar a criação de intelectos virtuais através da materialização do problema segundo a *Programação em Lógica Estendida*, tirando partido do envolvimento dos paradigmas acima referidos.

5.3.2 Implementação

O *VirtualInspector* corresponde a um sistema protótipo com o objectivo de apresentar aos utilizadores uma forma visual para a criação de intelectos virtuais (similar a uma rede neuronal evolutiva), que permite obter as melhores teorias (ou programas lógicos) para a resolução de um problema. Conforme referimos, a base formal deste sistema assenta nos paradigmas simbólico, através da *Programação em Lógica Estendida* para a representação do conhecimento e mecanismo de raciocínio, e os paradigmas evolutivo e conexionista para a componente dinâmica e arquitectural do sistema.

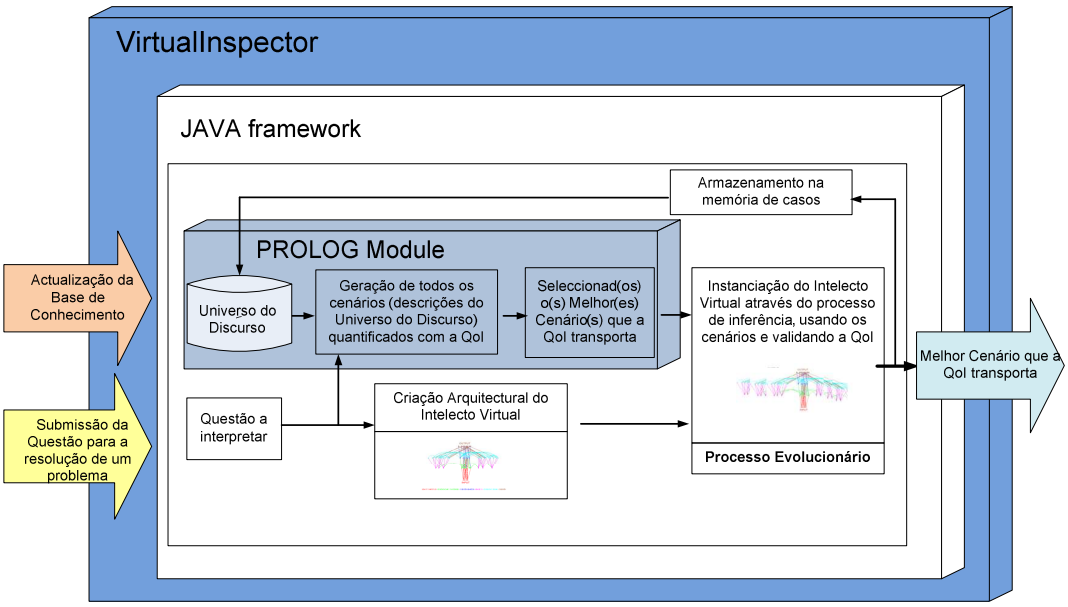


Figura 5.1 -- VirtualInspector – Arquitectura.

O *VirtualInspector* foi desenvolvido na linguagem de programação *JAVA (versão 1.6)* com uma interligação com um módulo *PROLOG* orientado para a representação do conhecimento e mecanismo de inferência. A sua arquitectura é ilustrada na figura 5.1, sendo tipicamente um diagrama de componentes em caixa, representando os principais componentes sobe os quais a implementação foi dividida. A arquitectura assegura o fluxo processual da criação do intelecto virtual apresentado na figura 4.3.

As entradas do sistema correspondem à actualização da informação na base de conhecimento e a submissão de questões a resolver (ou a interpretar), enquanto que

a saída corresponde às soluções (programas lógicos) que maximizam a *QoI* para a interpretação da questão em observação.

A framework *JAVA* foi utilizada pela sua potencialidade em poder ser disponibilizada em várias plataformas aplicacionais, assim como pelas suas vantagens em termos de visualização em três dimensões de gráficos dinâmicos, permitindo disponibilizar uma forma visual e interativa com o sistema. Por outro lado, o *JAVA* foi utilizado na implementação das metodologias associadas aos paradigmas evolutivo e conexionista com a interligação do paradigma simbólico implementado em *PROLOG*[1].

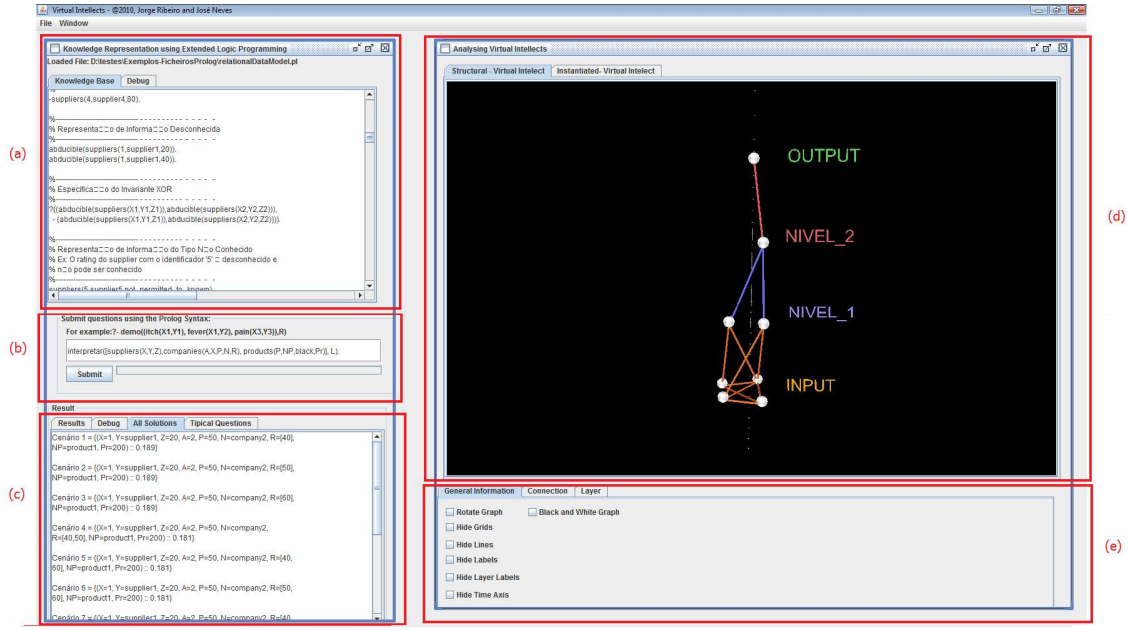


Figura 5.2 - VirtualInspector – Interface Principal.

Assim que a aplicação é carregada, é disponibilizado ao utilizador uma interface simples e intuitiva para a utilização do sistema. O sistema está dividido em cinco áreas: uma referente ao carregamento da informação inicial para a base de conhecimento (figura 5.2 (a)), outra referente à parte de submissão das questões (figura 5.2 (b)), outra referente à disponibilização das soluções e visualização do "debug"do processo de inferência (figura 5.2 (c)), outra referente à visualização dos intelectos estruturais e de instanciação (figura 5.2 (d)) e outra referente à disponibilização de

[1]usando o Swi-Prolog [Swi2010] versão 5.10 e Sicstus-Prolog [Sic2010]

funcionalidades para a visualização a três dimensões dos intelectos virtuais (figura 5.2 (e)).

Em termos de utilização, o utilizador para carregar uma base conhecimento deverá seleccionar a opção "File", sendo-lhe disponibilizada a possibilidade de carregar a informação a partir de ficheiro (*PROLOG*) ou através de uma ligação a uma base de dados. Depois de carregada a informação, o sistema está preparado para se invocar uma questão para interpretação.

A figura 5.3 apresenta com mais detalhe o carregamento da informação ilustrada na figura 5.2 (a). Na figura 5.3 (a) apresenta-se a representação do conhecimento do exemplo apresentado na secção 4.4.5, a qual é baseada na *Programação em Lógica Estendida* utilizada neste trabalho, enquanto que a figura 5.3 (b) apresenta o exemplo da especificação da questão (25) a submeter ao sistema.

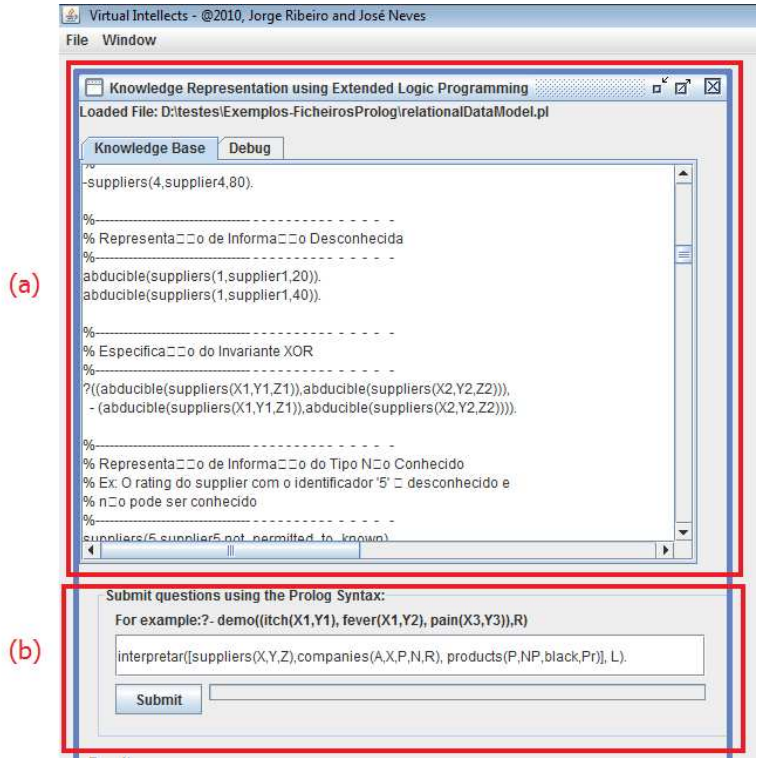


Figura 5.3 - VirtualInspector – Carregamento da Informação e submissão de questões.

Tal como apresentamos no início da secção 4.4.1, depois de submetida uma questão, o *VirtualInspector* irá criar dois intelectos: um arquétipo estrutural referente à repre-

sentação das entidades envolvidas no processo de inferência para a interpretação da questão, e outro referente ao processo de instanciação do arquétipo estrutural para a obtenção das soluções. A figura 5.2 (d) ilustra o intelecto virtual (ou arquétipo) estrutural das entidades (ou predicados) envolvidos na interpretação da questão (25). Esta representação em grafo corresponde ao intelecto ilustrado na figura 4.17 [2] referente ao intelecto estrutural criado para a interpretação da questão (25). Tal como o intelecto da figura 4.17, o arquétipo ilustrado na figura 5.2 (d) contém quatro níveis. O primeiro (*input*) contém as entidades representadas no *universo do discurso* (neste caso quatro) e as suas relações, sendo activados apenas três neurónios envolvidos na interpretação da questão (25), nomeadamente "suppliers", "companies"e "products". No nível um, os dois neurónios correspondem à relação entre essas entidades, nomeadamente "companies"e "products"e, "suppliers"e "companies". Neste caso, no nível dois o único neurónio realiza a união entre os neurónios do nível um.

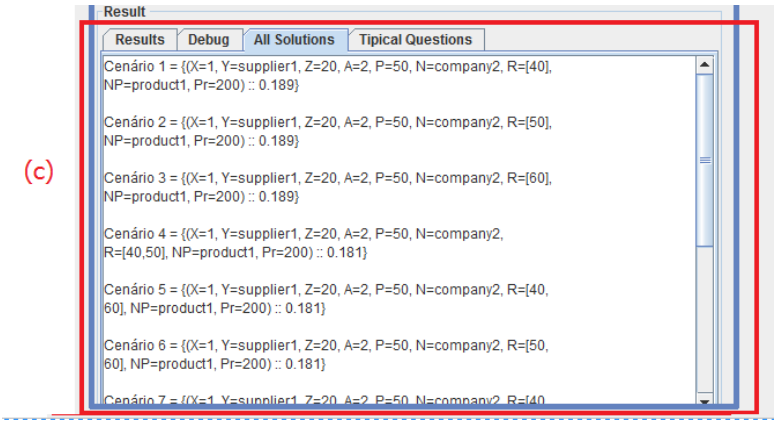


Figura 5.4 - VirtualInspector – Apresentação das soluções para a interpretação do problema.

Como saída é criado um neurónio (*output*) que interliga com o neurónio da camada inferior (nível dois), permitindo definir a estrutura de um intelecto virtual para ser utilizado no processo de inferência para a instanciação das variáveis[3] (ou termos) dos predicados envolvidos na questão submetida ao sistema (figura 5.2 (b)), neste caso

[2] ilustração a verde, uma vez que a azul corresponde à actualização do intelecto criado para a interpretação da questão (24,) o qual é ilustrado na figura 4.11.

[3] instanciação dos termos dos predicados (ou variáveis) corresponde à atribuição de valores para esse predicado. Por exemplo, a instanciação dos termos do predicado *suppliers(X,Y,Z)*, corresponde à atribuição de valores para X, Y e Z. Se considerarmos que poderemos ter extensões ao predicado (i.e. valores desconhecidos), poderemos atribuir vários valores em combinação para X, Y e Z.

obter o valor das variáveis para os predicados "suppliers", "companies"e "products". Na figura 5.4 ilustra-se a apresentação das soluções para a interpretação da questão submetida ao sistema (25).

Depois de submetida a questão e criado o intelecto arquitectural (figura 5.2 (d)) que define a abstracção das entidades envolvidas na interpretação da questão, este intelecto é instanciado com os predicados e as suas extensões, no sentido de executar o processo de inferência e atribuir significado às variáveis ou termos dos predicados, permitindo obter as soluções (figura 5.4) que interpretam a questão em observação (figura 5.3 (b).

Durante o processo de inferência os neurónios do arquétipo estrutural são instanciados. Em cada um deles é executada a questão que se apresenta como entrada, sendo calculada a *QoI* sobre as extensões dos predicados que modelam o contexto do *universo do discurso* existente nesse neurónio. Por exemplo, na camada de entrada do arquétipo estrutural, a questão (ou lista de problemas) que se coloca é interpretar a questão (25). Tendo como orientação o arquétipo ilustrado na figura 4.17, nos dois neurónios do nível um, a questão colocada a um neurónio é obter a instanciação dos termos dos predicados "companies"e "products", e no outro neurónio dos predicados "suppliers"e "companies". Por sua vez, a questão a interpretar no neurónio do nível dois é instanciar os termos dos predicados "suppliers", "companies"e "products", os quais, neste exemplo, foram "transportados"dos neurónios da camada inferior.

No decorrer do processo de inferência, os termos dos predicados são instanciados em cada neurónio. Contudo, tal como apresentamos na secção 4.2.3, poderão existir extensões dos predicados que originem que a instanciação de um termo de um predicado possa assumir vários valores. Por este facto, para responder á questão colocada no *input* do sistema, irão ser efectuados vários "caminhos"pelo intelecto arquitectural consoante o tipo de extensões existentes em cada predicado envolvido na interpretação da questão. Por sua vez, em cada neurónio será avaliada a *qualidade da informação* do programa lógico correspondente ao contexto que modela o *universo do discurso* para responder á questão submetida a esse neurónio. Se a medida da *QoI* estiver abaixo de um valor predefinido, o processo de instanciação desse caminho terminará, pois apesar de ainda se poder continuar o processo de instanciação noutros neurónios, a qualidade que esse caminho transporta não oferece uma qualidade desejável e, por isso, o processo deste caminho em particular termina o processo de instancação.

A figura 5.5 (a) ilustra a representação gráfica da instanciação do intelecto da

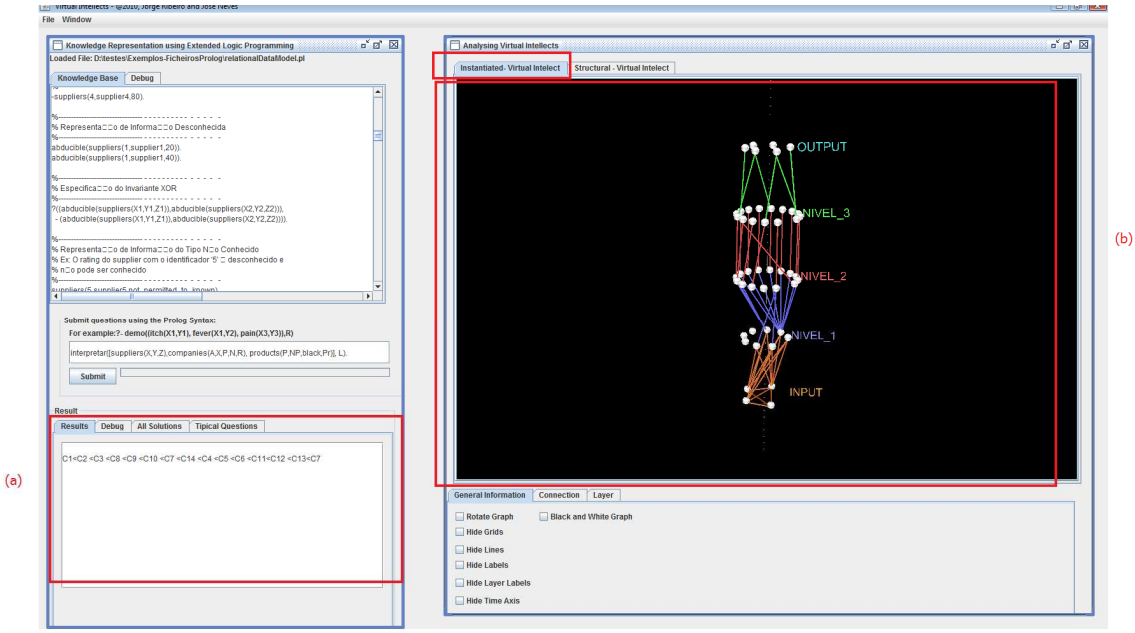


Figura 5.5 - VirtualInspector – Instanciação do arquétipo estrutural para a interpretação do problema em observação.

figura 5.2 (d). A camada de entrada corresponde à questão colocada ao sistema (25) e, em cada camada intermédia da rede são criados neurónios consoante as extensões dos predicados envolvidos na interpretação da questão. Por este facto, em consonância com o intelecto virtual arquitectural ilustrado na figura 5.2 (d), são encontradas várias soluções para a questão em observação, nomeadamente catorze (descritas na secção 5.2.1 e detalhadas no anexo A). No entanto, foram seleccionadas apenas as melhores seis soluções (ou cenários), uma vez que, por um lado, interpretam a questão colocada ao sistema e por outro, maximizam a *QoI*, uma vez que o limite mínimo foi definido como "0.185". Essas soluções são apresentadas na figura 5.5 (a) através da criação dos neurónios da camada de *output*, correspondendo à instanciação dos termos dos predicados que envolvem a questão em observação, nomeadamente C_1, C_2, C_3, C_8, C_9 *e* C_{10}, os quais foram apresentados na secção 5.2.1.

Por outro lado, no sentido de auxiliar a visualização a três dimensões, este protótipo disponibiliza um conjunto de funcionalidades (figura 5.2 (e)) que permite por exemplo organizar os neurónios de cada camada (ex. forma circular, esférica), ampliar a visualização para "consultar"a estrutura do neurónio (tal como ilustrado na figura 4.5), activar ou desactivar a identificação das camadas e dos neurónios, entre outras fun-

cionalidades básicas existentes em termos de visualização de grafos a três dimensões.

Como exemplo da aplicabilidade dos conceitos abordados neste trabalho, considerando as duas questões colocadas, nomeadamente a (27) e (28), o sistema irá proceder à execução do processo de inferência e, consequentemente seleccionar as melhores teorias lógico matemáticas. Tal como apresentamos na secção 4.4.6 o sistema irá proceder à sua componente evolutiva, partindo das melhores teorias obtidas nesta primeira fase para a resolução do problema (interpretação das duas questões do exemplo) e descobrir novos relacionamentos.

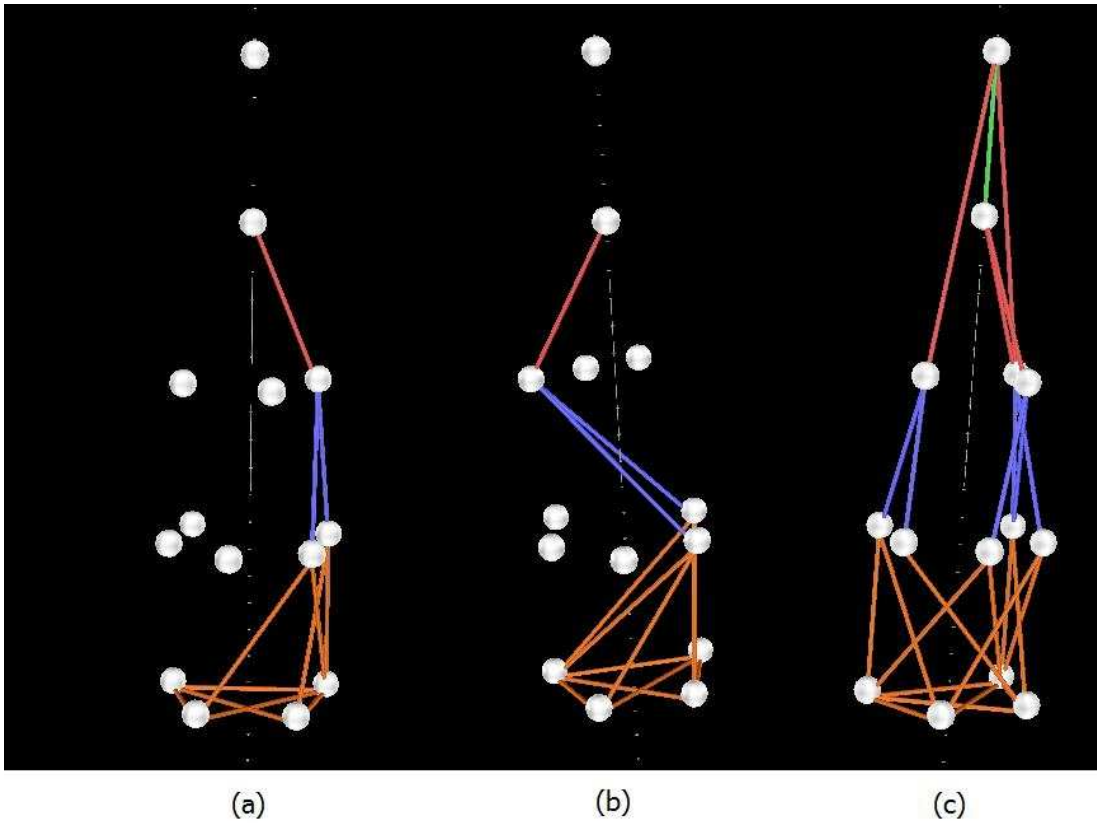


Figura 5.6 - VirtualInspector – Exemplo da evolução do itelecto arquitectural.

No seguimento do apresentado na secção 4.4.6, a figura 5.6 ilustra a evolução do intelecto e a consequente aprendizagem de novo conhecimento, sendo a sua estrutura autoorganizada. Na figura 5.6 ilustra-se cada um dos intelectos para cada uma das questões (27), (28) e (26), através dos intelectos (a), (b) e (c) respectivamente. Por outro lado, na figura 5.7 é ilustrado o arquétipo da sua instanciação aquando da invocação da questão (28).

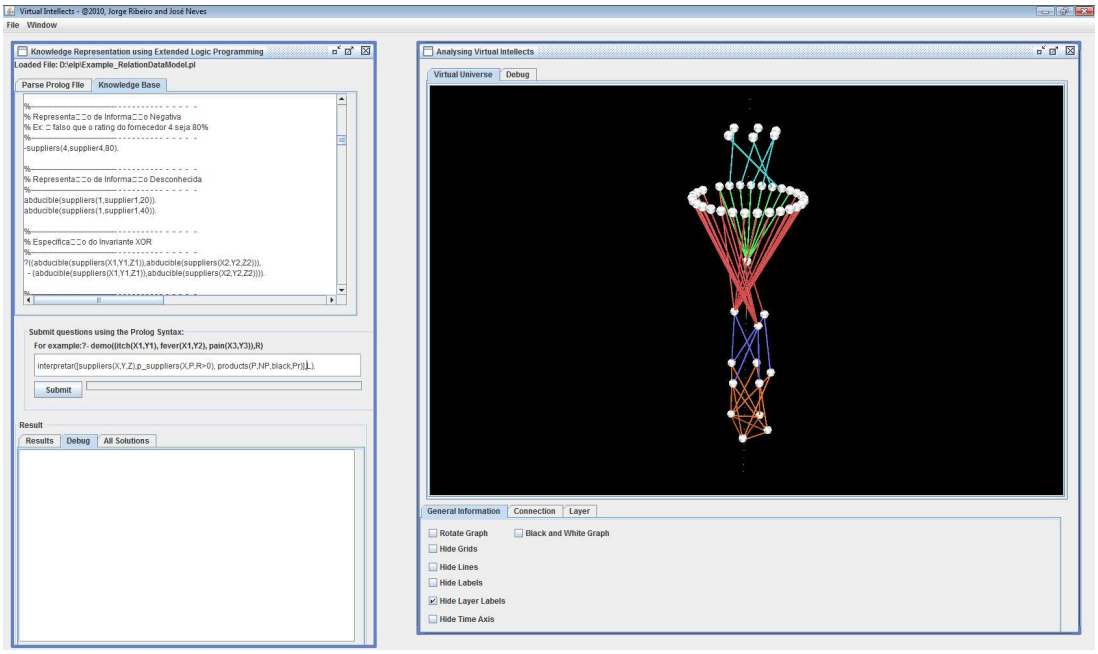


Figura 5.7 - VirtualInspector – Instanciação do intelecto arquitectural.

Através deste sistema, o utilizador tem a possibilidade de "navegar"sobre o processo de inferência e avaliar as melhores teorias que modelam o *universo do discurso* em cada entidade. No final da instanciação do arquétipo, na camada anterior à de saída, são constituidos todos os cenários (Anexo A) referentes à interpretação da questão. Na camada de saída irão imergir apenas aquelas que maximizam o *universo do discurso*.

Por outro lado, com base nesta nova reorganização, a resposta a uma nova questão, poderá seguir percursos diferentes, sendo encontradas novas soluções capazes de responder às mesmas ou a novas questões, em que o processo de avaliação da evolução é efectuado através da medida da *qualidade da informação* analisada em cada entidade do arquétipo. Por fim, as entidades da camada de saída apresentam as melhores teorias lógico matemáticas que maximizam o *universo do discurso*.

5.3.3 Acompanhamento do processo de inferência

Em termos de utilização, o *PROLOG* funciona como uma "caixa negra"onde são invocados predicados, sendo devolvida a instanciação das variáveis do predicado invocado. Apesar de se puder utilizar um predicado especifico para efectuar o "*trace*"da

execução dos predicados, o *PROLOG* não permite a partir de uma linguagem de programação externa aceder ao processo interno de execução dos predicados. Por este facto, torna-se dificil acompanhar o processo de execução dos predicados a partir do exterior da plataforma do *PROLOG*. No sentido de colmatar esta limitação, desenvolvemos uma "*script*"ou conjunto de instruções, que após inseridas na base de conhecimento, todo o processo de execução do *PROLOG* é armazenado internamente numa lista, permitindo desta forma aceder à informação de todo o "*trace*"de execução a partir de um programa externo. A lista de instruções a seguinte:

```
:-dynamic traceList/1.

prolog_trace_interception(Port, Frame, _PC, continue) :-
        prolog_frame_attribute(Frame, goal, Goal),
        prolog_frame_attribute(Frame, level, Level),
        insertTraceList((Port,Goal,Frame)),
        recordz(trace, trace(Port, Level, Goal)).
prolog_trace_interception(Port, Frame, _PC, continue).
insertTraceList(H):-bagof(ActualList,traceList(ActualList),L),!,
        retract(traceList(_)),
        appendLists(L,H,NewList),
        assert(traceList(NewList)),
        traceList(L1).
insertTraceList(H):-appendLists([],H,NewList),assert(traceList(NewList)).
removeTraceList:-retract(traceList(_)).
```

appendLists([], List, List).

appendLists([First | Rest1], List, [First | Rest2]) :- appendLists(Rest1, List, Rest2).

Após o carregamento da informação para a base de conhecimento do protótipo, o sistema acrescentará este conjunto de predicados. Desta forma, sempre que se procede à invocação de um predicado dentro do *PROLOG* bastará preceder o predicado a invocar pela instrução "trace". Por exemplo, o protótipo ao receber a questão (25), internamente submete ao *PROLOG* a instrução "***trace****,interpretar([suppliers(A,B,C), companies(A,D,P,W,Z), products(P,N,black,X)],L).*". Desta forma, todas as instruções do processo de inferência são armazenadas na lista "*traceList*", podendo sr acessível através do *JAVA*. Este código, torna-se extremamente útil no sentido de obter a informação de todos os predicados ennvolvidos na interpretação de uma questão. Por exemplo, tendo em consideração o exemplo da secção 4.4.5, se pretendermos obter toda a informação sobre "*quais os fornedores que forneceram e fornecem produtos de cor preta"* e se submetermos a questão "*interpretar([suppliers(A,B,C), products(D,N,black,P)],[A,B,C,D,N],L).*", o sistema terá de encontar todos os caminhos entre "suppliers"e "products". Conforme a estrutura apresentada na figura 4.9, existem três caminhos possíveis entre estes dois predicados. No entanto, existem predicados (ex: "p_suppliers"e "companies") que poderão conter informação desconhecida e que irão ser necessários conhecer para a criação do intelecto. Assim, apesar de serem indicados na questão os predicados "suppliers"e "products", através da inclusão dos predicados (ou instruções) acima descritos, permitirá ao protótipo que sejam identificados outras entidades envolvidas na interpretação da questão, devendo ser tidas em consideração na criação do intelecto arquitectural a fim de assegurar o processo de execução da instanciação do contexto associado a cada neurónio desse intelecto.

5.4 Conclusão

O objectivo do Virtual Inspector correspondeu ao desenvolvimento de um sistema protótipo capaz por um lado, de lidar com a especificação associada à representação do conhecimento usando a *Programação em Lógica Estendida* em ambientes contemplados com a presenąa de informação incompleta e, por outro, a utilização dos

paradigmas evolutivo e conexionista.

Durante o período de realização desta dissertação de doutoramento sentiu-se a necessidade de ilustrar graficamente o processo evolutivo subjacente a este trabalho. O protótipo foi desenvolvido numa primeira fase tendo como orientação apresentar uma forma simples de visualização de todo o processo descrito no capitulo quatro. Contudo, apesar de conseguirmos estes objectivos e de terem sido desenvolvidas várias funcionalidades que enriqueceram o sistema, o protótipo poderá ser melhorado e aperfeiçoado com novas funcionalidades, no sentido de aumentar a potencialidade da sua utilização e a sua aplicabilidade noutros casos de estudo, em especial em sistemas distribudos ou mesmo em sistemas multi-agente.

Por outro lado, com este projecto, abrimos caminho para uma promissora linha de investigação no que se refere à interligação dos três paradigmas apresentados, em especial na exploração das suas potencialidades no se refere à criação de sistemas inteligentes virtuais, passiveis de serem utilizados em processos de optimização.

"Let every man be respected as an individual and no man be idolized"
[Einstein in His own words, Anne Rooney,2006]

Capítulo 6

Conclusões

6.1 Síntese da Dissertação

O objectivo principal deste trabalho centrou-se no desenvolvimento de comunidades (ou sistemas) virtuais evolutivos que ajudem na optimização de cenários reais. Com esta tese de doutoramento tentou-se contribuir para o aumento do conhecimento científico na área da *Neuro-evolução* [Yao1999] [Abra2004] [Nit2008] tirando partido das vantagens associadas ao paradigma simbólico (materializado aqui através da *Programação em Lógica Estendida* [Nev1984] [NM$^+$1997]) como mecanismos apropriado para a representação do conhecimento e mecanismo de raciocínio, assim como dos paradigmas evolutivo e conexionista.

Pretendíamos desenhar e implementar um sistema híbrido orientado para a resolução de problemas, i.e. orientamo-nos em tirar partido das vantagens associadas ao paradigma simbólico, pela sua característica e potencialidade para representar o conhecimento, assim como em disponibilizar mecanismos de raciocínio. O paradigma evolutivo foi utilizado pelas suas reconhecidas vantagens para a geração de soluções. O paradigma conexionista, em especial as *Redes Neuronais Evolutivas* foi utilizado pela sua capacidade de geração de topologias de rede.

Para descrever o trabalho desenvolvido, na primeira parte da dissertação abordamos de forma superficial os conceito teóricos associados a este trabalho, nomeadamente a *Lógica*, *Programação em Lógica*, sistemas evolucionários, sistemas conexionistas e a integração entre estes três paradigmas. Na segunda parte centramo-nos na especificação do sistema computacional através da apresentação de um exemplo,

assim como a apresentação de uma framework que implementa os conceitos abordados.

6.2 Tese

Conforme apresentamos, os objectivos específicos deste trabalho de doutoramento foram descritos na secção 1.5.1, sobre os quais passamos a discutir o que foi conseguido.

1. Pretendíamos criar um modelo formal para raciocinar com informação incompleta e quantificar todas as soluções usando a medida da qualidade da informação associada aos programas lógicos ou teorias. Para atingir este objectivo adoptamos uma representação intimamente ligada à sintaxe do *PROLOG* [CM1981] [Brat1990] [SS1994]. Adoptamos uma representação de alto nível utilizando a *Programação em Lógica Estendida* [Nev1984] [NM$^+$1997]. Este tipo de representação, não só permite especificar a representação do conhecimento e desenvolver mecanismos de raciocínio, mas também em permitir a especificação de teorias lógico matemáticas de formas não fixas, permitindo a representação tanto de cenários reais, com particularidades específicas, como de informação incompleta, lógica *multivalor*, cenários com dependências funcionais, etc.

2. Pretendíamos desenvolver uma *framework* baseada no *Paradigma da Computação Evolucionária* (neste caso a *Programação Genética*), de modo a prever a evolução dos programas, empregando metodologias para a resolução de problemas que beneficiam dos *abducibles* e maximizam a qualidade da informação associada ao *universo do discurso*. Para realizar este objectivo, utilizamos as características do esquemático arquitectural e processo de aprendizagem usado no paradigma conexionista em especial das *Redes Neuronais Artificiais Evolutivas* [Yao1999] [Abra2004] [Nit2008] [FDM2008].

3. Pretendíamos criar uma extensão à *framework* para ser usada em várias áreas, nomeadamente, médica, ambientes industriais e ambientes de tomada à decisão. Para atingir este fim, aplicamos o modelo computacional [RM$^+$2010bc] a três grandes áreas de aplicação, nomeadamente, em processos de tomada à decisão nos sistemas em

grupo [NS+2010], processsos de produção [RN+2009], saúde [RM+2010a] e bases de dados [RM+2010b].

4. Pretendíamos criar um sistema com acções visíveis para os seus utilizadores, no sentido de tornar o sistema visível e interactivo, apresentando uma interface simples e amigável. Para alcançar este objectivo implementamos um sistema protótipo apresentado na secção 5.3.

Em relação à fundamentação da *Lógica Matemática* e da utilização da *Lógica* como teoria do raciocínio, baseamo-nos nos conceitos da *Programação em Lógica Estendida*. Em relação à comparação entre a teoria dos modelos e teoria da prova, nesta nossa abordagem, um modelo deve ser entendido como uma composição de predicados que representam objectos e as suas relações estabelecidas entre eles, no sentido de modelar o *universo do discurso*.

A formalização do arquétipo para a resolução de problemas e optimização dos melhores cenários para a interpretação ou resolução de um problema é similar a uma rede neuronal a qual denota uma abstracção para a resolução de um problema. Isto é, é algo que é geral, mas que é passível de ser concretizado através de um processo de instanciação a uma situação particular. Neste sentido, em termos de pesquisa de soluções, pretendeu-se procurar os parâmetros das extensões dos predicados expressos através da *Programação em Lógica Estendida*. Neste ambiente evolucionário, uma solução óptima corresponde à melhor teoria lógico matemática (programa lógico ou teoria), que modela o *universo do discurso* e maximiza o factor da *Qualidade da Informação* associada a essas teorias. Finalmente, a medida e obtenção de valores para possíveis soluções durante o processo evolucionário é realizado através do mecanismo de prova de teoremas e fusão de programas [TF1995].

Neste trabalho utilizamos a medida da *Qualidade da Informação (QoI)* [NM+2007] para a quantificação dos programas lógicos associados à modelação do *universo do discurso*. Conforme apresentamos existem várias técnicas associadas à avaliação da qualidade do conhecimento. No entanto, seleccionamos a medida da *QoI*, uma vez que apresenta uma abordagem clara e possível em termos do seu processo de quantificação, que é feito atendendo à estrutura base do próprio registo das relações dos predicados (ou das tabelas das bases de dados), quando contempladas com informação incompleta. Neste sentido a utilização desta medida apresenta como resultado um

grau de confiança dado em termos de um valor de verdade que podemos associar à combinação de hipóteses para a resolução de problemas, i.e., através das associações possíveis entre os diferentes valores dos argumentos dos predicados ou registos das relações de uma base de dados.

Por outro lado, a medida da *QoI* pode ser aplicada a situações em que contemple informação incompleta em vários atributos dos registos de uma relação de uma base de dados. Este suposto problema de generalização da informação incompleta aplicada a um ou a vários atributos, é resolvida escolhendo os melhores registos de uma relação e só depois escolhendo o melhor cenário para a extensão da relação.

Neste sentido, tendo em consideração as questões e hipótese de trabalho enunciadas na secção 1.2 demonstramos que é possvel aliar as vantagens dos paradigmas evolutivo, conexionista e simbólico, permitindo criar sistemas inteligentes com capacidades adicionais de aprendizagem para a resolução de problemas.

6.3 Contribuições, Originalidades e Aplicabilidade

O trabalho desenvolvido procura providenciar respostas às questões levantadas anteriormente e outras relacionadas (secção 1.2), não tendo a pretensão de fornecer soluções definitivas, mas sim contribuir com uma parcela de conhecimento que possibilite algum avanço científico nas áreas em estudo. As principais contribuições e originalidades do presente trabalho são a seguir enunciadas:

- Definição de um terreno comum para situar os mecanismos baseados nos paradigmas: simbólico, evolutivo e conexionista;
- O uso de ferramentas formais (*Lógica*) para descrever o comportamento racional das entidades presentes no processo de resolução de problemas, envolvendo *Computação Evolucionária*;
- Definição e especificação de um sistema formal para representação de informação incompleta [NM+1997], nomeadamente no que respeita à especificação de nulos não permitidos e à elaboração do meta-interpretador [RN+2009] [RM+2010a] [RM+2010b];

- Formalização de um sistema evolutivo capaz de gerar novos cenários na resolução de problemas e desta forma seleccionar as melhores teorias lógico matemáticas com melhor qualidade do que as iniciais, em que as soluções candidatas são avaliadas por um único critério: a medida da *qualidade da informação* [AN+2006] [NM+2007] [RM+2010c] associada a cada teoria lógica ou programa que representa o *universo do discurso*;

Tirando partido do paradigma simbólico como esquema de representação do conhecimento e mecanismo de raciocínio, do paradigma evolutivo pela sua capacidade para a geração de novas soluções e pelo paradigma conexionista pelos seus esquemas arquitecturais de rede, o desenvolvimento de um protótipo de um intelecto virtual capaz de gerar cada vez melhores cenários para a resolução de problemas permitindo a obtenção das teorias lógico matemáticas que melhor modelam o *universo do discurso*;

A contribuição para a área das *Redes Neuronais Artificiais* centrou-se na elaboração de três estudos, dois [FR+2010a] [FR+2010b] em especial na apresentação de novos métodos e abordagens quando aplicadas a problemas de classificação, enquanto que o terceiro [NS+2011] centrou-se na extracção de informação a partir do sistema evolutivo, em particular em ambientes em que a informação é incompleta, no sentido de "preprocessar" a informaç ao incompleta para ser utilizada numa *Rede Neuronal Artificial* em particular quando aplicada a problemas de classificação, podendo contudo, ser utilizado para outros objectivos;

Por outro lado, a originalidade deste trabalho de doutoramento centrou-se na convergência das vantagens associadas aos paradigmas simbólico, evolutivo e conexionista, no sentido de criar sistemas dinâmicos para a resolução de problemas. Ao longo dos anos tem sido desenvolvido estudos centrados na interligação dos paradigmas simbólico e evolutivo, entre o paradigma simbólico e conexionista e entre o paradigma evolutivo e o paradigma conexionista. Contudo, tendo em consideração a literatura actual, a conjugação destes três paradigmas tem sido alvo de pouca atenção e exploração. Neste sentido, este trabalho tenta contribuir para este objectivo permitindo explorar a criatividade da conjugação destas técnicas associadas à *Inteligência Artificial*.

Em termos de aplicabilidade, este trabalho de doutoramento utiliza os três paradigmas acima referidos, os quais ao longo dos anos foram reconhecidos e aplicados com sucesso em várias áreas do conhecimento. A *Representação do Conhecimento* e o

mecanismo de raciocínio com que a criação do sistema formal desta tese de doutoramento se baseia, utiliza a *Programação em Lógica Estendida* [Nev1984] e o conceito da *Qualidade da Informação* [AN+2006] [NM+2007] como formalismo de quantificação da informação associada às teorias lógico matemáticas. Estes conceitos foram aplicados com sucesso em várias áreas do conhecimento, como por exemplo, [AN+2006] [MA+2006] [LC+2008] [MM+2009] [RN+2009] [LN+2009] [MA+2010] [NA+2010]. Contudo, os estudos associados e desenvolvidos neste trabalho de doutoramento (por exemplo [RM+2010c]) foram aplicados em áreas de processamento industrial [RN+2009], tomada à decisão [NS+2010], sistemas de bases de dados [RM+2010ª] e unidades de saúde [RM+2010b].

Adicionalmente uma parte especifica deste trabalho permitiu ser aplicado na classificação de problemas usando as *Redes Neuronais Artificiais* [FR+2010ª] [FR+2010b], em particular a típicos conjuntos de dados, nomeadamente, [Nea1998] e [NA2007]. Como contribuição complementar para a alimentação de informação incompleta quando utilizada nas *Redes Neuronais Artificiais*, o estudo [NS+2011] permite classificar as características dos pavimentos das estradas usando as redes, previamente "alimentadas" com informação extraída do sistema evolutivo criado. Desta forma, a rede é alimentada com informação completa[1] quantificada em cenários em que a informação é tipicamente incompleta, imprecisa ou omissa.

6.4 Limitações e Trabalho Futuro

Neste trabalho usamos conjuntamente três paradigmas computacionais da Inteligência Artificial, como o simbólico, o evolutivo e o conexionista. Foi utilizado o paradigma evolutivo associado ao paradigma simbólico para a representação do conhecimento e de mecanismos de raciocínio em ambientes em que a informação é incompleta, omissa ou sensível ao erro.

Consideramos que a fusão destes paradigmas, em que a sua aplicabilidade tem vindo a ser reconhecida com o sucesso na resolução de casos complexos, permite-nos abordar a futura criação de intelectos virtuais capazes de simular em certa medida

[1] Informação completa: Em oposição ao tipo de informação incompleta, isto é. Corresponde ao cenário em que toda ou parte da informação não é ambígua, não é desconhecida, não é imprecisa ou não é sensível ao erro.

as formas de representação e de raciocínio humano. Contudo, temos consciência que muito caminho ainda há para fazer. Uma das limitações deste trabalho e em particular do protótipo desenvolvido corresponde ao processo de entrada de informação para o sistema, quer em termos de interface de desenvolvimento (ou interligação com dispositivos de captura de informação, como por exemplo sensores de aquisição de dados), quer me termos da sua conversão para a representação do conhecimento utilizada. Esta limitação será um dos trabalhos futuros em termos de melhoria do prótipo desenvolvido.

Um outro interessante tópico de investigação será estudar e aplicar diferentes topologias de aplicação dos operadores genéticos, assim como avaliar os resultados obtidos. Um outro estudo seria a análise da sensibilidade da informação submetida à rede evolucionária, assim como a exploração das Redes Neuronais Evolucionárias.

Neste trabalho pretendeu-se seguir, explorar e potenciar uma linha de investigação na área da modelação de sistemas inteligentes tirando partido dos paradigmas simbólico, evolutivo e conexionista. Para isso utilizamos exemplos simples que no nosso entender se caracterizam pela sua simplicidade em demonstrar a potencialidade e aplicabilidade deste estudo. Apesar de termos consciência que a representação de informação em cenários reais é dificil e complexa, pretendemos no futuro aplicar o conhecimento adquirido em ambientes com caracteristicas próprias como por exemplo, sistemas multi-agente, sistemas baseados em casos, sistemas de aprendizagem automtica, entre outros. Desta forma iremos estudar e explorar as potencialidades da criação de sistemas de raciocínio evolutivos em outras áreas de aplicação.

Apêndice

Capítulo 7

Anexos

A Conjunto de soluções ou cenários para as questões sobe observação

Anexo A - Conjunto de soluções ou cenários para as questões (24), (25) e (26) sobe observação

Neste capítulo dos anexos iremos apresentar todo o conjunto de soluções ou cenários para as questões (1), (2) e (3) sobe observação apresentadas ao longo deste documento. No sentido de exemplificar a aplicabilidade e potencialidade deste trabalho, foi escolhido um exemplo simples de um modelo relacional contendo informação incompleta nos seus registos. As três questões colocadas ao sistema foram:

(24) ***"Quais os fornecedores que no presente estão em condições de fornecer produtos de cor preta"***

(25) ***"Quais os fornecedores que no passado forneceram produtos de cor preta"***

(26) ***"Quais os fornecedores que no passado forneceram produtos de cor preta e que no presente também estão em condições de o fazer"***

Para cada uma destas questões a interpretar (ou a resolver) iremos de seguida apresentar todos os cenários obtidos através do mecanismo de inferência representando-os através da Programação em Lógica Estendida. Através do conceito da Qualidade da Informação iremos quantificar esses cenários e apresentar as soluções quantificadas seguindo a semântica associada ao modelo relacional de base de dados.

1. Questão (24) a interpretar: ***"Quais os fornecedores que no presente estão em condições de fornecer produtos de cor preta"***

Para a interpretação desta questão, os predicados envolvidos são: "suppliers", "p_suppliers" e "products". O conjunto de soluções ou cenários que interpretam ou resolvem a questão, pode ser representado através da Programação em Lógica Estendida (definição 3) ou através da nova representação através das relações quantificadas pela *QoI* (secção 4.4.3):

1.1 Representação dos cenários usando a Programação em Lógica Estendida:

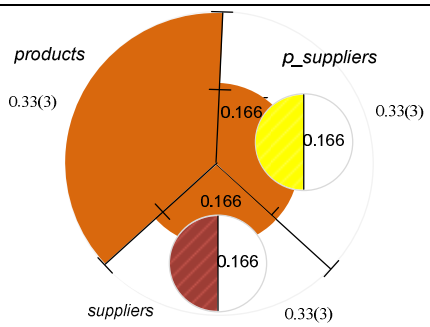
Cenário 1	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 2	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge$

	$(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 3	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1,\ supplier1,\ 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 4	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1,\ supplier1,\ 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$

Figura A.1 – Representação em PLE das soluções ou cenários que interpretam a questão (24).

1.2 Quantificação da Qualidade da Informação associada a cada um dos cenários para a interpretação da questão (24):

Considerando o grau de relevância (ou importância) dos predicados da seguinte forma: 0.25, 0.35 e 0.40 para "suppliers", "p_suppliers" e "products", é possível (face ao exposto na secção 4.3.2) quantificar e representar a quantificação do conhecimento associado a cada um dos cenários, em que $QoI_{(QN,M)}$, $N \in \{1,2,3\}$ correspondente ao número das questões e $M \in \{1,..R\}$, em que R é o número máximo de cenários possíveis de interpretar a questão em observação:

Cenário	Representação da QoI	Valor de Verdade (ou QoI)
Cenário 1		$QoI_{(Q1,1)} = 0,33 * 0,4 + 0,166 * 0,35 + 0,166* 0,25= 0.2316$

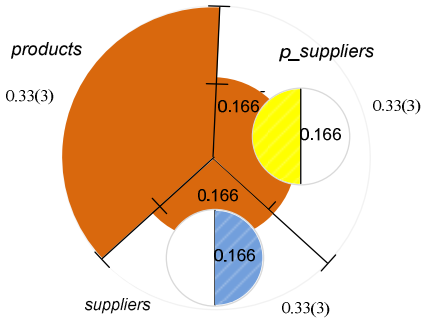
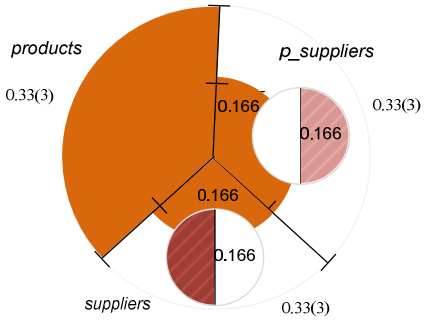
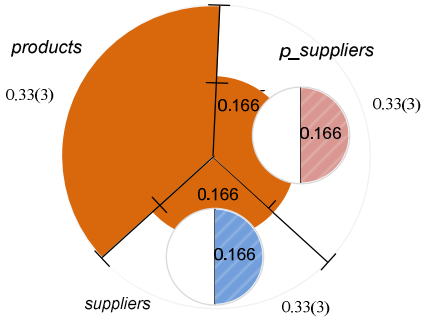
Cenário 2		$QoI_{(Q1,2)}$ = 0,33 * 0,4 + 0,166 * 0,35 + 0,166* 0,25= 0.2316
Cenário 3		$QoI_{(Q1,3)}$ = 0,33 * 0,4 + 0,166 * 0,35 + 0,166* 0,25= 0.2316
Cenário 4		$QoI_{(Q1,4)}$= 0,33 * 0,4 + 0,166 * 0,35 + 0,166* 0,25= 0.2316

Figura A.2 – Representação e Quantificação da *QoI* das soluções ou cenários que interpretam a questão (24).

Nota: Neste caso, para a questão a interpretar os registos associados aos predicados envolvidos, não contemplam situações hipotéticas (*abducibles* ou *excepções*) sem o invariante *XOR*. Por este facto, o valor do cálculo da *QoI* é similar para cada cenário. No entanto, caso por exemplo, no predicado "suppliers" não houvesse especificação do invariante *XOR*, o cálculo da *QoI* seria diferente. Esta situação é mais visível na questão que iremos apresentar mais adiante.

1.3 Representação gráfica da *QoI* sobe os cenários que interpretam a questão em observação

Tendo em consideração o exposto na secção anterior, o gráfico com a *QoI* associada a todos os cenários envolvidos na interpretação ou resolução da questão é o seguinte:

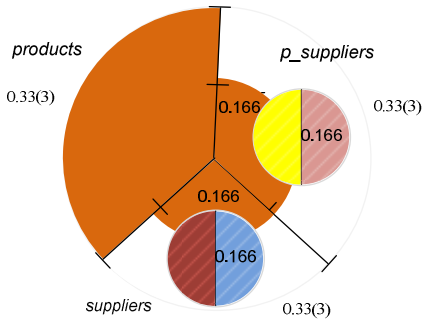


Figura A.3 – Gráfico da *QoI* global associada aos cenários envolvidos na interpretação da questão (24).

1.4 Representação dos cenários usando a sintaxe do modelo relacional:

Considerando a relevância de 0.25, 0.35 e 0.40 para “suppliers”, “p_suppliers” e “products”, é possível representar a informação associada a cada um dos cenários da seguinte forma:

Cenário	Representação e Quantificação dos Cenários
Cenário 1	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers(X,Y,Z,R,T) \wedge$ $suppliers(1, supplier1, 20, 0.25, 0.166) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers(X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,200,0.35,0.166) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 2	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers(X,Y,Z,R,T) \wedge$ $suppliers(1, supplier1, 20, 0.25, 0.166) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers(X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,400,0.35,0.166) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 3	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers(X,Y,Z,R,T) \wedge$ $suppliers(1, supplier1, 40, 0.25, 0.166) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers(X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,200,0.35,0.166) \wedge$ $\neg products(X,Y,Z,W,R,R) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 4	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers(X,Y,Z,R,T) \wedge$ $suppliers(1, supplier1, 40, 0.25, 0.166) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers(X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,400,0.35,0.166) \wedge$ $\neg products(X,Y,Z,W,R,R) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$

Figura A.4 – Representação segundo o modelo relacional das soluções ou cenários que interpretam a questão (24).

2. Questão (25) a interpretar: *“Quais os fornecedores que no passado forneceram produtos de cor preta”*

Para a interpretação desta questão, os predicados envolvidos são: ”suppliers”, “companies” e “products”. O conjunto de soluções ou cenários que interpretam ou resolvem a questão, pode ser representado através

da Programação em Lógica Estendida (definição 3) ou através da nova representação através das relações quantificadas pela *QoI* (secção 4.3.2):

2.1 Representação dos cenários usando a Programação em Lógica Estendida:

Cenário 1	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2, 40) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 2	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2, 50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 3	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2, 60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 4	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2, 40) \wedge abducible_{companies}(2,1,50,company2, 50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 5	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2, 40) \wedge abducible_{companies}(2,1,50,company2, 60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 6	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers}(1, supplier1, 20) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$

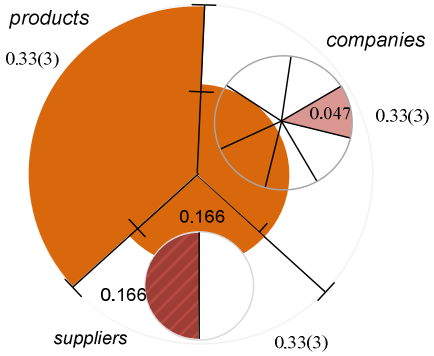
	$not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 50) \wedge$ $abducible_{companies}(2,1,50,company2,\ 60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 7	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 20) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 40) \wedge$ $abducible_{companies}(2,1,50,company2,\ 50) \wedge$ $abducible_{companies}(2,1,50,company2,\ 60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 8	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 40) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 9	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 50) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 10	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 11	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,\ 40) \wedge$ $abducible_{companies}(2,1,50,company2,\ 50) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 12	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$

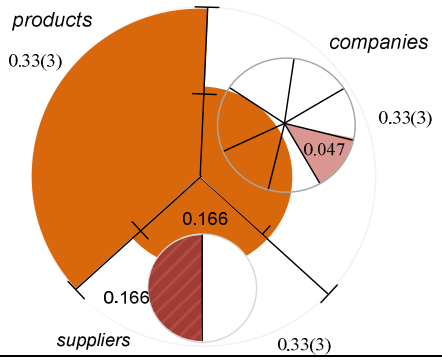
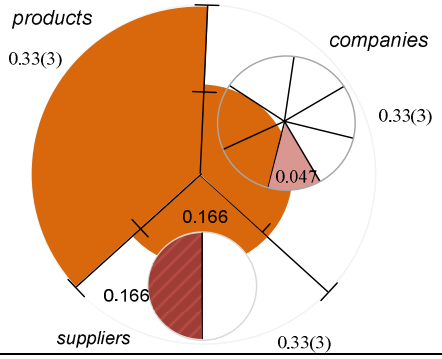
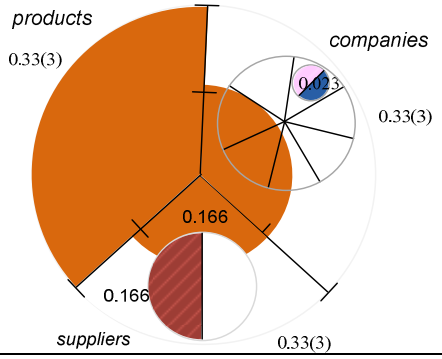
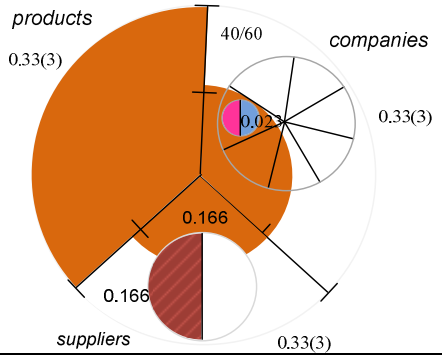
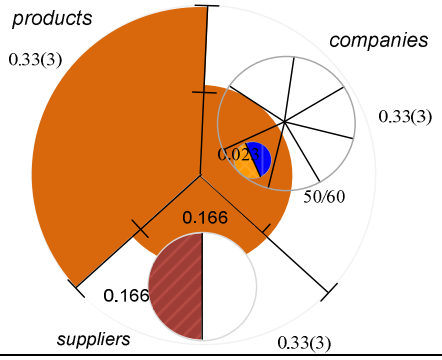
	$(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,\ 40) \wedge abducible_{companies}(2,1,50,company2,\ 60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)$
Cenário 13	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 40) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,\ 50) \wedge abducible_{companies}(2,1,50,company2,\ 60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 14	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 40) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,\ 40) \wedge abducible_{companies}(2,1,50,company2,\ 50) \wedge abducible_{companies}(2,1,50,company2,\ 60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$

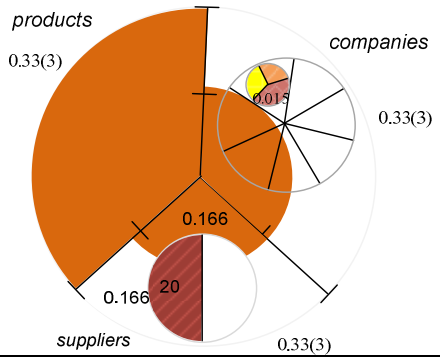
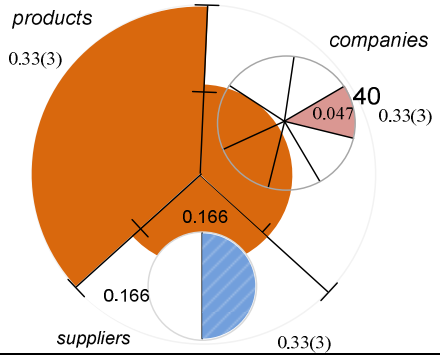
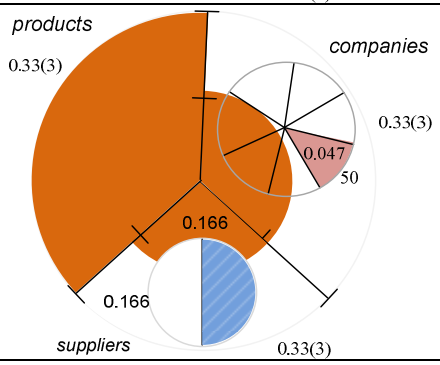
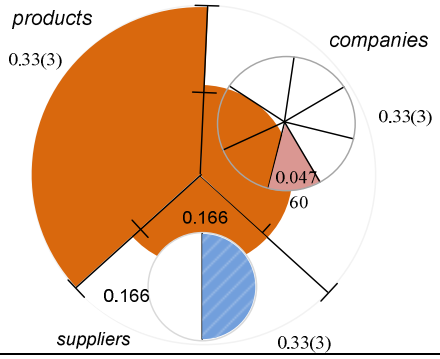
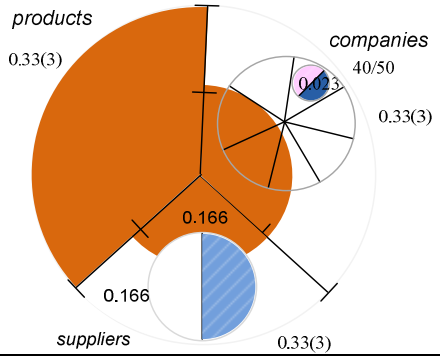
Figura A.5 – Representação em PLE das soluções ou cenários que interpretam a questão (25).

2.2 Quantificação da Qualidade da Informação associada a cada um dos cenários para a interpretação da questão (25):

Considerando o grau de relevância (ou importância) dos predicados da seguinte forma: 0.25, 0.35 e 0.40 para "suppliers", "companies" e "products", é possível (face ao exposto na secção 4.3.2,) quantificar e representar a quantificação do conhecimento associado a cada um dos cenários, em que $QoI_{(QN,M)}$, $N \in \{1,2,3\}$ correspondente ao número das questões e $M \in \{1,..R\}$, em que R é o número máximo de cenários possíveis de interpretar a questão em observação:

Cenário	Representação da QoI	Valor de Verdade (ou QoI)
Cenário 1		$QoI_{(Q2,1)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189

Cenário 2		$QoI_{(Q2,2)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189
Cenário 3		$QoI_{(Q2,3)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189
Cenário 4		$QoI_{(Q2,4)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181
Cenário 5		$QoI_{(Q2,5)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181
Cenário 6		$QoI_{(Q2,6)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181

Cenário 7		$QoI_{(Q2,7)}$= 0,25*0,166 + 0,35*0,015 + 0,33 * 0,4 = 0,179
Cenário 8		$QoI_{(Q2,8)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189
Cenário 9		$QoI_{(Q2,9)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189
Cenário 10		$QoI_{(Q2,10)}$= 0,25*0,166 + 0,35*0,047 + 0,33 * 0,4 = 0,189
Cenário 11		$QoI_{(Q2,11)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181

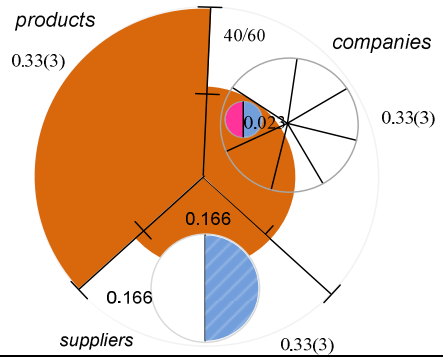
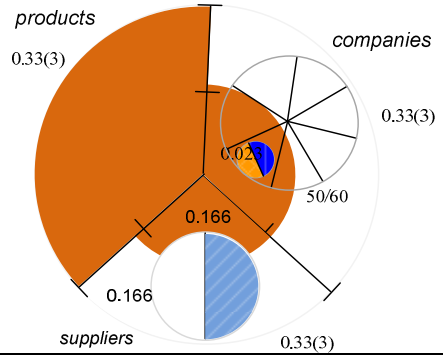
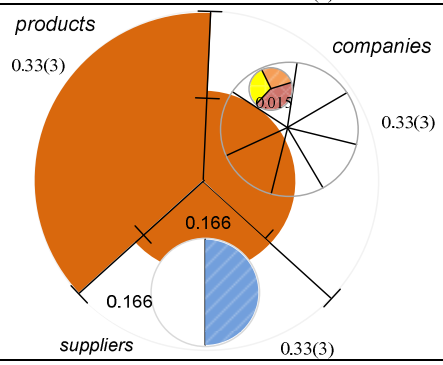
Cenário 12		$QoI_{(Q2,12)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181
Cenário 13		$QoI_{(Q2,13)}$= 0,25*0,166 + 0,35*0,023 + 0,33 * 0,4 = 0,181
Cenário 14		$QoI_{(Q2,14)}$= 0,25*0,166 + 0,35*0,015 + 0,33 * 0,4 = 0,179

Figura A.6 – Representação e Quantificação da *QoI* das soluções ou cenários que interpretam a questão (25).

1.3 Representação gráfica da *QoI* sobe os cenários que interpretam a questão em observação

Tendo em consideração o exposto na secção anterior, o gráfico com a *QoI* associada a todos os cenários envolvidos na interpretação ou resolução da questão é o seguinte:

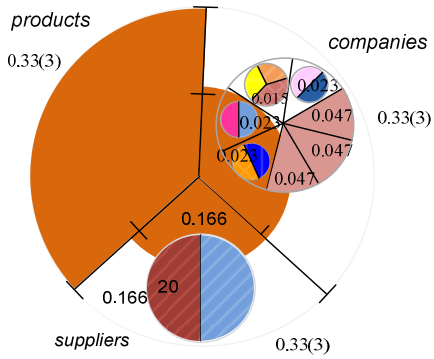


Figura A.7 – Gráfico da *QoI* global associada aos cenários envolvidos na interpretação da questão (25).

2.4 Representação dos cenários usando a sintaxe do modelo relacional:

Considerando a relevância de 0.25, 0.35 e 0.40 para “suppliers”, “companies” e “products”, é possível representar a informação associada a cada um dos cenários da seguinte forma:

Cenário	Representação e Quantificação dos Cenários
Cenário 1	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,40,0.35,0.189) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 2	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,50,0.35,0.189) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 3	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,60,0.35,0.189) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 4	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,40,0.35,0.181) \wedge$ $companies(2,1,50,company2,50,0.35,0.181) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 5	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,40,0.35,0.181) \wedge$ $companies(2,1,50,company2,60,0.35,0.181) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 6	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,50,0.35,0.181) \wedge$ $companies(2,1,50,company2,60,0.35,0.181) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.4,0.33)$ $\}$
Cenário 7	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $suppliers(1,\ supplier1,\ 20,0.25,0.166) \wedge$ $companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies\ (X,Y,Z,W,H,R,T)$ $companies(2,1,50,company2,40,0.35,0.179)$ $companies(2,1,50,company2,50,0.35,0.179)$

	companies(2,1,50,company2,50,0.35,0.179) products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) products(50,product1,black,200,0.4,0.33) }
Cenário 8	{suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) suppliers(1, supplier1, 40,0.25,0.166) companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) companies(2,1,50,company2,40,0.35,0.189) products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) products(50,product1,black,200,0.4,0.33) }
Cenário 9	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 40,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,50,0.35,0.189) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }
Cenário 10	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 50,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,60,0.35,0.189) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }
Cenário 11	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 40,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.35,0.181) ∧ companies(2,1,50,company2,50,0.35,0.181) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }
Cenário 12	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 40,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.35,0.181) ∧ companies(2,1,50,company2,60,0.35,0.181) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }
Cenário 13	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 40,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,50,0.35,0.181) ∧ companies(2,1,50,company2,60,0.35,0.181) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }
Cenário 14	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 40,0.25,0.166) ∧ ¬companies(X,Y,Z,W,H,R,T)← not companies (X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.35,0.179) ∧ companies(2,1,50,company2,50,0.35,0.179) ∧ companies(2,1,50,company2,60,0.35,0.179) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.4,0.33) }

Figura A.8 – Representação segundo o modelo relacional das soluções ou cenários que interpretam a questão (25).

3. Questão (26) a interpretar: ***"Quais os fornecedores que no passado forneceram produtos de cor preta e que no presente também estão em condições de o fazer"***

Para a interpretação desta questão, os predicados envolvidos são: "suppliers", "p_suppliers", "companies" e "products". O conjunto de soluções ou cenários que interpretam ou resolvem a questão, pode ser representado através da Programação em Lógica Estendida (definição 3) ou através da nova representação através das relações quantificadas pela *QoI* (secção 4.3.2):

Cenário 1	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 2	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 3	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 4	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge$

	$(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 5	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 6	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)\}$
Cenário 7	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)$

	$\}$
Cenário 8	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 9	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 10	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 11	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1,\ supplier1,\ 20) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 12	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$

	$not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 20) \wedge$ $(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$ $abducible_{p\_suppliers}(1,50,400) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,40) \wedge$ $abducible_{companies}(2,1,50,company2,60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 13	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 20) \wedge$ $(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$ $abducible_{p\_suppliers}(1,50,400) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,50) \wedge$ $abducible_{companies}(2,1,50,company2,60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 14	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 20) \wedge$ $(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$ $abducible_{p\_suppliers}(1,50,400) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,40) \wedge$ $abducible_{companies}(2,1,50,company2,50) \wedge$ $abducible_{companies}(2,1,50,company2,60) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$ $\}$
Cenário 15	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{suppliers}(X,Y,Z)) \wedge$ $abducible_{suppliers(}1,\ supplier1,\ 40) \wedge$ $(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$ $abducible_{p\_suppliers}(1,50,200) \wedge$ $(\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge$ $not\ abducible_{companies}(X,Y,Z,W,H)) \wedge$ $abducible_{companies}(2,1,50,company2,40) \wedge$ $(\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge$ $not\ abducible_{products}(X,Y,Z,W)) \wedge$ $products(50,product1,black,200)$

	$\}$
Cenário 16	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 17	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 18	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 19	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$

Cenário 20	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 21	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,200) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 22	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 23	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200)$

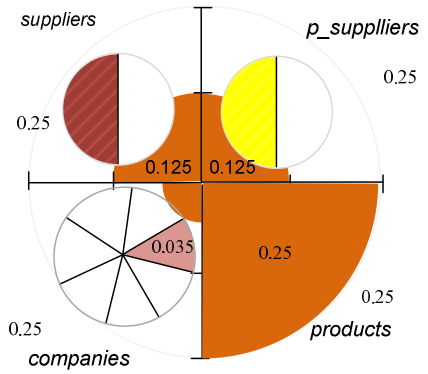
	$\}$
Cenário 24	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 25	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,50) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 26	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,40) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge products(50,product1,black,200) \}$
Cenário 27	$\{(\neg suppliers(X,Y,Z) \leftarrow not\ suppliers(X,Y,Z) \wedge not\ abducible_{suppliers}(X,Y,Z)) \wedge abducible_{suppliers(}1, supplier1, 40) \wedge (\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers(X,Y,Z) \wedge not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge abducible_{p\_suppliers}(1,50,400) \wedge (\neg companies(X,Y,Z,W,H) \leftarrow not\ companies(X,Y,Z,W,H) \wedge not\ abducible_{companies}(X,Y,Z,W,H)) \wedge abducible_{companies}(2,1,50,company2,50) \wedge abducible_{companies}(2,1,50,company2,60) \wedge (\neg products(X,Y,Z,W) \leftarrow not\ products(X,Y,Z,W) \wedge not\ abducible_{products}(X,Y,Z,W)) \wedge$

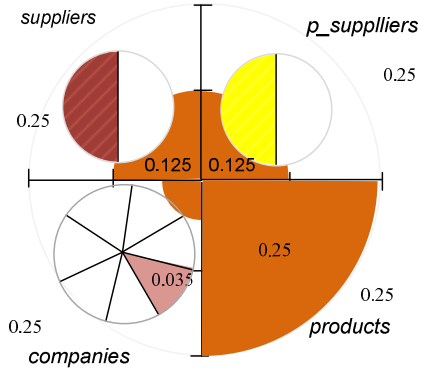
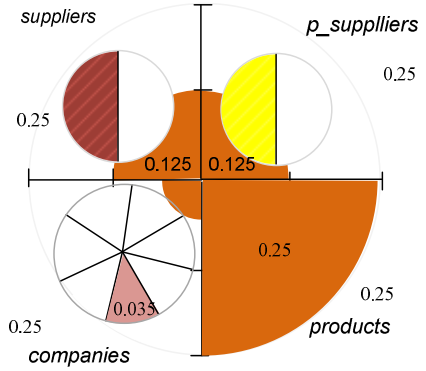
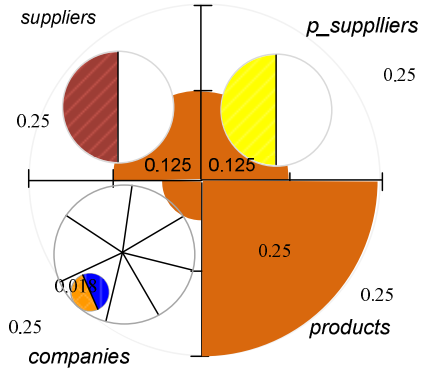
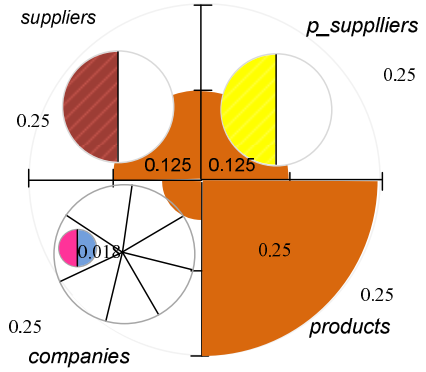
	products(50,product1,black,200) }
Cenário 28	{(¬suppliers(X,Y,Z)← not suppliers (X,Y,Z) ∧ not abducible$_{suppliers}$(X,Y,Z)) ∧ abducible$_{suppliers(}$1, supplier1, 40) ∧ (¬p_suppliers(X,Y,Z)← not p_suppliers (X,Y,Z) ∧ not abducible$_{p_suppliers}$(X,Y,Z)) ∧ abducible$_{p_suppliers}$(1,50,400) ∧ (¬companies(X,Y,Z,W,H) ← not companies(X,Y,Z,W,H) ∧ not abducible$_{companies}$(X,Y,Z,W,H)) ∧ abducible$_{companies}$(2,1,50,company2,40) ∧ abducible$_{companies}$(2,1,50,company2,50) ∧ abducible$_{companies}$(2,1,50,company2,60) ∧ (¬products(X,Y,Z,W) ← not products(X,Y,Z,W) ∧ not abducible$_{products}$(X,Y,Z,W)) ∧ products(50,product1,black,200) }

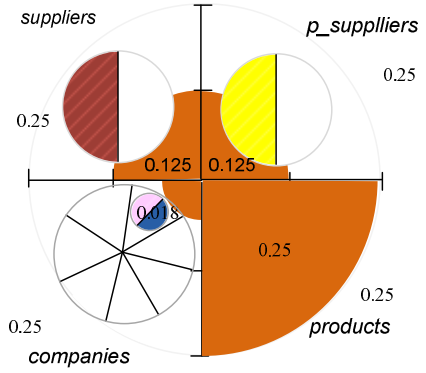
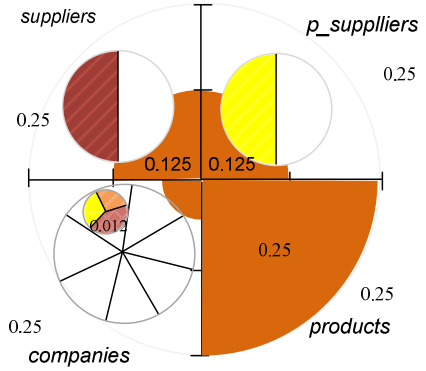
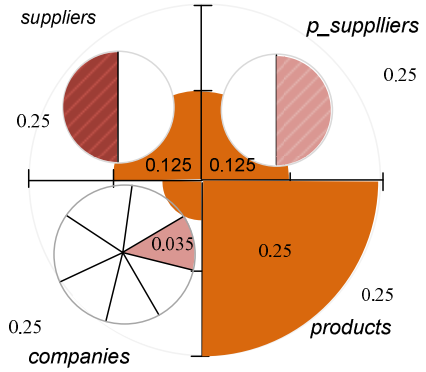
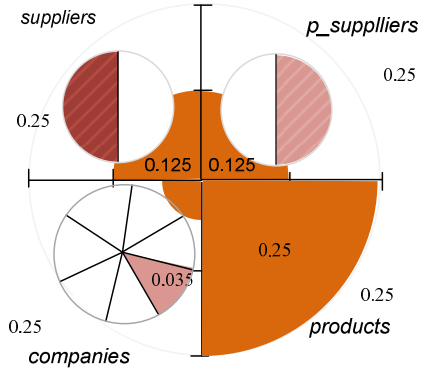
Figura A.9 – Representação em PLE das soluções ou cenários que interpretam a questão (26).

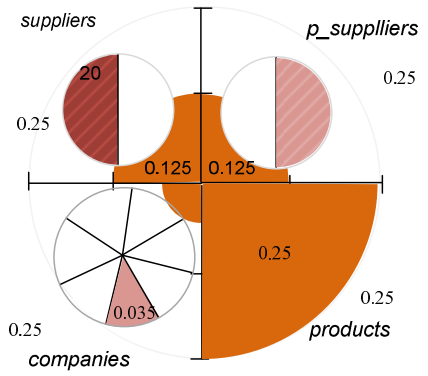
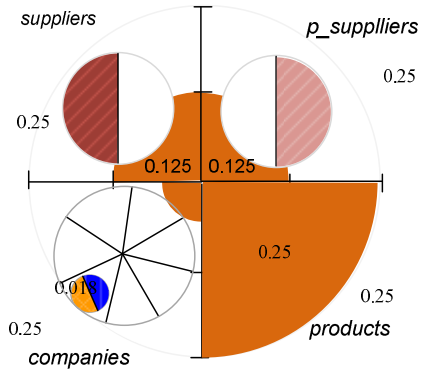
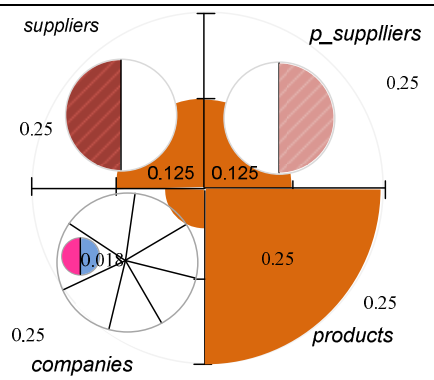
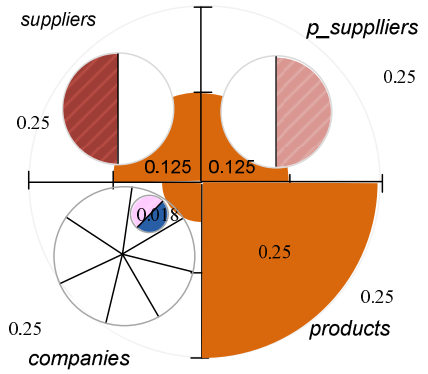
3.2 Quantificação da Qualidade da Informação associada a cada um dos cenários para a interpretação da questão (26):

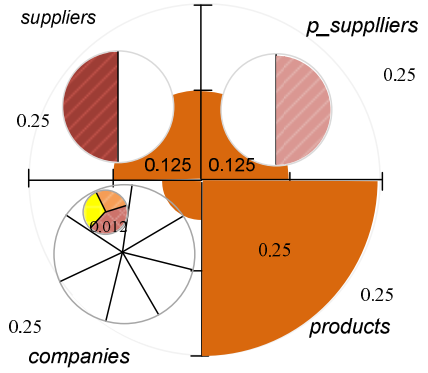
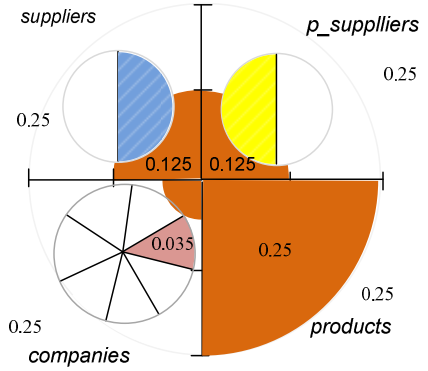
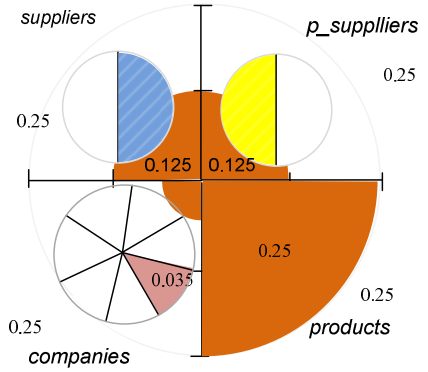
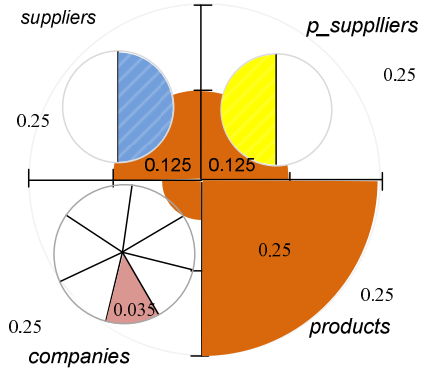
Considerando o grau de relevância (ou importância) dos predicados da seguinte forma: 0.25, 0.15, 0.25 e 0.35 para "suppliers", "p_suppliers", "companies" e "products", é possível (face ao exposto na secção 4.3.2,) é possível quantificar e representar a quantificação do conhecimento associado a cada um dos cenários, em que $QoI_{(QN,M)}$, $N \in \{1,2,3\}$ correspondente ao número das questões e $M \in \{1,..R\}$, em que R é o número máximo de cenários possíveis de interpretar a questão em observação.

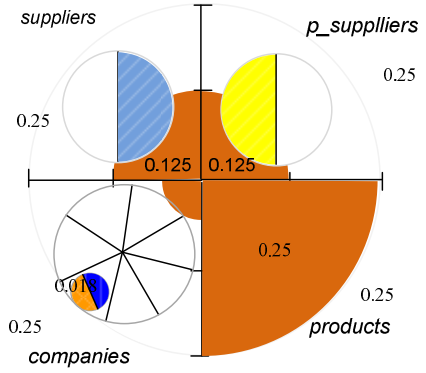
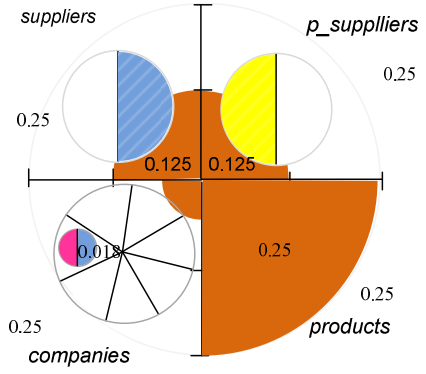
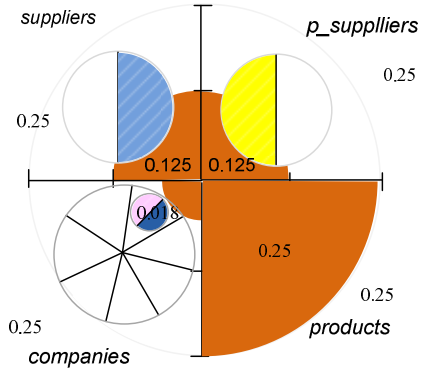
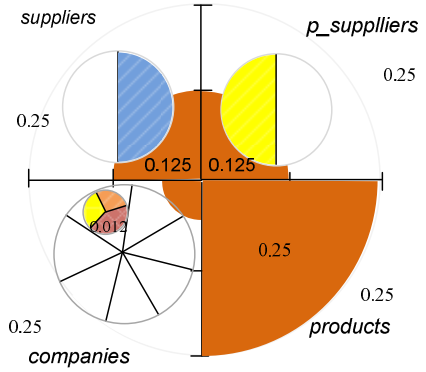
Cenário	Representação da QoI	Valor de Verdade (ou QoI)
Cenário 1		$QoI_{(Q3,1)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146

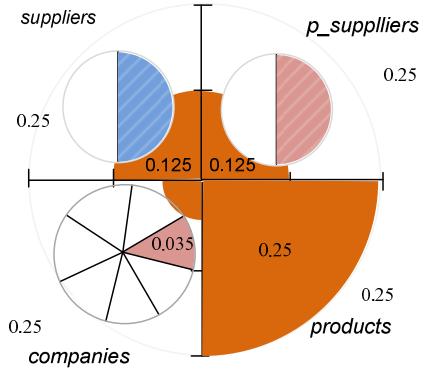
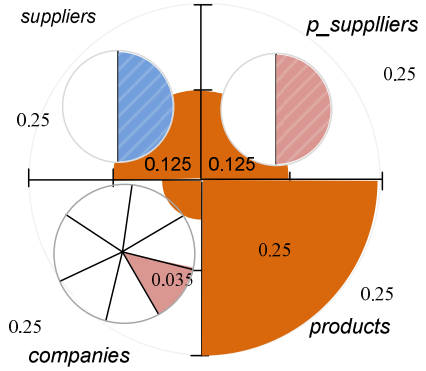
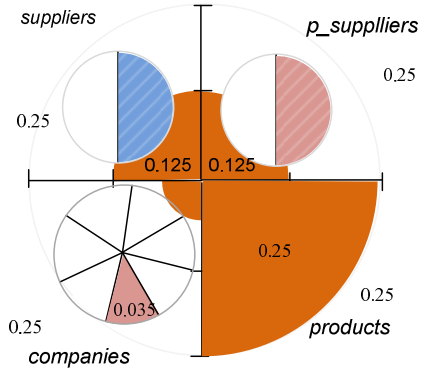
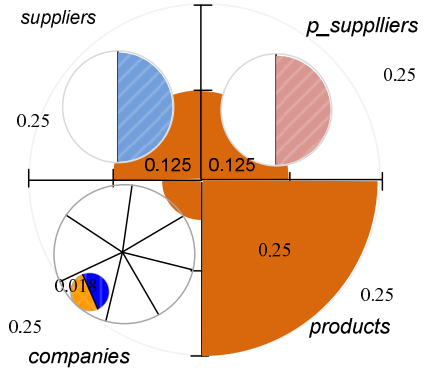
Cenário 2		$QoI_{(Q3,2)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 3		$QoI_{(Q3,3)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 4		$QoI_{(Q3,4)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 5		$QoI_{(Q3,5)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142

Cenário 6		$QoI_{(Q3,6)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 7		$QoI_{(Q3,7)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,012 + 0,35*0,25 = 0,140
Cenário 8		$QoI_{(Q3,8)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 9		$QoI_{(Q3,9)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146

Cenário 10		$QoI_{(Q3,10)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 11		$QoI_{(Q3,11)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 12		$QoI_{(Q3,12)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 13		$QoI_{(Q3,13)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142

Cenário 14		$QoI_{(Q3,14)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,012 + 0,35*0,25 = 0,140
Cenário 15		$QoI_{(Q3,15)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 16		$QoI_{(Q3,16)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 17		$QoI_{(Q3,17)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146

Cenário 18		$QoI_{(Q3,18)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 19		$QoI_{(Q3,19)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 20		$QoI_{(Q3,20)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 21		$QoI_{(Q3,21)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,012 + 0,35*0,25 = 0,140

Cenário 22		$QoI_{(Q3,22)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 23		$QoI_{(Q3,23)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 24		$QoI_{(Q3,24)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,035 + 0,35*0,25 = 0,146
Cenário 25		$QoI_{(Q3,25)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142

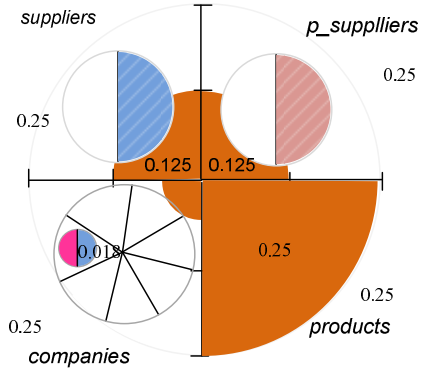
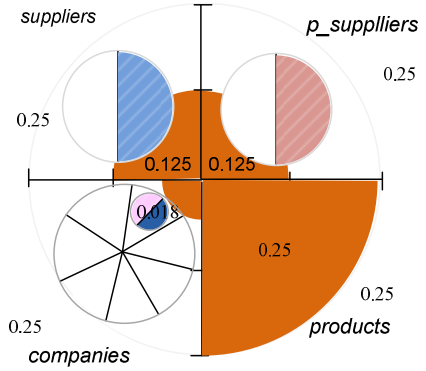
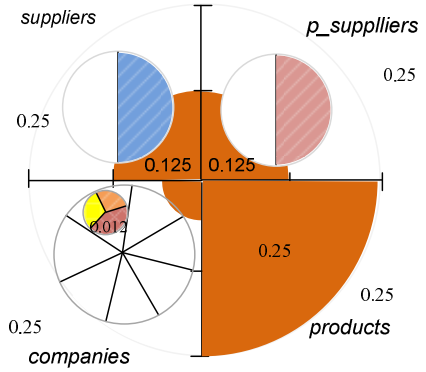
Cenário 26		$QoI_{(Q3,26)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 27		$QoI_{(Q3,27)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,018 + 0,35*0,25 = 0,142
Cenário 28		$QoI_{(Q3,27)}$= 0,25*0,125 + 0,15*0,125 + 0,25 * 0,012 + 0,35*0,25 = 0,140

Figura A.10 – Representação e Quantificação da *QoI* das soluções ou cenários que interpretam a questão (26).

1.3 Representação gráfica da *QoI* sobe os cenários que interpretam a questão em observação

Tendo em consideração o exposto na secção anterior, o gráfico com a *QoI* associada a todos os cenários envolvidos na interpretação ou resolução da questão é o seguinte:

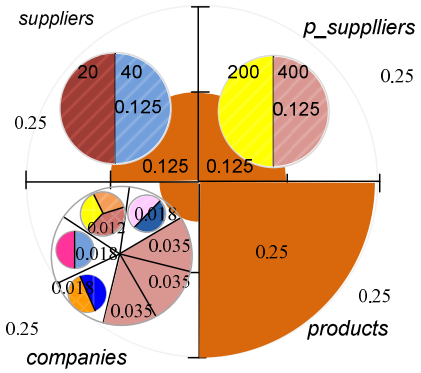


Figura A.11 – Gráfico da *QoI* global associada aos cenários envolvidos na interpretação da questão (26).

2.4 Representação dos cenários usando a sintaxe do modelo relacional:

Considerando a relevância de 0.25, 0.15, 0.25 e 0.35 para “suppliers”, “p_suppliers”, ””companies” e “products”, é possível representar a informação associada a cada um dos cenários da seguinte forma:

Cenário 1	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,200,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.25,0.035) ∧ (¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }*
Cenário 2	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 20,0.15,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,200,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,50,0.25,0.35) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }*
Cenário 3	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,200,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,60,0.25,0.125) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }*
Cenário 4	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 200.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,200,0.15,0.125) ∧*

	$\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies(X,Y,Z,W,H,R,T)$ $companies(2,1,50,company2,40,0.25,0.018)$ $companies(2,1,50,company2,50,0.25,0.018)$ $products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T)$ $products(50,product1,black,200,0.35,0.25)$ $\}$
Cenário 5	$\{suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T)$ $Suppliers(1,\ supplier1,\ 20,0.25,0.125)$ $p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers\ (X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,200,0.15,0.125)$ $companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies(X,Y,Z,W,H,R,T)$ $companies(2,1,50,company2,40,0.25,0.018)$ $companies(2,1,50,company2,60,0.25,0.018)$ $products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T)$ $products(50,product1,black,200,0.35,0.25)$ $\}$
Cenário 6	$\{suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T)$ $Suppliers(1,\ supplier1,\ 20,0.25,0.125)$ $p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers\ (X,Y,Z,R,T)$ $P\_suppliers(1,50,200,0.15,0.125) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies(X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,50,0.25,0.018) \wedge$ $companies(2,1,50,company2,60,0.25,0.018) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.35,0.25)$ $\}$
Cenário 7	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $Suppliers(1,\ supplier1,\ 20,0.25,0.125) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers\ (X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,200,0.15,0.125) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies(X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,40,0.25,0,012) \wedge$ $companies(2,1,50,company2,50,0.25,0,012) \wedge$ $companies(2,1,50,company2,60,0.25,0,012) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.35,0.25)$ $\}$
Cenário 8	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $Suppliers(1,\ supplier1,\ 20,0.25,0,125) \wedge$ $\neg p\_suppliers(X,Y,Z,R,T) \leftarrow not\ p\_suppliers\ (X,Y,Z,R,T) \wedge$ $p\_suppliers(1,50,400,0.15,0.125) \wedge$ $\neg companies(X,Y,Z,W,H,R,T) \leftarrow not\ companies(X,Y,Z,W,H,R,T) \wedge$ $companies(2,1,50,company2,40,0.25,0.035) \wedge$ $\neg products(X,Y,Z,W,R,T) \leftarrow not\ products(X,Y,Z,W,R,T) \wedge$ $products(50,product1,black,200,0.35,0.25)$ $\}$
Cenário 9	$\{\neg suppliers(X,Y,Z,R,T) \leftarrow not\ suppliers\ (X,Y,Z,R,T) \wedge$ $Suppliers(1,\ supplier1,\ 20,0.25,0.125) \wedge$ $(\neg p\_suppliers(X,Y,Z) \leftarrow not\ p\_suppliers\ (X,Y,Z) \wedge$ $not\ abducible_{p\_suppliers}(X,Y,Z)) \wedge$ $p\_suppliers(1,50,400,0.15,0.125) \wedge$

	¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,50,0.25,0.035) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 10	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) companies(2,1,50,company2,60,0.25,0.035) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 11	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.25,0.018) ∧ companies(2,1,50,company2,50,0.25,0.018) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 12	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.25,0.018) ∧ companies(2,1,50,company2,60,0.25,0.018) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 13	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,50,0.25,0.018) ∧ companies(2,1,50,company2,60,0.25,0.018) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 14	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 20,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.25,0.012) ∧ companies(2,1,50,company2,50,0.25,0.012) ∧

	companies(2,1,50,company2,60,0.25,0.012) ∧ *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 15	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *p_suppliers(1, supplier1, 40,0.25,0.125) ∧* *¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧* *p_suppliers(1,50,200,0.15,0.125) ∧* *¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧* *companies(2,1,50,company2,40,0.25,0.035) ∧* *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 16	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *Suppliers(1, supplier1, 40,0.25,0.125) ∧* *¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧* *p_suppliers(1,50,200,0.15,0.125) ∧* *¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧* *companies(2,1,50,company2,50,0.25,0.035) ∧* *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 17	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *Suppliers(1, supplier1, 40,0.25,0.125) ∧* *¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧* *P_suppliers(1,50,200,0.15,0.125) ∧* *¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧* *companies(2,1,50,company2,60,0.25,0.035) ∧* *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 18	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *Suppliers(1, supplier1, 40,0.25,0.125) ∧* *¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧* *p_suppliers(1,50,200,0.15,0.125) ∧* *¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧* *companies(2,1,50,company2,40,0.25,0.018) ∧* *companies(2,1,50,company2,50,0.25,0.018) ∧* *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 19	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *Suppliers(1, supplier1, 40,0.25,0.125) ∧* *¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧* *p_suppliers(1,50,200,0.15,0.125) ∧* *¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧* *companies(2,1,50,company2,40,0.25,0.018) ∧* *companies(2,1,50,company2,60,0.25,0.018) ∧* *¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧* *products(50,product1,black,200,0.35,0.25)* *}*
Cenário 20	*{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧* *Suppliers(1, supplier1, 40,0.25,0.125) ∧*

	¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *p_suppliers(1,50,200,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,50,0.25,0.018)* ∧ *companies(2,1,50,company2,60,0.25,0.018)* ∧ ¬*products(X,Y,Z,W,R,T)* ← *not products(X,Y,Z,W,R,T)* ∧ *products(50,product1,black,200,0.35,0.25)* }
Cenário 21	{ ¬*suppliers(X,Y,Z,R,T)*← *not suppliers (X,Y,Z,R,T)* ∧ *Suppliers(1, supplier1, 40,0.25,0.125)* ∧ ¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *p_suppliers(1,50,200,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,40,0.25,0.012)* ∧ *companies(2,1,50,company2,50, 0.25,0.012)* ∧ *companies(2,1,50,company2,60, 0.25,0.012)* ∧ ¬*products(X,Y,Z,W,R,T)* ← *not products(X,Y,Z,W,R,T)* ∧ *products(50,product1,black,200,0.35,0.25)* }
Cenário 22	{ ¬*suppliers(X,Y,Z,R,T)*← *not suppliers (X,Y,Z,R,T)* ∧ *Suppliers(1, supplier1, 40,0.25,0.125)* ∧ ¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *p_suppliers(1,50,400,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,40,0.25,0.035)* ∧ ¬*products(X,Y,Z,W,R,T)* ← *not products(X,Y,Z,W,R,T)* ∧ *products(50,product1,black,200,0.35,0.25)* }
Cenário 23	{ ¬*suppliers(X,Y,Z,R,T)*← *not suppliers (X,Y,Z,R,T)* ∧ *Suppliers(1, supplier1, 40,0.25,0.125)* ∧ ¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *p_suppliers(1,50,400,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,50,0.25,0.035)* ∧ ¬*products(X,Y,Z,W,R,T)* ← *not products(X,Y,Z,W,R,T)* ∧ *products(50,product1,black,200,0.35,0.25)* }
Cenário 24	{ ¬*suppliers(X,Y,Z,R,T)*← *not suppliers (X,Y,Z,R,T)* ∧ *Suppliers(1, supplier1, 40,0.25,0.125)* ∧ ¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *P_suppliers(1,50,400,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,60,0.25,0.035)* ∧ ¬*products(X,Y,Z,W,R,T)* ← *not products(X,Y,Z,W,R,T)* ∧ *products(50,product1,black,200,0.35,0.25)* }
Cenário 25	{ ¬*suppliers(X,Y,Z,R,T)*← *not suppliers (X,Y,Z,R,T)* ∧ *Suppliers(1, supplier1, 40,0.25,0.125)* ∧ ¬*p_suppliers(X,Y,Z,R,T)*← *not p_suppliers (X,Y,Z,R,T)* ∧ *p_suppliers(1,50,400,0.15,0.125)* ∧ ¬*companies(X,Y,Z,W,H,R,T)* ← *not companies(X,Y,Z,W,H,R,T)* ∧ *companies(2,1,50,company2,40,0.25,0.018)* ∧

	companies(2,1,50,company2,50,0.25,0.018) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 26	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 40,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) ∧ companies(2,1,50,company2,40,0.25,0.018) ∧ companies(2,1,50,company2,60,0.25,0.018) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }
Cenário 27	{¬suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) ∧ Suppliers(1, supplier1, 40,0.25,0.125) ∧ ¬p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) ∧ ¬companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) companies(2,1,50,company2,50,0.25,0.018) companies(2,1,50,company2,60,0.25,0.018) products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) products(50,product1,black,200,0.35,0.25) }
Cenário 28	{suppliers(X,Y,Z,R,T)← not suppliers (X,Y,Z,R,T) Suppliers(1, supplier1, 40,0.25,0.125) p_suppliers(X,Y,Z,R,T)← not p_suppliers (X,Y,Z,R,T) ∧ p_suppliers(1,50,400,0.15,0.125) companies(X,Y,Z,W,H,R,T) ← not companies(X,Y,Z,W,H,R,T) companies(2,1,50,company2,40,0.25,0.012) companies(2,1,50,company2,50,0.25,0.012) companies(2,1,50,company2,60,0.25,0.012) ∧ ¬products(X,Y,Z,W,R,T) ← not products(X,Y,Z,W,R,T) ∧ products(50,product1,black,200,0.35,0.25) }

Figura A.12 – Representação segundo o modelo relacional das soluções ou cenários que interpretam a questão (26).

B Excerto do código em PLE usado no estudo

```
%-------------------------------- - - - - - - - - -  -  -  -  -   -
%
% @ Jorge Ribeiro, 2010
%
% Este código corresponde a um excerto do trabalho realizado
% por Jorge Manuel Ferreira Barbosa Ribeir, no âmbito do
% seu doutoramento realizado na Universidade de Santiago de
% Compostela.
%
% A reprodução integral ou total do seu conteúdo só é permitida
% com a autorização escrita por parte do autor.
%
%
% IMPORTANTE
%
% Este código corresponde a um excerto de um exemplo ilustrativo
% realizado no âmbito do trabalho de doutoramento. O seu código é
% gerado em runtime a partir da framework VirtualInspector, um
% sistema computacional desenvolvido em Java que tem como objectivo
% a criação de sistemas virtuais que materializam a fusão entre
% os paradigmas simbólico, conexionista e evolutivo
%
%  versão 1.0
%
%-------------------------------- - - - - - - - - -  -  -  -  -   -

:- op( 900,xfy,'::' ).
:- op( 900,xfy,'?' ).
:-dynamic '-'/1.
:-dynamic abducible/1.
:-dynamic nao/1.
:- dynamic suppliers/3.
:- dynamic p_suppliers/3.
:- dynamic companies/5.
:- dynamic products/4.
:- dynamic listaDeSolucoesDaQuestao/1.

%-------------------------------- - - - - - - - - -  -  -  -  -   -
%
% Representação do Conhecimento
%
%-------------------------------- - - - - - - - - -  -  -  -  -   -

%-------------------------------- - - - - - - - - -  -  -  -  -   -
% Informação Associada ao Predicado Suppliers
%
% sintaxe: suppliers(#idS, nameS,ratingS)
%-------------------------------- - - - - - - - - -  -  -  -  -   -

-suppliers(X,Y,Z) :- nao( suppliers(X,Y,Z) ),
                     nao( abducible(suppliers(X,Y,Z))).
%-------------------------------- - - - - - - - - -  -  -  -  -   -
% Representação Positiva
%-------------------------------- - - - - - - - - -  -  -  -  -   -
suppliers(2,supplier2,60).
suppliers(3,supplier3,25).


%-------------------------------- - - - - - - - - -  -  -  -  -   -
% Representação de Informação Negativa
% Ex: É falso que o rating do fornecedor 4 seja 80%
%-------------------------------- - - - - - - - - -  -  -  -  -   -
-suppliers(4,supplier4,80).

%-------------------------------- - - - - - - - - -  -  -  -  -   -
% Representação de Informação Desconhecida
%-------------------------------- - - - - - - - - -  -  -  -  -   -
abducible(suppliers(1,supplier1,20)).
abducible(suppliers(1,supplier1,40)).

%-------------------------------- - - - - - - - - -  -  -  -  -   -
% Especificação do Invariante XOR
%-------------------------------- - - - - - - - - -  -  -  -  -   -
```

```
?((abducible(suppliers(X1,Y1,Z1)),abducible(suppliers(X2,Y2,Z2))),
  - (abducible(suppliers(X1,Y1,Z1)),abducible(suppliers(X2,Y2,Z2)))).

%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Representação de Informação do Tipo Não Conhecido
% Ex: O rating do supplier com o identificador '5' é desconhecido e
% não pode ser conhecido
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
suppliers(5,supplier5,not_permitted_to_known).

abducible(suppliers(X,Y,Z)) :-  suppliers(X,Y,not_permitted_to_known).
null_value(not_permitted_to_known).

%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Indicação de um Invariante que reestringue a inserção de informação
% numa determinada condição.Ex: não permitir que seja inserido o rating
% para o fornecedor com o identificador '5'
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
+suppliers(X,Y,Z) :: (solucoes( (X,Y,Z),(suppliers(5,Y,Z),nao(null_value(Z))),S),
                  comprimento( S,N ), N == 0
                  ).

%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Invariante Estrutural:  nao permitir a insercao de conhecimento
%                         repetido
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
+suppliers(X,Y,Z) :: (solucoes( (X,Y,Z),(suppliers(X,Y,Z)),S ),
                  comprimento( S,N ), N == 1
                  ).
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Invariante Referencial: nao admitir mais do que dois suppliers
%                         para um mesmo identificador
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
+suppliers(X,Y,Z) :: (solucoes( (X),(suppliers(X,Y,Z)),S ),
                  comprimento( S,N ), N =< 2
                  ).



%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Informação Associada ao Predicado p_suppliers
%
% sintaxe: p_suppliers(#idS,#idP, plafond)
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
-p_suppliers(X,Y,Z) :- nao( p_suppliers(X,Y,Z) ),
                       nao( abducible(p_suppliers(X,Y,Z))).

abducible(p_suppliers(1,50,200)).
abducible(p_suppliers(1,50,400)).

%Invariante XOR
?((abducible(p_suppliers(X1,Y1,Z1)),abducible(p_suppliers(X2,Y2,Z2))),
  - (abducible(p_suppliers(X1,Y1,Z1)),abducible(p_suppliers(X2,Y2,Z2)))).

p_suppliers(2,51,300).
p_suppliers(3,52,300).



%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Informação Associada ao Predicado companies
%
% sintaxe: companies(#idC,#idS,#idP,nameC,ratingC)
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
-companies(X,Y,Z,W,H) :- nao( companies(X,Y,Z,W,H) ),
                         nao( abducible(companies(X,Y,Z,W,H))).
abducible(companies(2,1,50,company2,40)).
abducible(companies(2,1,50,company2,50)).
abducible(companies(2,1,50,company2,60)).
companies(1,3,50,company1,35).
companies(3,2,51,company1,45).

%--------------------------------- - - - - - - - - - -  -  -  -  -   -
```

```
% Informação Associada ao Predicado products
%
% sintaxe: products(#idP,nameP,Color,Price)
%-------------------------------- - - - - - - - - - -  -  -  -  -   -
-products(X,Y,Z,W,H) :- nao( products(X,Y,Z,W,H) ),
                        nao( abducible(products(X,Y,Z,W,H))).
products(50,product1,black,200).
products(51,product2,white,150).
products(52,product3,blue,300).


%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicados especificando as dependências de Relacionamento entre os predicados.
% O predicado connection tem como objectivo definir as ligações entre duas entidades
%-------------------------------- - - - - - - - - - -  -  -  -  -   -

connection(suppliers(A,B,C),companies(D,A,P,N,R)).
connection(suppliers(A,B,C),p_suppliers(A,P,R)).
connection(p_suppliers(A,P,R),companies(D,S,P,N,Z)).
connection(p_suppliers(A,P,R),products(P,N,C,Z)).
connection(companies(D,S,P,N,R),products(P,NP,C,PR)).


%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado evolução: Tem como objectivo acrescentar informação à
% base de conhecimento. O predicado é bem sucedido se não houver
% invariantes que reestringam a inserção da informação.
%-------------------------------- - - - - - - - - - -  -  -  -  -   -
evolucao( Termo ) :-
    solucoes( Invariante,+Termo::Invariante,Lista ),
    insercao( Termo ),
    teste( Lista ).

insercao( Termo ) :- assert( Termo ).
insercao( Termo ) :- retract( Termo ),!,fail.

teste( [] ).
teste( [R|LR] ) :-
    R,
    teste( LR ).


%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado demo: Questao,Resposta -> {V,F}
%-------------------------------- - - - - - - - - - -  -  -  -  -   -

demo( Questao,verdadeiro ) :-
    Questao.
demo( Questao, falso ) :-
    -Questao.
demo( Questao,desconhecido ) :-
    nao( Questao ),
    nao( -Questao ).


%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado nao: Questao -> {V,F}
%-------------------------------- - - - - - - - - - -  -  -  -  -   -

nao( Questao ) :-     Questao, !, fail.
nao( Questao ).

%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% O predicado interpretar tem como objectivo aplicar o sistema de inferência
% a um número determinado de entidades
% interpretar([suppliers(X,Y,Z)],products(P,NP,black,PR),S)
%-------------------------------- - - - - - - - - - -  -  -  -  -   -

interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N==1,!,
                                                  retractall(listaDeSolucoesDaQuestao(_)),
                                                  provaTeoremas(Teorema, TipoDeResultado),
                                                  obterTodasAsSolucoes(Solucoes).
```

```
interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N==2,!,

                                                   daHeadLista(Teorema, H,T),
                                                   caminhos(H, T, Lc),
                                                   provaTeoremas(Lc, TipoDeResultado),
                                                   obterTodasAsSolucoes(Solucoes).

interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N>2,!,
                                          interpretaQuestao(Teorema, TipoDeResultado,
Solucoes).


interpretaQuestao([], _,_).
interpretaQuestao(Teorema, TipoDeResultado, Solucoes) :- daHeadLista(Teorema, H,T),
                                                    provaTeoremas([H], TipoDeResultado),
                                                    interpretaQuestao(T, TipoDeResultado,
Solucoes).


provaTeoremas([], _).
provaTeoremas([[H|T]], TipoDeResultado):- !,
                                       demoTeorema(H, TipoDeResultado),!,
                                       provaTeoremas(T, TipoDeResultado).

provaTeoremas([H|T], TipoDeResultado):- !,
                                       demoTeorema(H, TipoDeResultado),
                                       provaTeoremas(T, TipoDeResultado).


provaTeoremas([[H]|T], TipoDeResultado):- !,
                                       demoTeorema(H, TipoDeResultado),
                                       provaTeoremas(T, TipoDeResultado).


demoTeorema([Teorema], TipoDeResultado):- demoTeorema(Teorema, TipoDeResultado).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,verdadeiro),
                                        findall([verdadeiro,
TipoDeResultado],Teorema,SolucoesObtidas),
                                        assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,falso),
                                        findall([falso,TipoDeResultado],-
Teorema,SolucoesObtidas),
                                        assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,desconhecido),
                                            findall([desconhecido,TipoDeResultado],
                                            abducible(Teorema),SolucoesObtidas),
                                  assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).

%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicados Auxiliares
%-------------------------------- - - - - - - - - - -  -  -  -  -   -

obterTodasAsSolucoes(Solucoes):-findall(X,listaDeSolucoesDaQuestao(X),Solucoes).

listarTodasAsSolucoes:-findall(X,listaDeSolucoesDaQuestao(X),Solucoes),
                       imprimeLista(Solucoes).


imprimeLista([]).
imprimeLista([H|T]):- write(H), nl,
                      imprimeLista(T).

%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado que determina todos os caminhos entre uma origem e um destino
%
%-------------------------------- - - - - - - - - - -  -  -  -  -   -
caminhos(A, B, Lc) :- setof(Cam, caminho(A, B, Cam), Lc), !.
caminhos(_,_,[]).

caminho(A, B, Cam) :- travessia(A, B, [A], Cam).

travessia(A, B, Visitados, [B|Visitados]) :- connection(A,B).
travessia(A, B, Visitados, Cam) :- connection(A,C),
```

```
                                    C\==B,
                                    \+ member(C, Visitados),
                                    travessia(C, B, [C|Visitados], Cam).


%------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado que determina se há caminho entre os nodos A e B
%
%------------------------------- - - - - - - - - - -  -  -  -  -   -
ha_caminho(A, B) :- connection(A,B), !.
ha_caminho(A, B) :- connection(A,X),
                    ha_caminho(X,B).


daHeadLista([], _,_).
daHeadLista([H|[T]], H,T):-!.
daHeadLista([H|T], H,T):-!.

%------------------------------- - - - - - - - - - -  -  -  -  -   -
%
% No sentido de obter o código de debug do Prolog, tornou-se necessário
% desenvolver as rotinas abaixo endicadas. Desta forma, é armazenada
% numa lista, o debug efectuado internamente pelo Prolog, estando desta
% forma disponivel a partir do Java após a invocação de uma query
%
%------------------------------- - - - - - - - - - -  -  -  -  -   -
%%DYNAMIC_DEBUG_PROG%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
:-dynamic traceList/1.
prolog_trace_interception(Port, Frame, _PC, continue) :-
     prolog_frame_attribute(Frame, goal, Goal),
     prolog_frame_attribute(Frame, level, Level),
     insertTraceList((Port,Goal,Frame)),
     recordz(trace, trace(Port, Level, Goal)).
prolog_trace_interception(Port, Frame, _PC, continue).

insertTraceList(H):-bagof(ActualList,traceList(ActualList),L),!,
                    retract(traceList(_)),
                    appendLists(L,H,NewList),
                    assert(traceList(NewList)),
                    traceList(L1).
insertTraceList(H):-appendLists([],H,NewList),
                    assert(traceList(NewList)).
removeTraceList:-retract(traceList(_)).
appendLists([], List, List).
appendLists([First | Rest1], List, [First | Rest2]) :- appendLists(Rest1, List, Rest2).
%%END_OF_DYNAMIC_DEBUG_PROG%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
```

C Excerto do código em PLE contemplando a evolução temporal e a negação explícita

```
%-------------------------------- - - - - - - - - - -  -  -  -  -   -
%
% @ Jorge Ribeiro, 2010
%
% Este código corresponde a um excerto do trabalho realizado
% por Jorge Manuel Ferreira Barbosa Ribeir, no âmbito do
% seu doutoramento realizado na Universidade de Santiago de
% Compostela.
%
% A reprodução integral ou total do seu conteúdo só é permitida
% com a autorização escrita por parte do autor.
%
%
% IMPORTANTE
%
% Este código corresponde a um excerto de um exemplo ilustrativo
% realizado no âmbito do trabalho de doutoramento. O seu código é
% gerado em runtime a partir da framework VirtualInspector, um
% sistema computacional desenvolvido em Java que tem como objectivo
% a criação de sistemas virtuais que materializam a fusão entre
% os paradigmas simbólico, conexionista e evolutivo
%
%  versão 1.0
%
%-------------------------------- - - - - - - - - - -  -  -  -  -   -
:- op( 900,xfy,'::' ).
:- op( 900,xfy,'?' ).
:-dynamic '-'/1.
:-dynamic abducible/1.
:-dynamic nao/1.
:- dynamic suppliers/3.
:- dynamic p_suppliers/3.
:- dynamic companies/5.
:- dynamic products/4.
:-dynamic temp/1.
:- dynamic listaDeSolucoesDaQuestao/1.

-suppliers(T,X,Y,Z) :- nao( suppliers(T,X,Y,Z) ),
                       nao( abducible(suppliers(T,X,Y,Z))).
nao(suppliers(0,2,supplier2,60)).
suppliers(2,6,supplier6,88).
suppliers(1,2,supplier2,60).

-suppliers(1,4,supplier4,80).

abducible(suppliers(1,1,supplier1,20)).
abducible(suppliers(1,1,supplier1,40)).

?((abducible(suppliers(T,X1,Y1,Z1)),abducible(suppliers(T,X2,Y2,Z2))),
  - (abducible(suppliers(T,X1,Y1,Z1)),abducible(suppliers(T,X2,Y2,Z2)))).

suppliers(1,5,supplier5,not_permitted_to_known).

abducible(suppliers(T,X,Y,Z)) :-  suppliers(T,X,Y,not_permitted_to_known).
null_value(not_permitted_to_known).

-p_suppliers(T,X,Y,Z) :- nao( p_suppliers(T,X,Y,Z) ),
                         nao( abducible(p_suppliers(T,X,Y,Z))).

abducible(p_suppliers(0,1,50,200)).
abducible(p_suppliers(0,1,50,400)).

%Invariante XOR
?((abducible(p_suppliers(T,X1,Y1,Z1)),abducible(p_suppliers(T,X2,Y2,Z2))),
  - (abducible(p_suppliers(T,X1,Y1,Z1)),abducible(p_suppliers(T,X2,Y2,Z2)))).
```

```
p_suppliers(0,2,51,300).
p_suppliers(0,3,52,300).


-companies(T,X,Y,Z,W,H) :- nao( companies(T,X,Y,Z,W,H) ),
                           nao( abducible(companies(T,X,Y,Z,W,H))).
abducible(companies(0,2,1,50,company2,40)).
abducible(companies(0,2,1,50,company2,50)).
abducible(companies(0,2,1,50,company2,60)).
companies(0,1,3,50,company1,35).
companies(0,3,2,51,company1,45).

-products(T,X,Y,Z,W,H) :- nao( products(T,X,Y,Z,W,H) ),
                          nao( abducible(products(T,X,Y,Z,W,H))).
products(0,50,product1,black,200).
products(0,51,product2,white,150).
products(0,52,product3,blue,300).


connection(suppliers(T,A,B,C),companies(T,D,A,P,N,R)).
connection(suppliers(T,A,B,C),p_suppliers(T,A,P,R)).
connection(p_suppliers(T,A,P,R),companies(T,D,S,P,N,Z)).
connection(p_suppliers(T,A,P,R),products(T,P,N,C,Z)).
connection(companies(T,D,S,P,N,R),products(T,P,NP,C,PR)).


%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado evolução: Tem como objectivo acrescentar informação à
% base de conhecimento. O predicado é bem sucedido se não houver
% invariantes que reestringam a inserção da informação.
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
evolucao( Termo ) :-
    solucoes( Invariante,+Termo::Invariante,Lista ),
    insercao( Termo ),
    teste( Lista ).

insercao( Termo ) :- assert( Termo ).
insercao( Termo ) :- retract( Termo ),!,fail.

teste( [] ).
teste( [R|LR] ) :-
    R,
    teste( LR ).


%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado nao: Questao -> {V,F}
%--------------------------------- - - - - - - - - - -  -  -  -  -   -

nao( Questao ) :-     Questao, !, fail.
nao( Questao ).

%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% O predicado interpretar tem como objectivo aplicar o sistema de inferência
% a um número determinado de entidades
% interpretar([suppliers(X,Y,Z)],products(P,NP,black,PR),S)
%--------------------------------- - - - - - - - - - -  -  -  -  -   -

interpretar(Teorema, Solucoes):- term_variables(Teorema, TipoDeResultado),
                                 interpretar(Teorema, TipoDeResultado, Solucoes).

interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N==1,!,

retractall(listaDeSolucoesDaQuestao(_)),
                                                  provaTeoremas(Teorema,
TipoDeResultado),
```

```
                                                obterTodasAsSolucoes(Solucoes).

interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N==2,!,

                                                daHeadLista(Teorema, H,T),
                                                caminhos(H, T, Lc),
                                                provaTeoremas(Lc, TipoDeResultado),
                                                obterTodasAsSolucoes(Solucoes).

interpretar(Teorema, TipoDeResultado, Solucoes):- length(Teorema,N), N>2,!,
                                        interpretaQuestao(Teorema, TipoDeResultado,
Solucoes).


interpretaQuestao([], _,_).
interpretaQuestao(Teorema, TipoDeResultado, Solucoes) :- daHeadLista(Teorema, H,T),
                                                provaTeoremas([H], TipoDeResultado),
                                                interpretaQuestao(T,
TipoDeResultado, Solucoes).


provaTeoremas([], _).
provaTeoremas([[H|T]], TipoDeResultado):- !,
                                        demoTeorema(H, TipoDeResultado),!,
                                        provaTeoremas(T, TipoDeResultado).

provaTeoremas([H|T], TipoDeResultado):- !,
                                        demoTeorema(H, TipoDeResultado),
                                        provaTeoremas(T, TipoDeResultado).


provaTeoremas([[H]|T], TipoDeResultado):- !,
                                        demoTeorema(H, TipoDeResultado),
                                        provaTeoremas(T, TipoDeResultado).


demoTeorema([Teorema], TipoDeResultado):- demoTeorema(Teorema, TipoDeResultado).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,verdadeiro),
                                        findall([verdadeiro,
TipoDeResultado],Teorema,SolucoesObtidas),

assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,falso),
                                        findall([falso,TipoDeResultado],-
Teorema,SolucoesObtidas),

assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).
demoTeorema(Teorema, TipoDeResultado):- demo(Teorema,desconhecido),
                                        findall([desconhecido,TipoDeResultado],
abducible(Teorema),SolucoesObtidas),

assert(listaDeSolucoesDaQuestao(SolucoesObtidas)).



%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicados Auxiliares
%--------------------------------- - - - - - - - - - -  -  -  -  -   -

obterTodasAsSolucoes(Solucoes):-findall(X,listaDeSolucoesDaQuestao(X),Solucoes).

listarTodasAsSolucoes:-findall(X,listaDeSolucoesDaQuestao(X),Solucoes),
                        imprimeLista(Solucoes).


imprimeLista([]).
```

```
imprimeLista([H|T]):- write(H), nl,
                      imprimeLista(T).



%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado que determina todos os caminhos entre uma origem e um destino
%
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
caminhos(A, B, Lc) :- setof(Cam, caminho(A, B, Cam), Lc), !.
caminhos(_,_,[]).

caminho(A, B, Cam) :- travessia(A, B, [A], Cam).

travessia(A, B, Visitados, [B|Visitados]) :- connection(A,B).
travessia(A, B, Visitados, Cam) :- connection(A,C),
                                   C\==B,
                                   \+ member(C, Visitados),
                                   travessia(C, B, [C|Visitados], Cam).


%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Predicado que determina se há caminho entre os nodos A e B
%
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
ha_caminho(A, B) :- connection(A,B), !.
ha_caminho(A, B) :- connection(A,X),
                    ha_caminho(X,B).


daHeadLista([], _,_).
daHeadLista([H|[T]], H,T):-!.
daHeadLista([H|T], H,T):-!.


%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% Tratamento da Evolução Temporal da Informação
%
% Exemplo de uma possivel implementação do interpretador
% que contempla a evolução temporal da informação ao
% longo de vários estados, assim como a negação explicita
%
% NOTA: Este código é gerado em run-time a partir do Virtual
%       Inspector, sendo os predicados produzidos a partir deste
%       sistema.
%--------------------------------- - - - - - - - - - -  -  -  -  -   -
% NOTA: A interpretação usual deve ser entendida para regras, goals and assertations.
%

demo2(true,_).
demo2(!,_).
demo2(!,cut).
demo2((P,Q), V):- !, demo2(P,V), demo2(Q,V).
demo2((P;Q), V):- !, demo2(P,C); demo2(Q,V).
demo2((P:-Q), V):- !, demo2(Q,V), (V==cut, !, fail; V).


%INterpretador para a tempo e negacao explicita
demo2(nao(P), _):- P =..[H, CW|A], executarRegra(H, [CW|A], CW).
demo2(nao(P), _):-!, fail.
demo2(P, V):- P=..[H,T|A], demoBase(P,V),
              executarPredicado(H, [T|A], T).
demo2(_,_) :-!, fail.
```

```
demoBase( Questao,verdadeiro ) :-  Questao.
demoBase( Questao, falso ) :-  -Questao.
demoBase( Questao,desconhecido ) :- nao( Questao ),
                                    nao( -Questao ).



executarPredicado(H, [T|A], TA):-
                          (F1=..[H,T|A], F1, negacao_regra(H, T,TA, A);
terminacao(TA,NTA)),
                          executarPredicado(H, [NTA|A], TA).

executarPredicado(H, [CW|A], SCW):-
                          (name(H,LH), name(nao, [110,111, 116|H]),
                          F1=..[NAO, CW|A],
                          F1, executa_regra_posterior(H,CW,SCW,A);
                          terminacao(CW,PCW),
                          executarPredicado(H, [PCW|A], SCW)).

negacao_regra(_,_,CW,_):- var(CW), !.
negacao_regra(_,CW, CW,_) :- !.
negacao_regra(H,CW, CW,A) :-( name(H,LH), name(nao, [110, 111, 116| LH]),
                           F2=..[nao, SCW|A],
                           F2, !, fail;
                           estado_anterior(SCW, NSCW),
                           negacao_regra(H, CW, NSC, A)),!.

%
% Exemplo: executa_regra_posterior(A,B,C,D).
%

executa_regra_posterior(_,_,R,_) :- var(R), !.
executa_regra_posterior(_,R,R,_):-!.
executa_regra_posterior(H, R, RS, A):- (F2=..[H,RS|A],F2, !,fail;
                                        estado_anterior(RS,NR),
                                        executa_regra_posterior(H,R,NR,A)),!.

terminacao(CW,_):-(var(CW); CW==0), !, fail.
terminacao(CW, PCW):- PCW is CW-1, !.

estado_anterior(T,_) :- T==0, !.
estado_anterior(T, TA):- TA is T-1,!.
```

Bibliografia

[AA1998] O. Arieli e A. Avron. The value of the four values. *Artificial Intelligence, 102(1),* pp.:97-141, 1998.

[AA2006] Iara Almeida e José Alferes. An Argumentation-Based Negotiation for Distributed Extended Logic Programs. In *Proceedings of CLIMA VII*, pp.191–210, 2006.

[AB2002] C. Anglano and M. Botta. NOW G-Net: learning classification programs on networks of workstations. In *IEEE Trans. Evol. Comput.*, 6(5), pp.463-480, 2002.

[AB$^+$2002] J. Alferes, A. Brogi, J. A. Leite and L. M. Pereira. Evolving Logic Programs. In S. Flesca, S. Greco, N. Leone, G. Ianni (eds.), Proceedings of the 8^{th} European Conference on Logics in Artificial Intelligence (JELIA'02), pp.: 50-61, Spriger-Verlag, LNCS 2424, 2002.

[AB$^+$2003] J. J. Alferes, A. Brogi, J. A. Leite, L. M. Pereira. An Evolving Agent with EVOLP, in N. Leone, P. Rullo (eds.), Procs. Of APPIA-GULP-PRODE'03 Joint Conf. On Declarative Programming (AGP'03), Reggio Calabria, Italy, September, 2003.

[ABM2008] P. Auer, H. Burgsteiner, and W. Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks*, *no. 21*, pp.:786–795, 2008.

[Abd1991] W. Wan Abdullah. Neural Network logic. In O. Benham et al. (eds) Neural Networks from Biology to High Energy Physiscs, pp.: 135-142, 1991.

[Abra2002] A. Abraham. Optimization of evolutionary neural networks using hybrid learning algorithms, In *IEEE 2002 Joint International Conference on Neural Networks, Vol. 3, IEEE Press, New York,* pp.: 2797–2802, 2002.

[Abra2004] A. Abraham. Meta learning evolutionary artificial neural networks. In *Neurocomputing 56,* pp.:1-38, 2004.

[ADP1993] J. J. Alferes, P. M. Dung and L. M. Pereira. Scenario Semantics of Extended Logic Programs. In *2nd Int. Workshop on Logic Programming and Non-Monotonic Reasoning, L. M. Pereira and A. Nerode (eds.),* pp.: 334-348, MIT Press, 1993.

[Ahm1996] Wan Ahmad Tajuddin Wan. Logic Programming in Neural Networks. In Proceedings of the Malaysian Journal of Computer Science, Vol. 9 No. 1, June 1996, pp.: 1-5, 1996.

[AHP2000] J. J. Alferes, H. Herre and L. M. Pereira. Partial Models of Extended Generalized Logic Programs. *In J. Lloyd et al. (eds.), Procs. Of First International Conference on Computational Logic (CL 2000), London, UK,* pp.: 149-163, LNAI 1861, Springer, July 2000.

[Ait2009] Hassan Ait-Kaci. Children's Magic Won't Deliver the semantic, *CACM, March, Volume 52 , Issue 3,* pp.: 8-9, 2009.

[AK1989] Aarts, E., Korst, J. Simulated Annealing and Boltzman Machines. *Wiley, Chichester, UK*, 1989.

[ALP+1998] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska and T. C. Przymusinski. Dynamic Logic Programming. *In A. Cohn, L. Schubert and S. Shapiro (eds.), Procs. Of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy,* pp.: 98-109, Morgan Kaufmann, June 1998.

[Ann2006] Anne Rooney. Einstein in his own words. *Gramercy – Arcturus Publishing Limited*, 2006.

[AN+2004] C. Analide, P. Novais, J. Machado and J. Neves. Anthropathy and its Assessment in Virtual Entities. In Innovations through Information Technology, proceedings of the IRMA 2004 International Conference, New Orleans, USA, 2004.

[AN+2006] C. Analide, P. Novais, J. Machado and J. Neves. Quality of Knowledge in Virtual Entities, *in Encyclopedia of Communities of Practice in information and Knowledge Management, Idea Group Inc,* pp.: 436-442, 2006.

[Ang2000] P. Angeline. Parse Trees - Evolutionary Computation I: Basic Algorithms and Operators. *In: Back, T., et al (eds), Institute of Physics Publishing, Bristol,* 2000.

[AN2007] A. Asuncion and D. Newman.UCI machine learning repository, *2007. Available: www.ics.uci.edu/_mlearn/MLRepository.html*

[AP+1999] J. J. Alferes, L. M. Pereira, H. Przymusinska and T. C. Przymusinski. LUPS – a language for updating logic programs. In M. Gelfond, N. Leone and G. Pfeifer (eds.), Procs. Of the 5^{th} International Conference on Logic Programming and Nonmonotonic Reasoning, El Paso, Texas USA, pp.: 162-176, LNAI-1730, Springer, December 2-4, 1999.

[AP+2010] Carlos Alberto, A. Padilha, N. Lima et al. An Genetic Approach to Support Vector Machines in Classification Problems. In proceedings of the WCCI 2010 IEEE World Congress on Computational Intelligence, July, 18-23, 2010 – CCIB, Barcelona, Spain, pp.:776-399, 2010.

[AP1996] J. J. Alferes and L. M. Pereira. Reasoning with Logic Programming. In *Lecture Notes in Artificial Intelligence. Berlin. Springer-Verlag,* pp.:326, 1996.

[AR1994] Krzysztof Apt and Ronald Bol. Logic Programming and negation: a survey. *Journal of Logic Programming, vol. 19, 20,* pp.:9-71, 1994.

[AT1995] N. H. Anderson and D. M. Titterington. Beyond the Binary Boltzmann Machine. In *IEEE Transactions on Neural Networks, vol. 6, n.º 5*, pp.: 1229-1236, 1995.

[BAH2005] Sebastian Bader, A. D'Avila Garcez, P. Hitzler, Computing First-Order Logic Programs by Fibring Artificial Neural Networks. *Russell, Z. Markov (eds.): Proceedings of the Eighteenth International Florida Artificial Intelligence Research Symposium Conference, Clearwater Beach, Florida, USA. AAAI Press 2005*, pp.:314-319, 2005.

[Bel1977] N. Belmap. How a computer should think. In *Comtemporary aspects of philosophy,* pp.: 30-56, 1977.

[Ben1999] P. Bentley. Evolutionary Design by Computers. *Morgan Kaufmann Publishers Inc.*, 1999.

[BF1987] Nicole Bidoit and Christine Froidevaux. Minimalism subsumes default logic and circumscription. *In Proc. LICS-87*, pp.:89-97, 1987.

[BFM2000a] Back, T., Fogel, D. B., and Michalewicz, Z. Evolutionary Computation 1: Basic Algorithms and Operators. *Institute of Physics Publishing*, 2000.

[BFM2000b] Back, T., Fogel, D. B., and Michalewicz, Z. Evolutionary Computation 2: *Advance Algorithms and Operators. Institute of Physics Publishing*, 2000.

[BG1994] Baral, C., Gelfond, M., Logic Programming and Knowledge Representation, Research Report, Computer Science Department, University of Texas at El Paso, Texas, 1994.

[BG1993] I. Bratko and M. Grobelnik. Inductive Learning Applied to program construction and verification. *In proceedings of the 3^{rd} International Workshop on Inductive Logic Programming,* pp.:279-292, 1993.

[BHH2004] S. Bader, P. Hitzler and S. Holldobler. The integration of connectionist and first order logic knowledge representation and reasoning as challenge for Artificial Intelligence. *In: L. Li and K.K. Yen, Proceedings of the Third International Conference on Information, Tokyo, Japan, International Information Institute,* pp.: 22-33, 2004.

[BH+2007] S. Bader, P. Hitzler and S. Holldobler and A. Witzei. A fully connectionist model generator for covered furst-order logic programs. *In Manuela Veloso, editor, Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), Hyderabad, India*, pp.: 666-671, AAAI Press, 2007.

[BHH2008] S. Bader, P. Hitzler and S. Holldobler. Connectionist Model Generation: a first-order approach. *In Neurocomputing 71,* pp.:2420-2432. 2008.

[BK2005] E. Bauer, G. Kókai. Learning from noise data with the help of logic programming systems. *In : Freund, R. WSEAS, AIKED2004*, pp.:104-110, 2005.

[BNT2008] Gerhard Brewka, I. Niemela and M. Truszczynski. NonMonotonic Reasoning. *In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation, chapter 6*, pp.:239-284. Elsevier Science, Amsterdam, 2008.

[BN+1998] W. Banzhaf, P. Nordin, R. Keller, F. Francone. Genetic Programming: Aa Introduction. *Morgan Kaufmann, San Francisco*, 1998.

[Bod2002] Mickael Boden. A Guide to Recurrent Neural Networks and Backpropagation. In the dallas project, SICS Technical Report, SICS, 2001.

[Boo1854] George Boole. An Investigation of the Laws of Thought. *Walton & Maberly*, 1854.

[B+2001] A. Blanco et al. A real-coded genetic algorithm for training recurrent neural networks. In *Neural Networks 14,* pp.:93-105, 2001.

[Bra1995] J. Branke. Evolutionary algorithms in neural network design and training – A review. In *Proceedings of the First Nordic Workshop on Genetic Algorithms and their Applications*, *Jarmo T. Alander, Ed., Vaasa, Finland*, pp.:145–163, 1995.

[Brat1990] I. Bratko. Prolog Programs for Artificial Intelligence. *Addison Wesley reading : MA*, 1990.

[Brew1996] Gerhard Brewka. Well-founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, vol. 4, pp.19-36, 1996.

[Bro1980] D. Brohow. Special Issue on Non Monotic Logics. *Artificial Intelligence 13*, 1980.

[BS2001] A. Browne and R. Sun. Connectionist inference models. *In Neural Networks 14(10)*, pp.:1331–1355, 2001.

[Bus1995] Buss, Samuel R.. On Herbrand's Theorem, *in Maurice, Daniel; Leivant, Raphaël, Logic and Computational Complexity, Lecture Notes in Computer Science, Berlin, New York: Springer-Verlag*, pp.:195–209, 1995.

[CA+2002] P. A. Castillo, M. G. Arenas, J. J. Castillo-Valdivieso, J. J. Merelo, A. Prieto, and G. Romero, "Artificial neural networks design using evolutionary algorithms," in *Proceedings of the Seventh World Conference on Soft Computing*, 2002.

[CE2009] Thira Chavarnakul e D. Enke. A hybrid stock trading system for intelligent technical analysis-based equivolume charting. *In Neurocomputing archive, vol. 72 ,* Issue 16-18, pp.:3517-3528, 2009.

[CG+1997] D. Clouse, C. Giles, B. Horne, G. Cottrell. Time-delay neural networks: representation and induction of finite-state machines. *In Neural Networks, IEEE Transactions, vol. 8,* Issue:5, pp.:1065-1070, 2009.

[CH1985] Ashok Chandra and David Harel. Horn clause queries and generalizations. *In Journal of Logic Programming, 2 (1),* pp.:1-5, 1985.

[Chi2008] H. Christiansen. Executable specifications for hypothesis-based reasoning with Prolog and Constraint Handling Rules. *In Journal of Applied Logic*, 2008.

[CK1990] C. Chang and H. Keisler. Model Theory - Third edition. *Studies in Logic and Foundations of Mathematics*, North Holland, 1990.

[Cla1978] Keith Clark. Negation as failure. *In Herve Gallaire and Jack Minker editors, Logic and Data Bases,* pp.: 293-322. Plenum Press, New York, 1978.

[CM1981] W. Clocksin and C. Mellish. Programming in PROLOG. *Springer-Verlag*, 1981.

[CN+2008] Ricardo Costa, P. Novais, G. Marreiros, C. Ramos and J. Neves. VirtualECare: Group Decision, Supported by Idea Generation and Argumentation. In the *9th IFIP Working Conference on Virtual Enterprises (PRO-VE 2008)*, Poznan, Poland, 2008.

[CRN2001] Paulo Cortez, M. Rocha, J. Neves. Evolving Time Series Forecasting Neural Network Models. *In Proceedings of the third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models, ISAS2001*, pp.:84-91, 2001.

[CRF1994] D. Corne, Ross, P., H. Fang. Fast practical evolutionary timetabling. *In lecture notes in Computer science, vol. 865,* pp.251-263, Springer-Verlag, 1994.

[CS+2006] Alexandros Chortaras, Giorgos Stamou, Andreas Stafylopatis, and Stefanos Kollias. A Connectionist Model for Weighted Fuzzy Programs. *In Proc. Of the International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada* pp.:3005-3062, 2006.

[CY+2006] C. Lee, H. Yang, T. Chen., S. Ma. A Comparative Study on Supervised and Unsupervised Learning Approaches for Multilingual Text Categorization. *In Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, 2006.

[dA+2001] A. d'Avila Garcez, K. Broda, and D. Gabbay. Symbolic knowledge extraction from trained neural networks: a sound approach. *In Artificial Intelligence Series, vol. 125, 1(2)*, pp.:155-207 2001.

[dA+2002] A. d'Avila Garcez, K. Broda, and D. Gabbay. Neural-Symbolic Learning Systems. *In Foundations and Applications, Springer,* 2002.

[dA+2007] A. d'Avila Garcez, L. Lamb and D. Gabbay. Connectionist modal logic: Representing modalities in neural networks. *Theoretical Computer Science, 371 (1-2),* pp.:34-53, 2007.

[Dar1859] Charles Darwin. "On the Origin of Species by Means of Natural Selection", *John Murray*, 1859.

[DHN1976] R. Duda, P. Hart and N. Nilsson. Subjective Bayesian methods for rule-based inference systems. *In AFIPS Conference Proceedings of the 1976 National Computer Conference, vol. 45,* pp.:1705-1082, 1976.

[Din2006] Amorim Dinani. Redes ART com categorias internas de geometria irregulas. *PhD thesis, faculdade de física, departamento de Electrónica e Computacion, Universidade de Santiago de Compostela, Spain*, 2006.

[Div2001] F. Divina. Evolutionary concept learning in first order logic. *Technical Report IR-494, Vrje Universiteit Amsterdam*, 2001.

[Div2004] F. Divina Hybrid Genetic Relational Search for Inductive Learning. *Ph.D. Thesis. Vrje Universiteit Amsterdam*, 2004.

[DK2002] M. Denecker e A. Kakas. Abduction in logic programming. *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I. A. Kakas and F. Sadri, Springer Verlag. 2407*: pp.:402-436, 2002.

[DM2001] F. Divina, E Marchiori. Knowledge based evolutionary programming for inductive learning in first-order logic. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp.:173. Morgan Kaufman, 2001.

[DS2004] M. Dorigo and T. Stutzle. Ant Clony Optimization. *MIT Press*, 2004.

[EK1976] Van Emden, M. and Kowalski, R. The Semantics of Predicate Logic as a Programming Language In JACM , Vol. 23, No. 4, pp.:733-742, 1976.

[Emd2006] Maarteen van Emden. The early days of Logic Programming: A Personal Perspective. *The Association of Logic Programming, newsletter,* 2006.

[ES2003] Eiben, A. and J. Smith. Introduction to Evolutionary Computing, *Springer-Verlag*, 2003.

[EST2005] T. Eiter, M. Fink, G. Sabbaini and H. Tompits. Reasoning about Evolving NonMonotonotic Knowledge Bases. *In ACM Transactions on Computational Logic (TOCL) \t _self* archive vol 6, issue 2, pp:.389–440, 2005.

[FDM2008] D. Floreano, P. Durr and C. Mattiussi. Neuroevolution: from architectures to learning. *Evol. Intell. 1(1),* pp.:47-62, 2008.

[FK1997] T. Fung and R. Kowalski. The IFF Proof procedure Logic Programming. *Journal of Logic Programming 133 (2),* pp.:151-165, 1997.

[FK2000] Peter Flach, Antonis Kakas. Abduction, Induction, and the Logic of Scientific Knowledge Development. *Pure and Applied Logic, Kluwer*, 2000.

[For1984 K. Forbus. Qualitative Process Theory. *Artificial Intelligence, 24*, pp.: 85-168, 1984.

[For1996] K. Forbus. Qualitative reasoning. *CRC Handbook of Computer Science and Engineering,* pp.: 715-733, 1996.

[FOW1966] L. Fogel, A. Owens and M. Walsh. Artificial Intelligence through a Simulated Evolution. *John Wiley & Sons*, 1966.

[FR$^+$2010^a] M. Fernandez-Delgado, J. Ribeiro, E. Cernadas and S. Barro. Fast weight calculation for kernel-based perceptron in two-class classification problems. *In proceedings of the IEEE – 2010 International Joint Conference on Neural Networks, Barcelona*, pp.:1940-1945, ISBN:978-1-4244-6920-8, 2010.

[FR$^+$2010b] M. Fernandez-Delgado, J. Ribeiro, E. Cernadas and S. Barro. A Parallel Perceptron network for classification with direct calculation of the weights optimizing error and margin. *In proceedings of the IEEE – 2010 International Joint Conference on Neural Networks, Barcelona*, pp.:341-348, ISBN:978-1-4244-6920-8, 2010.

[Gab1982] D. Gabbay. Intuitionistic: Basis for Non-Monotic Logic. *In Proceedings CADE-6, LNCS 138, Springer-Verlag*, pp.: 260-273, 1982.

[GB+2010] M. Guillame-Bert, K. Broda e A. D'Avila Garcez. First-order logic learning. In *WCCI 2010 IEEE World talto ƒ Computational Intelligence,* pp.:2004-2011, Barcelona, Spain, 2010.

[GC+2007] Frederico Guimares, F. Campelo et al. Optimization of Cost Functions Using Evolutionary Algorithms With Local Learning and Local Search. In *Proc. of the IEEE Transactions on Magnetics, vol. 43, issue 4,* pp.:1641-1644, 2007.

[Gel1987] Michael Gelfond. On stratified autoepistemic theories. *In Proc. AAAI-87,* pp.:207-211, 1987.

[Gel1988] Allen Van Gelder. Negation as failure using tight derivations for general logic programs. In *Jack Minker, editor, Foundations of Deductive Databases and Logic Programming*, pp.:149-176. Morgan Kaufmann, San Mateo, CA, 1988.

[Gel1987] Michael Gelfond. On stratified autoepistemic theories. *In Proc. AAAI-87,* pp.207-211, 1987.

[GHR1995] D. Gabbay, C. Hogger and J. Robibsob. Handbook of Logic. *In Artificial Intelligence and Logic Programming, vol 4, Oxford University Press*, pp.:1-34, 1995.

[GK2002] Gerstner W. e Kistler, W. Spiking Neuron Model - Single Neurons, Populations, Plasticity Handbook of Logic. Cambridge University Press, 2002.

[Gin1991] M. Gingsberg. Readings in Nonmonotic Reasoning. *Morgan Kaufman Publishers*, Inc, Los Altos, Califórnia, EUA, 1991.

[Gir1987] J. Girard. Linear Logic. *In Theoritical Computer Science, 50*, pp.:1-102, 1987.

[GL1990] Michael Gelfond and Vladimir Lifschitz. Logic Programs with classical negation. *In David Warren and Peter Szeredi, editors, Logic Programming: Proc. Seventh Int'l Conf.,* pp.:579-597, 1990.

[GL1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing, 9:* pp.:365-385, 1991.

[GL1998] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. *In Robert Kowalski and Kenneth Bowen, editors, Logic Programming: Proc. Fifth Int'l Conf. And Symp.*, pp.:1070-1080, 1998.

[Glo1996] F. Glover. Tabu search and adaptive memory programming – advances, applications, and challenges. In *R.S. Barr, R.V. Helgason, J.L. Kennington, Eds. Interfaces in Computer Science and Operations Research. Kluwer Academic Publishers*, Norwell, MA, pp.:1-75, 1996.

[GM+2010] P. Gomes, A. Marques, A. Costa, P. Novais and J. Neves. Patient Monitoring under Ambient Intelligence Setting. *In Ambient Intelligence, Future Trends – International Symposium on Ambient Intelligence (ISAmI 2010), Augusto JC., Corchado JM, Novais P., Analide C. (Eds.), Springer – Series Advances in Intelligent and Soft Computing, vol. 72, International Symposium on Ambient Intelligence,* Guimarães, Portugal, 2010.

[GR1986] J. Gallier and S. Raatz. SLD-Resolution Methods for Horn Clauses with Equality Based on *E*-Unification. In *Proc. 1986 Symp. On Logic Programming,* Salt Lake City, pp.:168-179, 1986.

[GR1987] D. Goldberg and J. Richardson. Genetic Algorithms with sharing for multimodal function optimization. In *Proceedings of the 2nd Int'l conf. On Genetic Algorithms,* pp.:41-49, Morgan Kaufmann Publishers, 1987.

[Gre1969] Cordell Green. Application of Theorem Proving to Problem Solving. *IJCAI*, 1969.

[G+2008] F. Gomez et al. Accelerated Neural Evolution through Cooperatively Coevolved Synapses. *In Journal of Machine Learning Research* 9, pp.:937-965, 2008.

[Hal2005] J. Halpern. Reasoning about uncertainty. *MIT Press*, 2005.

[Hal1986] J. Halpern. Reasoning about knowledge: na Overview. In *theoretical Aspects of Reasoning about knowledge. Proceedings of the 1986 Conference*, Morgan Kaufman, 1986.

[Har1993] G. Hardy. A Mathematician's Apology. *Reprinted with a foreword by C. P. Snow. New York: Cambridge University Press*, pp.:34, 1993.

[Har2002] G. Harman. Internal Critique: A Logic is not a theory of reasoning and a theory of reasoning is not a logic. In D.M. Gabbay, R.H. Johnson, H.J. Ohlbach, and J. Woods, eds., *Handbook of the Logic of Argument and Inference: The Turn Towards the Practical, Volume 1 in Studies in Logic and Practical Reasoning. Amsterdam: Elsevier Science B.V.,pp. 171-86, 2002.*

[Has1992] S. Hashem. Sensitivity analysis for Feedforward artificial neural networks with differentiable activation functions. *IJCNN1992, vol.*1, pp.419-424, Baltimore, 1992.

[Hay1998] S. Haykin. Neural Network: A comprehensive Foundation. *Mac William, New York*, 1998.

[Hay2008] S. Haykin. Neural Networks and Learning Machines. *Prentice Hall, Hardcover*, 3^{rd} edition, Published 2008.

[HA1988] Carl Hewitt e Gul Agha. Guarded Horn Clause Language: are they deductive and logical?. In Procs of the *International Conference on Fifth Generation Computer Systems*, 1988.

[Hei1971] J. van Heijenoort, editor, From Frege to Godel: A source book in Mathematical Logic. *Harvard University Press,* pp.:1879-1931, 1971.

[Hek1998] J. Hekanaho. An Evolutionary Approach to Concept Learning. *PhD thesis, Department of Computer Science, ABO AKADEMI University*, 1998.

[Hew1969] Carl Hewitt. Planner: A language for Proving theorems in robots. *IJCAI*, 1969.

[Hew2008] Carl Hewitt. History of Logic Programming: What went wrong, What was done about it, and What it might mean for the future. *Stanford CSLI CogLunch*, 2008.

[HHS2004] Pascal Hitzler, S. Holldobler and A. Seda. Logic Programs and Connectionist Networks. *In Journal Applications of Logic.* vol.2, pp.:245-272, 2004.

[HLB2004] A. Hommersom, A., P. J. F. Lucas and M. Balser. Meta-level verification of the quality of medical guidelines using interactive theorem proving. *Lecture Notes in Computer Science,* pp.:654-666, 2004.

[HLB2008] Hommersom, A., P. J. F. Lucas and P. van Bommel: Checking the quality of clinical guidelines using automated reasoning tools. *Theory and Practice of Logic Programming* 8(5-6), pp.:611-641, 2008.

[Hol1962] J. Holland. Outline for logical theory of adaptive systems. *JACM,* vol.9, pp.:297-314, 1962.

[Hop1985] J.J. Hopfield. Neural computation of decisions in optimization problems. *Biol. Cybern., 52*, pp.:141-152, 1985.

[HP2007] B. Hammer e P. Hitzler (eds). Perspectives of Neural Symbolic Integration. *Studies in Computational Intelligence, vol. 77, Springer-Verlag*, 2007.

[Hus1994] Don Hustad. Do we need the closed-world assumption in knowledge representation?". In *Working Notes of the KI'94 Workshop*, pp.:24-26, Baader, Buchheit, Jeusfeld, Nutt (eds.), Saarbrücken, Germany, 1994.

[HW2009] Patricia M. Hill, David Scott Warren (Eds.). Logic Programming. In *the 25th International Conference, ICLP 2009, Pasadena, CA, USA*, July 14-17, 2009. *Proceedings. Lecture Notes in Computer Science* 5649 Springer-Verlag, 2009.

[HX2008] Hua, Z., B. Gong and X. Xu. A DS-AHP approach for multi-attribute decision making problem with incomplete information. *Expert Systems with Applications* 34(3), pp.:2221-2227, 2008.

[IM$^+$1993] Hideya Iiawahara, H. Murakoshi, N. Furiakubo and S. Ishijiina. A Neural Network Approach to Inference Mechanism for Logic Programming Language. In *Proceedings of 1993 International Joint Conference on Neural Networks,* pp.:167-170, 1993.

[Ing1993] L. Ingber. Simulated Annealing: Practice Versus Theory. Math. Comput. Modelling 18, pp.:29-57, 1993.

[IS1994] Katsumi Inoue and Chiaki Sakama. On positive occurrences of negation as failure. In *Proc. Fourth Int'l Conf. On Principles of Knowledge Representation and Reasoning,* pp.:293-304, 1994.

[Jec1978] Thomas J. Jech. Set theory. *Academic Press*, 1978.

[Jec2006] Thomas J. Jech. Introduction to set theory. *Springer, 3rd edition*, 2006.

[JGB1992] W. Jakob, M. Georges-Shleuter and C. Blume. Application of Genetic Algorithms to task planning and learning. In *Manner, R. And Manderick, B. Eds, Parallel Problem Solving from Nature, 2nd Workshop, Lecture Notes in Computer Science,* pp.: 291-300, Amsterdam, 1992.

[JJ1994] R. Josephson, and G. Josephson, Susan,.Abductive Inference: Computation, Philosophy, Technology. *In Cambridge University Press, Cambridge, UK*, 1994.

[Jon2006] Jong, K. Evolutionary Computation a Unified Approach. *MIT Press*, 2006.

[JR1991] P. N. Johnson-Laird and Ruth M. J. Byrne. Deduction. In *Lawrence Erlbaum Publishing, Hove* 1991.

[Kas1998] N. Kasabov. Evolving fuzzy neural networks—algorithms, applications and biological motivation. In: *T. Yamakawa, G. Matsumoto (Eds.), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, Singapore*, pp.:271–274, 1998.

[KC2008] K. Kim and S. Cho. Evolutionary ensemble of diverse artificial neural networks using specification. *Neurocomputing vol. 71,* pp.:1604-1618, 2008.

[Kei2002] M. Keijzer, Scientific Discovery using Genetic Programming. *PhD thesis, Danish Technical University, Lyngby, Denmark*, 2002.

[Ken2006] J. Kennedy. Search of the essential particle swarm. *In CEC 2006. Proceedings of the IEEE Conf. on Evolutionary Computation, IEEE Computer Society, Los Alamitos*, 2006.

[KKT1992] Antonis Kakas, Robert Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6): pp.:719-770, 1992.

[KK1998] A. Kakas, R. Kowalski, F. Toni, The role of abduction in logic programming. *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol.5, pp.:235-324, 1998.

[KM2008] E. Kolman e M. Margaliot. Extracting Symbolic Knowledge from recurrent neural networks – a fuzzy logic approach. *In Press,* 2008.

[KM2008] Peep Küngas, Mihhail Matskin. Symbolic negotiation: Partial deduction for linear logic with coalition formation. *Web Intelligence and Agent Systems 6(2):* pp. 193-215, 2008.

[Koh2001] T. Kohonen. Self-Organizing Maps. Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York. Third Extended Edition, 2001.

[Kok2001] Kókai. Gelog – A System Combining Genetic Algorithm with Inductive Logic Programming. *In: Reush, B. (Hvs). Proc. Of the Intl' Conf. In Computational Intelligence, Springer Verlag*, pp.:326-345, 2001.

[Kow1973] R. Kowalski. Predicate Logic as Programming Language. *Memo 70, Department of Artificial Intelligence, Edinburgh University*, 1973.

[Kow1984] R. Kowalski. The Relationship between Logic Programming and Logic Specification. *In Phil. Trans. R. Soc. Lond. A,* vol 312, pp. 345-361, 1984

[Kow1986] R. Kowalski. Predicate Logic as Programming Language, In Computers for Artificial Intelligence Applications. In. Wah, B. and Li, G.-J. (eds), *IEEE Computer Society Press, Los Angeles*, pp.:68-73, 1986.

[Kow1995] R. Kowalski. Logic without Model Theory. *In: What is a logical system?, D. Gabbay ed., Oxford University Press*, 1995.

[Kow1996] R. Kowalski. The Limitations of Logic. *In Proceedings of ACM Computer Science Conference, 1986,* pp 7-13. Revised version in Proceedings of SEAS, Heidelberg, pp.:1-13, 1996.

[Kow2002] R. Kowalski. Directions for Logic Programming. *In Computational Logic: Logic Programming* and Beyond (eds. A C Kakas and F Sadri) Springer, pp.:26-32, 2002.

[Kow2006] R. Kowalski. The logical way to be artificially intelligent. In: Toni, F., Torroni, O. (eds), *Proceedings of the CLIMA VI.LNCS (LNAI*), pp.:1-22, Springer, Heidelberg, 2006.

[Kow2008] R. Kowalski. From Mathematical Logic, to Natural Language. In *Artificial Intelligence, and Human Thinking Andrzej Mostowski and Foundational Studies (Eds. A. Ehrenfeucht, V.W. Marek and M. Srebrny) IOS Press,* 2008.

[Koz1992] J. Koza. Genetic Programming. *MIT Press, Cambridge, MA,* 1992.

[Koz1994] J. Koza. Genetic Programming II. *MIT Press, Cambridge, MA*, 1994.

[KR2010] Boris Kovalerchuck e G. Resconi. Agent-based uncertainty Logic Network. *In proceedings of the WCCI 2010 – IEEE World Conf. on Computational Intelligence*, Barcelona, Spain, pp.:596-603, 2010.

[KS1995] Martin Kummer and Frank Stephan. Recursion theoretic properties of frequency computation and bounded queries. *Information and Computation,* pp.:120:59–77, 1995.

[Kui1994] B. Kuipers. Qualitative reasoning: modeling and simulation with incomplete knowledge. *MIT press*, 1994.

[LAP2002] J. A. Leite, J. J. Alferes, L. M. Pereira. MINERVA – A Dynamic Logic Programming Agent Architecture. *In J. J. Meyer, M. Tambe (eds.), Intelligent Agents VIII*, pp.:141-157, Springer-Verlag, LNAI 2333, 2002.

[LC$^+$2008] Luis Lima, R. Costa, P. Novais, et al. Quality of Information in the Context of Ambient Assisted Living. In *Advances in Soft Computing, Springer-Verlag.* vol. 50: pp.:624-633, 2008.

[LD1995] N. Lavrac e S. Dezeroski. Inductive Logic Programming. *Techniques and Applications, Ellis Horwood,* 1995.

[LDM2009] N. Lima, A. Neto e J. Melo. Creating na Esemble of Diverse Support Vector Machines using Adaboost. *In proc. On Intl. Joint Conf on Neural Networks*, 2009.

[Lif1996] V. Lifschitz. Foundations of logic programming. In: G. Brewka, Editor, Principles of Knowledge Representation, CSLI Publications, pp.:69–128, 1996.

[Lif2002] V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence 138,* pp. 39-54, 2002.

[Lig1997] A. Ligeza. Logical Analysis of Completeness of Rule-Based Systems with Dual Resolution. In *Proceedings of the European Symposium on the Verification and Validation of Knowledge Based Systems* (EuroVAV'97), pp.:155-165, Leuven, Bélgica, 1997.

[LJ2009] Lukoseevicius, M. e Jaeger, H. Reservoir computing approaches to recurrent neural network training. In *Proceedings of Computer Science Review, Vol. 3, No. 3. pp.:127-149, 2009.*

[LK2005] Lindblad, T. e Kinser, J. Image Processing Using PulseCoupled Neural Networks. SpringerVerlag, 2005.

[LL1959] C. Lewis and C. Langford. Symbolic Logic 2^{nd} ed., *Dover, New York*, 1959.

[LN1994] Donald Loveland e G. Nadathur. Proof Procedures for Logic Programming. Volume 5 of Handbook of Logic in AI and Logic Programming, Oxford University Press, pp.:163-234, 1994.

[LN$^+$2009] Luis Lima, P. Novais, J. Bulas Cruz. A Process Model For Group Decision Making With Quality Evaluation. In Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing and Ambient Assisted Living, Omatiu S., et al. (Eds.) LNCS 5518, Springer-Verlag, pp.:566-573, 2009.

[Loy2004] Loyer, Y. Spyratos and Stamate, D. Hypothesis-based semantics of logic programs in multivalued logics. *ACM Transactions on Computational Logic*, 15(3), 2004.

[LP$^+$1998] E. Lamma, L. M. Pereira, F. Riguzzi. Learning with extended logic programs. In *KR'98 Ws. On Non-Monotonic Reasoning (LP Track)*, pp.:99-108, Trento, Italy, June 1998.

[LP2006] Gonçalo Lopes, L. M. Pereira. Prospective Programming with ACORDA. In: Empirically Successful Computerized Reasoning (ESCoR'06) workshop at The 3^{rd} International Joint Conference on Automated Reasoning (IJCAR'06), Seattle, USA, August 21, 2006.

[LS1993] V. Lifschitz and G. Schwarz. Extended logic programs as autoepistemic theories. *In Proc. Second Internat. Workshop on Logic Programming and Non-Monotonic Reasoning, Lisbon*, pp.:101–114, 1993.

[LS1997]] Lakshmanan, V. and Sadri, F. On a theory of probabilistic deductive databases. *Journal on Theory and Practice of Logic Programming*, 1997.

[LT1994] Jimmy Lee e Wai Tam. Towards the integration of Artificial Neural Networks and Constraint Logic Programming. *The Chinese University of Hong Kong, Technical Report*, 1994.

[LT1995] Jimmy Lee e W. Tam. A Framework for Integrating Artificial Neural Networks and Logic Programming. *International Journal on Artificial Intelligence Tools, 4(1&2)*, pp.:3-32, World Scientific, June, 1995.

[Luc2003] Lucas, P. Quality checking of medical guidelines through logical abduction. *Proc. Of AI-2003 20*: pp.:309-321, 2003.

[LW1992] Vladimir Lifschitz and Thomas Woo. Answer sets in general nonmonotic reasoning (preliminary report). In Bernhard Nebel, Charles Rich, and William Swartout, editors, Proc. Third Int'l Conf. On Principles of Knowledge Representation and Reasoning, pp.:603-614, 1992.

[MA$^+$ 2006^a] José Machado, F. Andrade, J. Neves, P. Novais and C. Analide. Formal Models in Web Based Contracting. The 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, (WI-IAT 2006 Workshop Proceedings), Hong Kong, pp.:548-551, 2006.

[MA+2006b] José Machado, A. Abelha, M. Santos and J. Neves. Multi-agent based Problem Solving in Medical Decision Support Systems. In Proceedings of the 2^{nd} International Conference on Knowledge Engineering and Decision Support, Knowledge and Decision Technologies, Vale Z., Ramos C. And Faria L. (eds), Lisbon, Portugal, 2006.

[MA+2008] José Machado, A. Abelha, P. Novais, J. Neves and J. Neves. Improving Patient Assistance and Medical Practices through Intelligent Agents. In *Procs. of the the 5^{th} Workshop on Agents Applied in Health Care, in the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 08),* Cascais, Portugal, 2008.

[MA+2010] José Machado, A. Abelha, P. Novais, J. Neves and J. Neves. Quality of service in healthcare units. Int. *J. Computer Aided Engineering and Technology*, vol. 2, No. 4, pp.:436-449, 2010.

[Mar2000] David Marker. Model Theory, Algebra and Geometry. *MSRI Publications*, vol.39, 2000.

[Mag2001] Lorenzo Magnani. Abduction, Reason, and Science: Processes of Discovery and Explanation. *New York, Boston, Dordrecht, London, Moscow: Kluwer Academic/Plenum Publishers*, pp.:1-205, 2001.

[McC1973] John McCarthy. Review of Artificial Intelligence: A General Survey. *Artificial Intelligence: a paper Symposium, UK, Science Research Council,* 1973.

[McD1982] McDermott D. Non-Monotic Logic II: Non-Monotic Modal Theories. In *Journal of the Association for Computing Machinery 29,* pp.:33-57, 1982.

[McK1941] Richard McKeon editor. The Basic Works of Aristotle. Randons House, *XXXIX,* 1941.

[MD1980] D. McDermott and J. Doyle. Non-Monotic Logic I, in Special Issue on Non-Monotic Logics. *In Brobow D. (ed)*, 1980.

[Men1997] E. Mendelson. Introduction to Mathematical Logic. *Springer-Verlag, 4^{th} edition*, 1997.

[Mit1982] T. Mitchell. Generalization as search. *Artificial Intelligence*, vol.18, pp.:202-226, 1982.

[MM+2009] José Machado, M. Miranda, G. Pontes, D. Santos, M. Santos, A. Abelha and J. Neves. Intelligent Agents and Medical Moral Dilemmas. In *Proceedings of the 8th WSEAS International Conference on Applied Computer & Applied Computational Science (ACACOS '09),* Hangzhou, China, 2009.

[Moo1985] Robert Moore. Semantical considerations on nonmonotic logic. *Artificial Intelligence, 25 (1)* pp.:75-94, 1985.

[Moy2000] S. Moyle: An Investigation into theory completion on techniques in Inductive Logic Programming. *Phd thesis, Oxford University, Computing Laboratory, University of Oxford*, 2000.

[MP1943] W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in neurons activity. *Bulletin of Mathematical Biophysics*, vol.5, pp.:115-133, 1943.

[MP1969] Minsky and Papert. Perceptrons, an introduction to computational geometry. *MIT press, expanded edition*, 1969.

[MR1986] J. McClelland, D. Rumelhart and the PDP Research Group. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. *vol. 1: Foundations. Cambridge, MA: MIT Press*, 1986.

[MR1994] S. Muggleton and D. Raedt. Inductive Logic Programming: Theory and methods. *Journal of Logic Programming,* 19-20, pp.669-679, 1994.

[MS+2007] Goreti Marreiros. R. Santos, P. Novais, J. Machado et al. Argumentation-based Decision Making. In: J. M. Neves, M. F. Santos, J. M. Machado (eds.), Progress in Artificial Intelligence, Procs. 13th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07), pp.:73-86, Springer LNAI 4784, Guimarães, Portugal, December 2007.

[MUW2002] H. Molina-Garcia, J. Ullman and J. Widom. Database systems the complete book. *Prentice hall eds,* 2002.

[NA+2010] Paulo Novais, F. Andrade, J. Machado and J. Neves. Agents, Trust and Contracts. In Information Communication Technology Law, Protection and Access Rights: Global Approaches and Issues, Portela I., Cunha M. (Eds), IGI Global, 2010.

[Nea1998] R. Neal, Assessing relevance determination methods using Delve generalization in neural networks and Machine Learning. *C. Bishop, Ed. Springer, Berlin, 1998.* Available: www.cs.toronto.edu/_delve

[Nev1984] José Neves. A Logic Interpreter to Handle Time and Negation in Logic Data Bases. In *Proceedings of ACM'84, The Fifth Generation Challenge*, pp.:50-54, 1984.

[Nit2007] G. Nitschke. Neuro-evolution methods for designing emergent specialization. In Proceedings of the nith European Conference on Artificial Life, Springer, Lisbon, pp.:1120-1130, 2007.

[Nit2008] G. Nitschke. Neuro-evolution for emergent specialization in collective behaviour systems. *PhD Thesis, Computer-Science Department, Vrij Universiteit*, Amesterdam, 2008.

[NM+1997] José Neves, J. Machado, C. Analide, P. Novais and A. Abelha. Extended Logic Programming Applied to the Specification of Multi-Agent Systems and their Computing Environment. *In Proceedings of ICIPS'97, Special Session on Intelligent Networked Agent Systems*, Beijing, China, 1997.

[NM+2007] José Neves, J. Machado, C. Analide, A. Abelha, and L. Brito. The Halt Condition of the Genetic Programming. In J. M. Neves, M. F. Santos, J. M. Machado (eds.), Progress in Artificial Intelligence, Procs. 13^{th} Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07), pp. 73-86, Springer LNAI 4784, Guimarães, Portugal, December 2007.

[NM2000] V. Nilsson, J. Malusynski. Logic, Programming and Prolog. *JohnWiley & Sons*, 2000.

[NS+2010] Paulo Novais, M. Salazar, J. Ribeiro, C. Analide, J. Neves. Decision Making and Quality-of-Information. *In Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010),* Corchado E., Novais P., Analide C., Sedano J., (Eds.) Springer – Series Advances in Intelligent and Soft Computing, vol. 73, ISBN 978-3-642-13160-8, pp.:187-195, (International Workshop on Soft Computing Models in Industrial Applications, Guimarães, 2010.

[[NS+2011] Neves, J., Reus Salini, J. Ribeiro, Machado J., Abelha, A., Novais, P., Analide, C. and Fernndez-Delgado, M. Evolutionary Intelligence in Asphalt Pavement Modelling. In Procs. of the Journal in Progress on Artificial Intelligence, Herrera F. (ed), Springer-verlag, 2011.

[NY2002] W. Ng e D. Yeung. Input dimensionality reduction for radial basis neural network classification problems using sensivity measure. *ICMLC2002,* pp.:2214-2219, Beijin, 2002.

[OC+1995] T. Osborn, A. Charif, R. Lamas e E. Dubossarsky. Genetic Logic Programming. *In 1995 IEEE International Conference on Evolutionary Computation ,* vol. 2, pp.:728 – 732, Perth, Australia, 1995.

[OO1994] O'Reilly and F. Oppacher. The troubling aspects of analyzing block hypothesis for genetic programming. In *L. D. Whitley and M. D. Vose, editors, Foundations of Genetic Algorithms 3*, pp.:73-88, USA, 1994.

[ORW2005] Una-May O'Reilly, T. Yu, R. Riolo, B. Worzel. Genetic Programming Theory and Practice II. *Springer Science – Business Media*, Inc, 2005.

[PA2009] Luis M. Pereira, H. T. Anh. Evolution Prospection. *In K. Nakamatsu (ed.), Procs. First KES Intl. Symposium on Intelligent Decision Technologies (KES-IDT'09)*, Springer Verlag book in Engineering Series, Himeji, Japan, April 2009.

[Pal2008] M. Palomino. Predicate Logic. Encyclopedia of Sciences and Religions, Wiley Encyclopedia of Computer Science and Engineering, 2008.

[PCA1992] Luis M. Pereira Luis Moniz, Caires Luis and J. Alferes. SLWV – *A* Theorem Prover for Logic Programming. In *ELP'92, LNCS 6*60, Bologna, 1992.

[PDL2009] Luis M. Pereira, P. Dell'Acqua, G. Lopes. On Preferring and Inspecting Abductive Models. Invited paper in: *A.Gill, T. Swift (eds.), Procs. 11th Intl. Symp. Practical Aspects of Declarative Languages* (PADL'09), pp.:1-15, Springer LNCS 5418, Savannah, Georgia, USA, January 2009.

[Pei1992] Charles Peirce. The essential Peirce. *Selected Philosophical writings, vol 2*, pp.:1893-1913, the Peirce Edition Project, ed, Blooninghton. Indiana University Press, 1992.

[Per2002] D. Perkins. Standard Logic as a model of reasoning: the empirical critique. In *Studies in Logic and Practical Reasoning,* Volume 1, D.M. Gabbay et al., editors, Elsevier Science B.V., 2002.

[Pie2008] Brigitte Pientka. A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions. *In Principles of Programming Languages*, POPL, 2008.

[Pin1991] G. Pinkas. Propositional non-monotic reasoning and inconsistency in symmetrical neural networks. *In Proceedings of the IJCAA*, pp.:525-530, 1991.

[Pir1976] C. Piron. Foundations of Quantum Physics. *W. A. Benjamin*, 1976.

[PK2005] E. Paz, C. Kamath. An Empirical Comparison of Combinations of Evolutionary Algorithms and Neural Networks for Classification Problems. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 2005.

[PL2007] Luis M. Pereira, Gonçalo Lopes. Prospective Logic Agents. In J. M. Neves, M. F. Santos, J. M. Machado (eds.), Progress in Artificial Intelligence, Procs. 13th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07), pp.:73-86, Springer LNAI 4784, Guimarães, Portugal, December 2007.

[PLM2008] Riccardo Poli, Langdon, W., McPhee N., Field guide to Genetic Programmin, *Lulu Enterprises, UK Ltd*, 2008.

[Poh1989] Wolfram Pohlers. Proof Theory: An Introduction. In *lecture notes in Mathematics*, 1989.

[Poo1990] D. Poole. A methodology for using a default and abductive reasoning system. *International Journal of Intelligent Systems* 5(5): pp.:521-548, 1990.

[PP1994] Luis. M. Pereira, J. N. Aparicio and J. J. Alferes. Logic Programming for Nonmonotonic Reasoning. In *Knowledge Representation and Reasoning Under Uncertainty*, LNAI 808, M. Masuch and L. Polos (eds.), Springer-Verlag, pp.:107-122. Springer-Verlag, 1994.

[PP2005] Luis M. Pereira, A. M. Pinto. Revised Stable Models – a Semantics for Logic Programs. In C. Bento, A. Cardoso, G. Dias (eds.), Progress in Artificial Intelligence, Procs. 12^{th} Portuguese Intl.Conf. on Artificial Intelligence (EPIA'05), pp.:29-42, Springer, LNAI 3808, Covilhã, Portugal, December 2005.

[PR2009] Luis M. Pereira, Carroline Ramli. Modelling Probabilistic Causation in Decision Making. *In K. Nakamatsu (ed.), Procs. First KES Intl. Symposium on Intelligent Decision Technologies* (KES-IDT'09), Springer Verlag book in Engineering Series, Himeji, Japan, April 2009.

[Prz1991] Teodor Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing, 9:* pp.:401-424, 1991.

[PS2007] Luis M. Pereira, A. Saptawijaya. Modelling Morality with Prospective Logic. In J. M. Neves, M. F. Santos, J. M. Machado (eds.), Progress in Artificial Intelligence, Procs. 13^{th} Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07), pp.:99-111, Springer LNAI 4874, Guimarães, Portugal, December 2007.

[PS2009] Adam Poswolsky e C. Schurman. System Description:Delphin – A Functional Programming Language for Deductive Systems Electronic Notes. In Theoretical Computer Science, ENTCS archive, vol.228, pp.:113-120, 2009.

[PW1990] David Pearce and Gerd Wagner. Reasoning with negative information I: Strong negation in logic programs. *Acta Philosophica Fennica*, 49, 1990.

[QS+2009] M. Quinn, L. Smith, G. Mayley e P. Husbans. Evolving teamwork and role-allocation with real robots. In *Proceedings of the 8th International Conference on Artificial Life, MIT Press, Cambridge*, pp.:302-311, 2009.

[Rao2000] S. Rao. Engineering Optimization – Theory and Practice, *Third Edition, New Age International Limited, New Delhi*, 2000.

[Ray2005] Oliver Ray. Hybrid Abductive-Inductive Learning. Ph.D. dissertation, Department of Computing, Imperial College London, UK, 2005.

[RCN2003] Miguel Rocha, P. Cortez and J. Neves. Evolutionary Neural Network Learning. In *F. Pires and S. Abreu (Eds.), Progress in Artificial Intelligence, 11th Portuguese Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, pp.:24-28, Beja, Portugal, 2003.

[RCN2004] Miguel Rocha, P. Cortez and J. Neves. Evolutionary Neural Network Learning Algorithms for Changing Environments. In *WSEAS Transactions on Systems, WSEAS Press,* 3 (2): pp.:596-601, 2004.

[RD2005] J. Rabunal and J. Dorrado. Artificial Neural Networks in Real-life Applications, *Idea Group*, 2005.

[Rei1978] Raymond Reiter. On closed world data bases. In *Herve Gallaire and Jack Minker editors, Logic and Data Bases,* pp.119-140. Plenum Press, New York, 1978.

[Rei1980] R. Reiter. A Logic for Default Reasoning. In *Special Issue on Non-Monotonic Logics, Brobow D. (ed),* 1980.

[Res2008] Greg Restall. Assertion, Denial and Non-Classical Theories. *In Proceedings of the Fourth World Congress of Paraconsistency, Melbourne, Australia*, 2008.

[RJJ1996] V. Rooij, H. Jain e R. Johnson. Network training using Genetic Algorithms. In series in machine perceptron and Artificial Intelligence, vol. 26, H. Bunke & P. Wang (eds), Singapore, World Scientific Publishing Co, 1996.

[RM+2002] A. Russo, R. Miller, B. Nuseibeh, and J. Kramer. An abductive approach for Analyzing event-based requirements specifications. In *Proc. ICLP-2002. Springer-Verlag*, 2002.

[RM+2010a] Jorge Ribeiro, J. Machado, A. Abelha, M. Fernandez-Delgado and J. Neves. Integrating Incomplete Information into the Relational Data Model. In *Proceedings of the The 2010 International Conference of Computational Intelligence and Intelligent Systems- World Congress on Engineering,* vol. 1, pp.:57-62, ISBN:978-988-17012-9-9, London, 2010.

[RM+2010b] Jorge Ribeiro, J. Machado, A. Abelha, A. Marques and J.Neves. The Inference Process with Quality Evaluation in Healthcare Environments. In *Proceedings of the 9^{th} IEEE/ACIS International Conference on Computer and Information Science (ICIS 2010), August 18-20, 2010, Kaminoyama (Yamagata),* pp.:183-188, ISBN: 978-0-7695-4147-1, Japan, 2010.

[RM+2010c] Jorge Ribeiro, J. Machado, A. Abelha and J. Neves. Handling Incomplete Information in an Evolutionary Environment". In *Proceedings of the IEEE CEC 2010 - World Congress on Computational Intelligence*, Barcelona, pp.:2786-2793, ISBN:978-1-4244-6910-9, 2010.

[RN+2009] Jorge Ribeiro, P. Novais, J. Neves J., M Fernandez-Delgado. Quality of the Information: The application in the winification process in wine production, in Proceedings of the 2009 European Computing Conference of the World Scientific and Engineering Academy and Society, Tbilisi, Georgia, June 26-28, pp.:62-70, ISBN:978-960-474-088-8, 2009.

[RN1995] S. Russell and P. Norvig. Artificial Intelligence: a modern approach. *Prentice Hall: New Jersey, USA*, 1995.

[Rob1965] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. CACM. 1965.

[Rob2006] Robert Hanna. Rationality and Logic, MIT Press, 2006.

[Roj1996] R. Rojas. Neural Networks - A Systematic Introduction. *Springer Verlag, Berlin, Germany,* 1996.

[Ros2001] Brian Ross. Logic-based Genetic Programming with Definite Clause Translation Grammars. *New Generation Computing*, v.19, n.4, pp.:313-337, 2001.

[RS1965] I. Rechemerg. Cybernetic solution path of an experimental problem. In *Library Translation 1122. Farnborough: Royal Aircraft Establishment*, 1965.

[Rud1994] G. Rudolph. Convergence Analysis of Canonical Genetic Algorithms. In *IEEE Transactions on Neural Networks, special issue on Evolutionary Computation, 5(1)*: pp. 96-101, 1994.

[Saa2001] T. Saaty. The Analytic Network Process. *RWS Publications*, 2001.

[SA2000] H. Seridi and H. Akdag. A qualitative approach for processing uncertainty. In *intelligent and information systems 20*: pp.46–57, 2000.

[SA2008] Saratha Sathasivam and Wan Abdullah. Logic Learning in the Hopfield Networks, *Modern Applied Science*, 2(3), pp.:57-62, 2008.

[Sat2006] Daniel Stamate. Assumprion based multi-valued semantics for extended logic programs. In *proceddings of the 36^{th} International Symposium on Multiple-valued Logic*, 2006.

[Sat2007] Saratha Sathasivam. Logic Mining in Neural Networks. *PhD Thesis, University Malasya*, 2007.

[Sat2009] Saratha Sathasivam. Enhancing Logic Programming Performance in Recurrent Hopfield Network. In *European Journal of Scientific Research, vol.37 No.1*, pp.:6-11, 2009.

[Sha1992] G. Shafer. The Dempster-Shafer theory. In *Encyclopedia of Artificial Intelligence, Second Edition. S. C. Shapiro*, Wiley, 1992.

[Sha1989] Ehud Shapiro. The family of Concurrent logic programming languages. *ACM Computing Surveys*, 1989.

[Sha2000] M. Shanahan. An abductive event calculus planner. *Journal of Logic Programming*, 44, pp.:207-239, 2000.

[She1991] F. Sheridan. A Survey of techniques for inference under uncertainty. *Artificial Intelligent Review* 5(1):89, 1991.

[Sic2010] Sicstus Prolog Website, http://www.sics.se/isl/sicstuswww/site/index.html (accessed 06-10- 2010).

[Sim2000] Stephen G. Simpson. Logic and mathematics. In *The Examined Life, Readings from Western Philosophy from Plato to Kant, edited by S. Rosen, Random House,* XXVIII + 628 pages, pp.:577-605, 2000.

[SKS2007] Nils T. Siebel, J. Krause, G. Sommer. Efficient learning of neural networks with evolutionary algorithms. In *Proceedings of the 29th DAGM conference on Pattern recognition*, pp.:466-475, 2007.

[SM2010] Motohiro Saiki e S. Matsuda. Evolutionary Neural Networks Model of Universal Grammars. *In proceedings of the WCCI 2010 IEEE World Congress on Computational Intelligence*, July, 18-23, 2010 – CCIB, Barcelona, Spain, pp.:1857-1861, 2010.

[Spe1990] William Spears e K De Jong. Using Neural Networks and Genetic Algoriuthms as heuristics for NP-Complete Problems. *Master's Thesis, George Mason University, Fairfax, VA*, 1990.

[SS1994] Leon Sterling e Ehud Shapiro. The Art of Prolog Advanced Programming Techniques (Logic Programming*), 2nd ed,The MIT Press*, 1994.

[Str2006] E. Straszecka. Combining uncertainty and imprecision in models of medical diagnosis. *Information Sciences,176(2),* pp.:3026-3059, 2006.

[Sub2001] V. Subrahmanian. Probabilistic databases and logic programming. In *Proc. Of the 17th International Conference of Logic Programming*, 2001.

[SWE1992] J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *International Workshop on Combinations of Genetic Algorithms and Neural Networks*. 1992, pp. 1–37, IEEE Computer Society Press.

[Swi2010] SWI Prolog Organization, SWI Prolog Website, http://www.swi-prolog.org/ (accessed 06-10- 2010).

[Tar1954] Alfred Tarski. Contributions to the theory of models. *Iindagations mathematicae vol 16*, pp.:572-581, 1954.

[TCM1998] L. Tang, M. Califf and R. Mooney. An experimental comparison of genetic programming and inductive logic programming on learning recursive list functions. *In 8^{th} Int. Workshop on Inductive Logic Programming*, 1998.

[Tel1996] A. Teller. Evolving programmers: the co-evaluation of intelligent recombination operators. In *K. Kinnear and P. Angeline, editors, Advances in Genetic Programming 2. MIT*, 1996.

[TF1995] M. Turner, G. Favconnier. Conceptual Integration and Formal Expression. Johnson, M.: Journal of Metaphor and Symbolic Activity 10(3), 1995.

[TG1993] B. Traylor and M. Gelfond. Representing Null Values in Logic Programming". In: *Proceedings of the International Logic Symposium* (ILPS'93). Vancouver, British Columbia, Canada, 1993.

[TK+2004] Y. Takeda, H. Kawahara, et al. A neural network approach to inference mechanism for logic programming languages. *Systems and Computers in Japan archive*, vol. 35, pp.:63-70, 2004.

[TS1996] A S Troelstra, H Schwichtenberg. Basic Proof Theory. *Cambridge Tracts in Theoretical Computer Science 43. Cambridge University Press*. Cambridge 1996.

[Tve1997] A. Tveit. Genetic Inductive Logic Programming. *Diplomarbeit, Novwegian University of science and technology, NTNU*, 1997.

[Vic2006] J. Vickers. The problem of Induction. *Standford Encyclopedia of Philosophy*, 2006.

[VKS1990] Allen Van Gelder, Kenneth Ross, and John Schlipf. The well-founded semantics for general logic programs. *Journal of ACM,* pp.:620-650, 1990.

[Wei2005] Haikun Wei. Theory and Method of Neural Network Structure Design. *National Defense Industry Press*, 2005.

[Win1971] Terry Winograd. Procedures as a representation for Data in a Computer Program for Understanding Natural Languages. *MIT, AI Tech. Report-235*, 1971.

[WK1989] D. Weld e J De Kleer. Readings in qualitative reasoning about physical systems. *Morgan Kaufman Publishers Inc. San Francisco, CA, USA*, 1989.

[WL1990] Widrow and M. Lehr. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation, *Proc. IEEE*, vol. 78 (9), pp.1415– 1442, 1990.

[WL1995a] M. Wong and K. Leung. Genetic Programming System: Inducting Logic Programs with genetic algorithms. *IEEE Expert/Intelligente Systems and their a application, 9, no. 5*, pp.68-76., 1995.

[WL1995b] M. Wong and K. Leung. Combining Genetic Programming and Inductive Logic Programming using Logic Grammars. In *Proceedings of the 1995 IEEE International Conference on Evolutionary Computing*, pp.:733-736., 1995.

[WL1997] M. Wong and K. Leung. Evolutionary Program Induction Directed by Logic Grammars. *In Evolutionary Computation, vol. 5,* pp.:143-180, 1997.

[WN1994] P. Whighan and R. Mckay. Genetic Programming and Inductive logic. *Technical report, University College pf new South Walles, Australia*, 1994.

[Won2001] M. Wong. A Flexible Knowledge Discovery System using Genetic Programming and Logic Grammars. *Decision Support Systems, 31*, pp.:405-428., 2001.

[Woo1992] M. Wooldridge. The Logical Modelling of Computational Multi-Agent Systems. *PhD Thesis, Faculty of Technology, University of Manchester*, August 1992.

[WRL1994] B. Widrow, D. Rumelhart and M. Lehr, Neural Networks: applications in industry, business and science. in *ACM, 37(3),* pp.:93-105, 1994.

[Xsb1999] XSB-Prolog. The XSB logic programming system, version 2.0, 1999. Available on: http://www.cs.sunysb.edu/ sbprolog (accessed in 06.10.2010)

[Yao1999] X. Yao. Evolving Artificial Neural Networks. In *Proc. of the IEEE, 87(9)*, pp.:1423-1447, 1999.

[Zad2001] L. Zadeh. Fuzzy Logic. *The MIT Encyclopedia of the Cognitive Sciences. R. A. Wilson and F. C. Keil, MIT Press*, 2001.

[ZY2001] X. Zeng e D. Yeung. Sensivity analysis of multilayer perceptron to input and weight perturbations. *In IEEE Trans. On Neural Networks vol. 12, no. 6, pp.:1358-1366, 2001.*