**Pontifícia Universidade Católica do Rio Grande do Sul**
**Faculdade de Informática**
**Pós-Graduação em Ciência da Computação**

# Implementação e Uso de Algoritmos de Conversão Neuro-Simbólica

Leandro Ávila de Ávila

**Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação**

Orientador: José Carlos Bins Filho

Porto Alegre, outubro de 2004.

**Agradecimentos**

Agradeço primeiramente a Deus. Sem Ele, nada seria possível. Agradeço também a minha família, a meu pai Hildo, a minha mãe Claudete e a meu irmão Alex, que sempre me apoiaram; à Cintia, minha namorada amada, que é uma pessoa maravilhosa, que me fortalece; ao Prof. Bins, pela sua compreensão e paciência; aos Profes Luís Lamb e Artur Garcez, pela ajuda e esclarecimentos prestados; a todos os amigos que me acompanharam durante o mestrado, em especial ao Marcirio, que se dispôs a revisar meu trabalho; ao Exército Brasileiro, por ter me proporcionado esta oportunidade; à PUC-RS, que ultimamente tem sido a minha casa, e à Dell, pelo apoio inestimável.

**Resumo**

O principal objetivo deste trabalho foi a implementação do algoritmo de conversão Neuro-Simbólica do sistema C-IL$^2$P Modal, o qual estende o sistema C-IL$^2$P através do emprego da Lógica Modal no seu algoritmo de conversão. O algoritmo do sistema C-IL$^2$P é utilizado para a conversão unidirecional de um programa lógico escrito em linguagem simbólica em uma Rede Neural Artificial. O algoritmo C-IL$^2$P Modal não possui implementação prévia, somente modelagens e simulações, ao contrário do C-IL$^2$P. Uma das principais aplicações destes algoritmos situam-se na área da Inteligência Artificial, mais especificamente na modelagem de sistemas que lidam com o conhecimento.

**Palavras-chave:** Integração Neuro-Simbólica, Conexionismo, Redes Neurais Artificiais, CIL$^2$P, Lógica e Linguagem Lógica.

**Abstract**

This work introduces some specific algorithm implementations in Neural-Symbolic Integration area. The main algorithms treated here, the C-IL$^2$P and C-IL$^2$P extended by modalities from Modal Logic, are translations algorithms. The two algorithms realize the unidirectional conversion from a specific Logical Program to an Artificial Neural Network. One of the objectives is an extra knowledge supervisor method in Artificial Intelligence area, based in the implementation of the Modal C-IL$^2$P.

**Keywords:** Neuro-Symbolic Integration, Connectionism, Artificial Neural Networks, CIL$^2$P, Modal Logic and Logic Programming.

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Lista de Símbolos e Abreviaturas

| | | |
|---|---|---|
| **INS** | Integração Neuro-Simbólica | 1 |
| **IA** | Inteligência Artificial | 1 |
| **RNA** | Rede Neural Artificial | 2 |
| **IAS** | Inteligência Artificial Simbólica | 2 |
| **IAC** | Inteligência Artificial Conexionista | 2 |
| **C-IL$^2$P** | *Connectionist Inductive Learning and Logic Programming* | 3 |
| **IAC** | Inteligência Artificial Conexionista | 7 |
| **MLP** | *Multi-Layer Perceptron* | 7 |
| **RMSET** | *Root Mean Square Error - Training* | 12 |
| **RMSEV** | *Root Mean Square Error - Validation* | 12 |
| **NARX** | *Nonlinear AutoRegressive with eXogenous inputs* | 12 |
| **IAS** | Inteligência Artificial Simbólica | 15 |
| $\Box$ | Operador necessidade | 16 |
| $\Diamond$ | Operador possibilidade | 16 |
| $\omega_i$ | Um mundo especificado pelo índice "i" | 18 |
| **K** | Conhecimento | 19 |
| $\varphi$ | Um fato específico | 19 |
| **KBANN** | *Knowledge Based Artificial Neural Networks* | 21 |
| $A_{min}$ | Menor valor de ativação do(s) neurônio(s) | 23 |
| $\theta$ | Limiar de ativação do neurônio, viés do somatório | 23 |
| **W** | Peso sináptico | 23 |
| $W^M$ | Peso sináptico modal | 45 |
| $\kappa$ | Número de literais de uma cláusula | 58 |
| $\mu$ | Número de cláusulas com o mesmo conseqüente | 58 |
| $MAX_p$ | Maior dos números entre $\kappa$ e $\mu$ | 58 |

# Capítulo 1

# Introdução

*"Vivemos em um mundo de maravilhosa complexidade e variedade, um mundo onde os eventos nunca se repetem exatamente, mas nem por isso são completamente diferentes. Heraclitus disse a dois milênios e meio atrás que: - Nunca damos o mesmo passo duas vezes no mesmo rio."*
Prof. James A. Anderson - Cientista de Cognição e Linguagem

## 1.1 Contextualização

Conhecer a inteligência e os seus mecanismos fascina a humanidade. Entendendo-a, podemos alcançar objetivos como a sua reprodução. No entanto, ainda sabemos pouco sobre o que é inteligência e como ela funciona. Buscando entender a inteligência, a humanidade procura aprender cada vez mais sobre o cérebro e seus processos mentais. Uma grande dificuldade, além da natural complexidade do cérebro, é que a maioria das funções cerebrais e processos mentais estão de alguma forma relacionadas. Isso faz com que um pequeno problema numa destas funções ou processos gere conseqüências imprevisíveis em uma outra função ou processo mental.

Mesmo que ainda não haja um modelo das funções e processos cerebrais, contribuindo para que não se estabeleça um conceito preciso do que é inteligência, muitos modelos teóricos de inteligência ou parte dela vêm sendo propostos. Esses modelos, embora incompletos e com imprecisões, contribuem para o avanço na compreensão da inteligência e, em muitos casos, permitem que se executem no computador atividades consideradas inteligentes.

Dentre as várias ciências que estudam o cérebro e a inteligência, a Inteligência Artificial (IA) [33] é a área da Ciência da Computação que, além de propor modelos e atividades, também se preocupa em implementá-los. Aplicações em IA podem ser encontradas em vários outros campos do conhecimento, tais como: robôs autônomos [6]; sistemas especialistas em medicina [5]; geologia [1]; reconhecimento de padrões em medicina, segurança, sensoriamento remoto, reconhecimento de faces [14, 31]; reconhecimento de linguagem natural [7]; representação do conhecimento ou lógica [38]; entre outras.

Da mesma forma que é difícil definir a inteligência, também o é definir a Inteligência Artificial e o que é exatamente abrangido por ela, pois sempre estão surgindo fatos novos que revigoram as discussões e também porque há diversas ramificações que constituem a área. Uma definição da IA que pode ser feita no momento é: A Inteligência Artificial é uma área do conhecimento humano que possui como um de seus atributos o estudo da simulação em computador dos processos de pensamento.

A IA pode ser desenvolvida em um sistema que simula os processos mentais (Figura 1.1). A partir dos conhecimentos existentes nesse sistema, obtém-se a resposta correta pela utilização de heurísticas, segundo critérios definidos. A aprendizagem e o raciocínio são dois processos mentais aplicáveis ao conhecimento. Na aprendizagem, as entradas de um sistema são processadas formando novos conhecimentos. As entradas podem estar sob a forma de dados e/ou informações representando conhecimento e podem ser expressas por regras. O raciocínio difere pela obtenção de *fatos* a partir das entradas e/ou das regras existentes. A Figura 1.1 fornece a visão genérica e superficial de um sistema desta natureza, em que foram salientadas as seguintes etapas: a inserção, o refinamento e a extração do conhecimento.

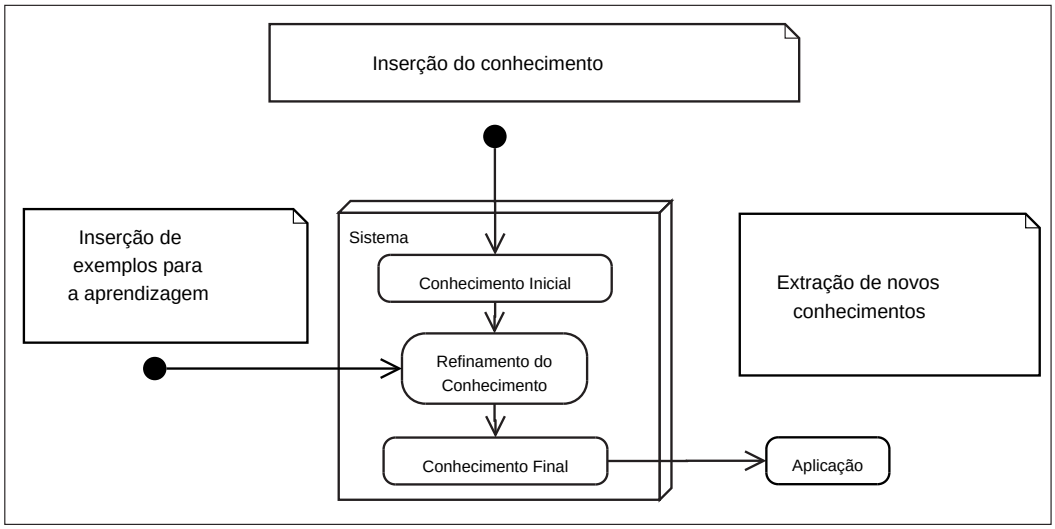


Figura 1.1: Sistema representativo de alguns processos mentais.

Existem vários tipos de processos de pensamento ou mentais, processos para síntese do conhecimento, processos para análise do conhecimento e processos que combinam a síntese com a análise de conhecimento [25]. Observa-se que os processos de pensamento nos homens seguem uma seqüência, partindo do nível simples da *memorização*, passando pela *compreensão* e pela *reflexão*, para a posterior *autonomia* do indivíduo [3]. A IA tenta simular esses níveis de processamento do pensamento. Dependendo da subárea da IA em que se esteja trabalhando, haverá ou não relação com algum desses processos. Nas últimas décadas, a evolução das subáreas, que constituem a IA, e suas interconexões se deram de forma crescente e abrangente.

Atualmente, configuraram-se dois paradigmas principais utilizados na Inteligência Artificial. A IA Simbólica (IAS), constituindo-se de elementos simbólicos organizados segundo a lógica, sendo naturalmente reconhecidos pelo homem. A IA Conexionista (IAC), mais abstrata, empregando Redes Neurais Artificiais (RNAs), trabalha com conceitos originados de modelos biológicos, ainda incompreendidos em sua totalidade. Assim, a IA inicialmente subdivide-se ou num contexto lógico-literal ou num contexto de modelo biológico-matemático. Uma modela a mente, e a outra, o cérebro [13].

O paradigma Neuro-Simbólico é o resultado da união de características complementares entre os paradigmas Conexionista e Simbólico. A Integração Neuro-Simbólica (INS) representa o paradigma Neuro-Simbólico pelo estudo das formas de união entre os dois paradigmas originais.

As conseqüências e possibilidades deste novo estudo são inúmeras, tais como a modelagem de sistemas para a representação do conhecimento ou de sistemas inteligentes.

Os sistemas inteligentes são tipicamente implementados de forma *simbólica*. Como exemplo, temos os sistemas especialistas [2, 33, 37] ou ainda a modelagem de agentes [10, 15], juntamente com a sua interação. Esses trabalhos representam exemplos de atuação em domínios do conhecimento na IA que podem ser modelados segundo a INS. Essa mudança de paradigma é um fato recente.

Em virtude de a INS ser uma área nova, as pesquisas ou trabalhos a serem realizados encontram-se em uma fase inicial. Podemos antever que sua evolução demandará trabalhos nas próximas décadas, tal como ocorreu com outras áreas do conhecimento. Já existem alguns, demonstrando o aprimoramento de sistemas Neuro-Simbólicos e as tendências de seu desenvolvimento, como exemplo [4, 17, 18], muitas vezes, em linhas divergentes de atuação. O nosso estudo segue a linha dos trabalhos dos autores e pesquisadores Artur S. d'Avila Garcez, Dov M. Gabbay, Gerson Zaverucha, Luís C. Lamb e Krysia Broda [9, 10]. Essa linha se caracteriza pelo emprego da Lógica Formal em conjunto com as Redes Neurais Artificiais.

O trabalho apresentado nesta dissertação foi desenvolvido visando principalmente a implementação de dois algoritmos situados no contexto da INS. São eles: o algoritmo de conversão do sistema *C-IL$^2$P* ("*Connectionist Inductive Learning and Logic Programming*") [12] e o algoritmo de conversão do sistema *C-IL$^2$P Modal* [11]. O algoritmo do sistema *C-IL$^2$P* é um algoritmo de conversão unidirecional, a conversão acontece no sentido do paradigma IAS para o IAC, não ocorrendo assim, a reversão (a conversão no sentido contrário). O algoritmo do sistema *C-IL$^2$P Modal* amplia as possibilidades de conversão, pois potencializa a modelagens de conceitos da IAS.

O algoritmo do sistema C-IL$^2$P já possui implementação com resultados [12]. No entanto, o algoritmo do sistema C-IL$^2$P estendido por Lógica Modal não possuía nenhuma implementação prévia. As implementações desses algoritmos geram alternativas de atuação da IA nas mais diversas áreas, em domínios específicos do conhecimento, tais como o reconhecimento de código genético ou a modelagem de raciocínio lógico.

Observa-se que, durante o transcorrer da dissertação, o termo C-IL$^2$P irá ser aplicado de igual forma a uma referência ao sistema e ao algoritmo de conversão desse sistema, por questão de conveniência. No entanto, o contexto ao qual o termo se aplica expressa o significado referido.

## 1.2 Aplicabilidade

Existem muitos exemplos de aplicação dos algoritmos INS, dependendo do foco de trabalho em questão. Comentaremos dois exemplos nos próximos parágrafos, onde notar-se-á que eles utilizam inferências lógicas em banco de dados. No primeiro exemplo temos o acréscimo de novos conhecimentos ao já existente, como ocorre em outros sistemas de IA. O segundo exemplo visa ao acompanhamento e à previsão de estados futuros, nos sistemas modelados pelo paradigma Neuro-Simbólico.

Neste primeiro exemplo, temos que, muitas vezes, um domínio incompleto de conhecimento pode ser aprimorado. Um banco de dados formado por regras empíricas, originadas do conhecimento de vários *especialistas*, pode ser revisado, confirmando algumas regras, refutando outras e também criando novas regras, contribuindo, dessa forma, para que se tenha uma base de dados mais sólida e consistente. Esses especialistas podem ser profissionais das mais diversas áreas do conhecimento, médicos, engenheiros, advogados, músicos, dentre outros, que traduzem a sua

experiência em fatos ou regras.

Quando desejamos um sistema inteligente, percebemos que a capacidade de adaptação a novos contextos deve estar presente. A adaptação significa a evolução desse sistema a partir do conhecimento inicial, que pode ser o conjunto de regras empíricas mencionado. A evolução reflete o aprimoramento do conhecimento inicial, que pode ser dado pelo acréscimo de novos fatos e regra, visando à obtenção de um domínio completo sobre o conhecimento para satisfazer o contexto trabalhado.

O segundo exemplo, de aplicação da INS, consiste na modelagem de raciocínios lógicos. A modelagem é feita pelo desenvolvimento seqüencial de sentenças lógicas. O raciocínio lógico permite o detalhamento das etapas evolutivas em um sistema Neuro-Simbólico projetado para modelar situações, para fins de acompanhamento. Quando levamos em consideração escalas de evolução "temporais", há a possibilidade de implementação em sistemas previsíveis. Esse exemplo é abordado na fase final do trabalho pelo emprego da Lógica Modal.

As possibilidades de aplicação da INS são inúmeras. Em nosso trabalho, consideramos apenas um pequeno conjunto de exemplos, quando comparado com as potencialidades de aplicação. A pretensão das implementações é a de contribuir, de alguma forma, para a orientação de novas pesquisas dentro do paradigma Neuro-Simbólico. Espera-se também que novos meios de modelagem de Inteligência Artificial resultem das pesquisas em INS feitas por este trabalho e os seus precedentes.

## 1.3 Questões técnicas

O trabalho, aqui apresentado, se baseia em um estudo da correspondência de conceitos entre os paradigmas da IA citados, derivando-se dos trabalhos [11, 12] existentes. As questões abordadas focam-se no estudo e emprego da INS.

Uma das principais questões na IA, a qual a INS busca solucionar, diz respeito à obtenção do conhecimento empírico. A principal dificuldade dos sistemas puramente conexionistas está em expressar o conhecimento contido nas Redes Neurais Artificiais (RNAs) [11, 25, 26, 30]. Podemos destacar três etapas no processamento do conhecimento por um sistema Neuro-Simbólico, sendo elas: a inserção do conhecimento, a obtenção de novos conhecimentos e a extração do novo conhecimento obtido. Em cada uma dessas etapas, existe um componente Neuro-Simbólico associado. A inserção e extração do conhecimento para o algoritmo estudado compõem interfaces do sistema INS; já a obtenção de novos conhecimentos restringiu-se ao emprego de RNAs.

O processo de aprendizado em uma RNA é difícil de ser acompanhado, conforme as técnicas atuais. Por outro lado, os sistemas simbólicos são facilmente compreendidos pelo homem, desde de que expostos na forma apropriada. A Lógica, muitas vezes, é utilizada em pesquisas na área de INS para formalizar o processo de aprendizado da RNA.

A Lógica também é empregada quando necessitamos da estruturação semântica de uma linguagem descrita por símbolos, resultando em uma *linguagem lógica*. A RNA obtida, conforme as implementações realizadas, aprende a partir das informações definidas pela linguagem lógica. O desejável é o domínio semântico sobre as referidas RNAs, de tal forma que os processos de aprendizagem sejam compreensíveis e transparentes.

## 1.4 Organização desta dissertação

A dissertação foi organizada a partir deste capítulo introdutório (Capítulo 1), em mais seis capítulos. A conceituação e o embasamento teórico, importantes ao entendimento do trabalho, são apresentados nos próximos três capítulos. O Capítulo 2 aborda as Redes Neurais Artificiais e seus processos. Os conceitos à programação baseada em linguagem lógica estão no Capítulo 3. O Capítulo 4 finaliza a parte conceitual apresentando alguns pontos importantes relativos à INS.

O trabalho desenvolvido está relacionado aos programas apresentados no Capítulo 5. Os principais exemplos nos quais os programas foram aplicados estão no Capítulo 6, assim como os resultados obtidos por eles. As considerações finais, no Capítulo 7, trazem reflexões sobre o trabalho realizado e apontam para algumas diretrizes para pesquisas futuras.

# Capítulo 2

# Aspectos relacionados à Inteligência Artificial Conexionista

"*Uma rede neural é um processador massivo, paralelo e distribuído, constituindo unidades simples de processamento, que tem propensão natural para armazenar o conhecimento e torná-lo disponível para uso. Assemelha-se ao cérebro...*"
Prof. Simon Haykin (1931) - Cientista conexionista

A Inteligência Artificial Conexionista (IAC), ou simplesmente Conexionismo, utiliza Redes Neurais Artificiais para simular, em uma proporção menor, o funcionamento do cérebro humano. Algumas vantagens desse paradigma são a fácil aprendizagem e a capacidade de generalização, tornando-o útil para sistemas de classificação. A principal desvantagem é a formação de resultados com difícil interpretação pelo homem.

Este capítulo foi dividido em três seções: A Seção 2.1 introduz teoricamente o que é uma RNA e os seus componentes; a Seção 2.2 menciona os processos que normalmente são empregados no aprendizado da RNA; a Seção 2.3 descreve, de forma genérica, alguns conceitos sobre RNAs recorrentes.

## 2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são modelos matemáticos com representação computacional de grupos de neurônios biológicos. As RNAs são constituídas por conjuntos de unidades básicas, conhecidas como Neurônios Artificiais ou Unidades Neurais, ou ainda apenas por neurônios ou unidades. A Figura 2.1 ilustra um exemplo de representação de um neurônio. Os neurônios podem estar agrupados em RNAs do tipo MLP (*Multi-Layer Perceptron*), que é a mais conhecida no meio acadêmico, devido a sua simplicidade.

A função que uma RNA irá desempenhar depende da soma do conjunto de funções de cada neurônio. As funções características do neurônio são indicadas na Figura 2.1 e também são mostradas no paralelo entre as partes similares que compõem os dois tipos de neurônios, apresentado pela Tabela 2.1. Resumidamente, as "n" entradas $X_i(t)$ do neurônio (sendo que, $1 < i < n$) são

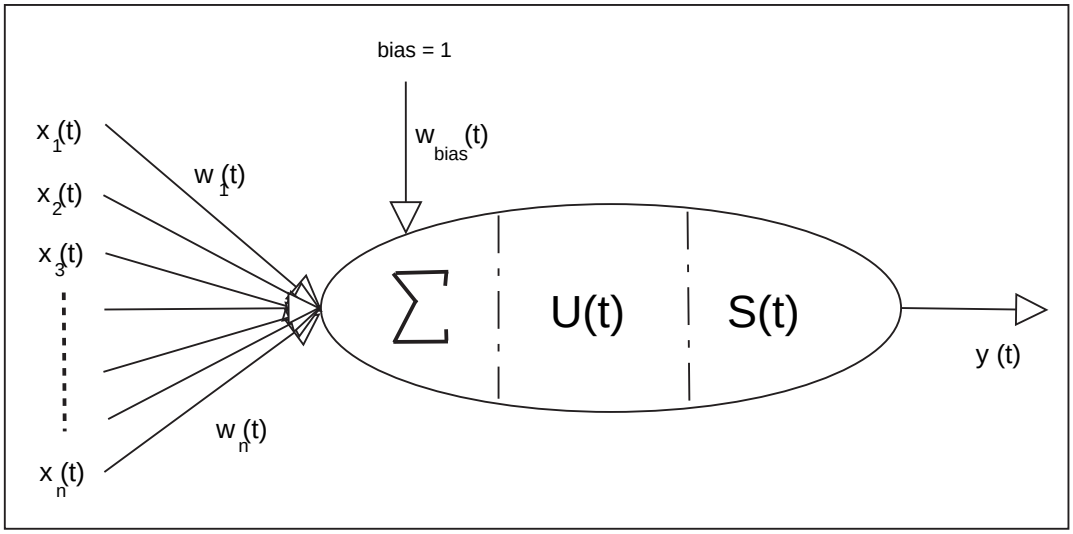


Figura 2.1: Representação Neural.

somadas, o resultado é processado no corpo do neurônio para produzir a saída $Y(t)$, todas são dependentes do tempo ("t").

Tabela 2.1: Analogia entre neurônios.

| **N$^o$** | **Biológico** | **Artificial** |
|---|---|---|
| 1 | Corpo | Funções $\sum$ , $U_i$(t) e $S_i$(t) |
| 2 | Núcleo | Função $U_i$(t) |
| 3 | Axônio | Caminho até $Y_i$(t) |
| 4 | Dendrito | Entradas $X_i$(t) |
| 5 | Árvores Axioniais | Conjunto de saídas Y(t) |
| 6 | Sinapses | Ligações: $Y_i(t) \longleftrightarrow X_j(t)$ |

Genericamente, cada neurônio pode ser considerado uma unidade de processamento independente, que, quando agrupada em uma Rede Neural, contribui para a execução do processamento massivo de informações. O aprendizado em uma RNA inicia pelo processamento de um conjunto de informações. A cada novo processamento, as informações são classificadas. Ao final de todo o processamento, a RNA estará cada vez mais sensível a novos processamentos ao mesmo conjunto de informações.

A maioria das RNAs são projetadas para aprender as informações que são passadas a elas, deixando-as aptas à interpretação apropriada de novas informações, conforme o objetivo desejado na sua aplicação. Se uma RNA aprendesse todas as informações, ela atingiria 100% de acertos, mas esse é um fato raro.

Quando, por ventura, se atinge um percentual alto de acertos em uma RNA (100% ou valor próximo), podemos formular duas hipóteses, ou o problema foi completamente solucionado, ou a RNA decorou as informações passadas a ela. Na primeira hipótese, a RNA está aberta para soluções de outros problemas pertencentes à mesma classe para a qual foi treinada. Na segunda,

que não é desejável em muitas aplicações, o universo de soluções apresentado pela RNA fica restrito.

O funcionamento dos neurônios pode ser descrito da seguinte forma: eles se interligam através de conexões (sinapses), que possuem, num primeiro momento, parâmetros ajustáveis, os pesos sinápticos ($w_i(t)$). Esses pesos controlam a intensidade das conexões e, ao término do aprendizado da RNA, são responsáveis pelo armazenamento do conhecimento adquirido pela rede.

Na entrada de cada neurônio, é feita a soma (conforme equação 2.1) dos valores das saídas dos neurônios anteriores, os quais são ponderados pelos pesos sinápticos correspondentes.

$$\sum(X_i(t) * w_i(t)) \tag{2.1}$$

O resultado desse somatório é mapeado não-linearmente por uma função de ativação $U_i(t)$, que usualmente é do tipo sigmóide (ver Figura 2.2). Para esse somatório, existe um viés ("bias"), que auxilia no estabelecimento do limite de aceitação de uma entrada à função ativação; normalmente possui o valor "1". Obtemos o limiar de entrada da função-somatório através da multiplicação do viés pelo peso da conexão correspondente ($w_{bias}(t)$), como é mostrado na Figura 2.1. O resultado dessa multiplicação é representado por $\theta_i$ ("*threshold*"), conforme equação 2.2.

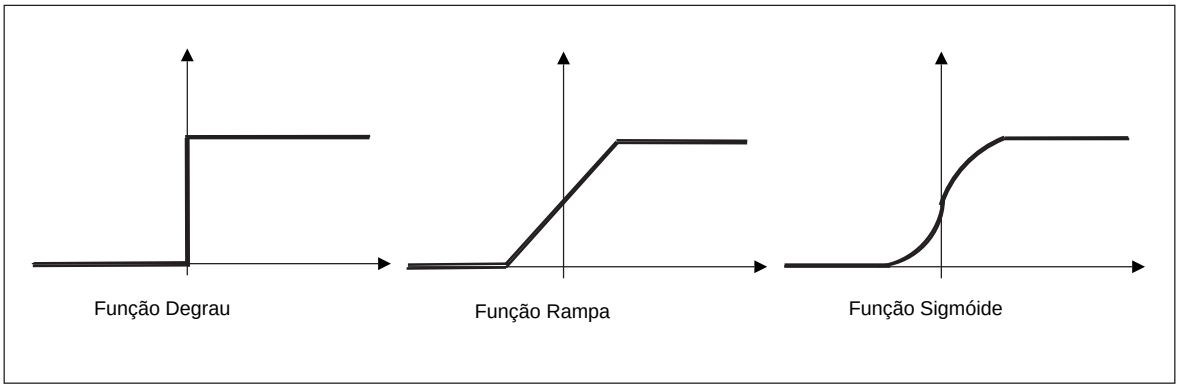


Figura 2.2: Tipos de funções de ativação dos neurônios.

$$U_i(\sum(X_i(t) * w_i(t)) - \theta_i) \tag{2.2}$$

O próximo passo é a execução da função de saída do neurônio $S_i(t)$, para a qual se pode aplicar um limiar[1] de saída. No trabalho apresentado nesta dissertação, utilizou-se $S_i(t)$ como função identidade, ou seja:

$$S_i(t) = U_i(t) \tag{2.3}$$

Somente então, o valor $Y_i(t)$ é obtido na saída do neurônio [19, 23].

$$Y_i(t) = S_i(t) \tag{2.4}$$

Uma arquitetura ou topologia básica empregada nas RNAs é a do tipo MLP, na qual existem no mínimo três camadas. Em uma RNA com três camadas e um conjunto variável de neurônios é

[1] A ativação de um neurônio pode ser controlada através dos limiares atribuídos a ele.

representado por ($N_i$ x $N_j$ x $N_k$)[2]. Acrescentamos outras camadas à camada interna ou escondida da RNA, quando trabalhamos com RNAs de múltiplas camadas com número superior a três. A Figura 2.3 representa uma RNA, na qual existe a separação entre a camada de entrada, a de saída e o conjunto interno de camadas escondidas.

Cada neurônio da camada de entrada tem duas particularidades: Nele existe apenas uma entrada, e o valor lido por ele é repassado inalterado à saída. Nos neurônios das demais camadas da RNA, os círculos com múltiplas entradas, existentes na Figura 2.3, equivalem-se ao neurônio representado pela Figura 2.1.

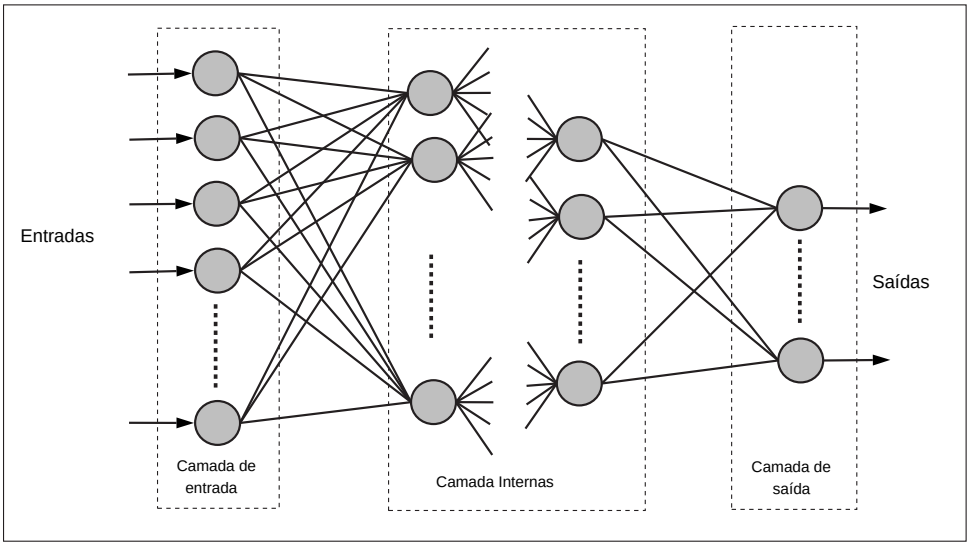


Figura 2.3: Representação de uma RNA.

As conexões sinápticas entre as camadas podem ser totalmente conectadas ou, no caso da existência de algum conhecimento prévio, parcialmente conectadas. As conexões neurais se modificam durante o aprendizado da RNA. O conjunto de neurônios, as conexões sinápticas e as funções de ativação constituem o que se chama de *topologia da RNA*.

Após a topologia da RNA estar definida, inicia-se o treinamento da rede, que é feito com o auxílio de um algoritmo de aprendizagem. A modificação dos valores da RNA, conforme o andamento do aprendizado, ocorrem no sentido direto[3], pela propagação das alterações na RNA, ou no sentido inverso, através da retropropagação. Um dos algoritmos mais conhecidos é o de retropropagação ("*backpropagation*") [32], cujos passos são resumidos a seguir [39]:

1. A ativação da camada de entrada é originada pelo ambiente externo à RNA.
2. A ativação se propaga diretamente na rede, através das conexões, indo da camada de entrada até a camada de saída.
3. Os erros são determinados em função das diferenças entre as ativações computadas e as desejadas, nas unidades de saída.
4. Os erros refletem-se, propagando-se de volta (retropropagação) nos mesmos caminhos utilizados nas ativações das unidades.

[2]Exemplo: Se a camada de entrada contém cinco neurônios, a interna três neurônios e a camada de saída apenas um, então, a distribuição dos neurônios é expressa por (5 x 3 x 1).

[3]O sentido direto em uma RNA é o sentido que parte da camada de entrada até a última camada (saídas).

5. As mudanças nos pesos dos caminhos persistem até a redução dos erros para um patamar mínimo aceitável.

O algoritmo de retropropagação utiliza a Regra Delta para computar o valor diferencial de peso ($\Delta$w). Esse valor é calculado e somado ao peso atual, para corrigi-lo, conforme o passo 4 do algoritmo. As equações abaixo expressam o cálculo do $\Delta$w, conforme [19].

$$\Delta w_{kj}(n) = \eta * \delta_k(n) * y_j(n) \tag{2.5}$$

onde, $\eta$ é a taxa de aprendizagem da RNA; $\delta_k(n)$ é o gradiente local, descrito abaixo e $y_j(n)$ é composto pelos valores de entrada dos neurônios da última camada.

$$\delta_k(n) = e_k(n) * \varphi'_k(v_k(n)) \tag{2.6}$$

O gradiente local é composto pelo erro de saída $e_k(n)$ e a derivada da função de saída $\varphi'_k(v_k(n))$. Os valores são calculados para o *exemplo "n"*. Os índices "j" e "k", identificam, respectivamente, os neurônios da camada anterior e da camada posterior a conexão, no sentido *entrada→saída*.

O erro aplicado à fórmula do gradiente local (2.6) origina-se da última camada (camada de saída). A RNA aprende guiada por processo supervisionado. Isso significa que já se conhece a saída para as entradas que são passadas para ela. Visa-se à máxima aproximação desse valor conhecido de saída ou o mínimo erro. Esse erro é obtido diretamente no neurônio de saída por:

$$e_k(n) = (valor_{desejado} - valor_{obtido})_k(n) \tag{2.7}$$

O erro calculado com (2.7) não se aplica às demais camadas internas e à de entrada, no sentido *saída→entrada* (retropropagação). Nesse caso, o erro será uma estimativa.

## 2.2 Aprendizagem na Rede Neural Artificial

A aprendizagem processa-se durante o tempo no qual os padrões a serem aprendidos são passados para a RNA. Os padrões ou exemplos são valores a serem inseridos na camada de entrada. Cada neurônio faz a leitura do valor que lhe é passado. Conjuntos de exemplos são utilizados repetidas vezes para que o aprendizado da RNA seja efetivado.

Um conjunto de exemplos específico é passado à RNA por várias épocas durante o aprendizado; por sua vez, cada época indica que todos os exemplos do conjunto foram lidos. O tempo total de aprendizagem depende principalmente da quantidade e da qualidade dos exemplos apresentados à RNA.

Normalmente, o aprendizado do conjunto total de exemplos é dividido em três subconjuntos, que dentro do contexto conexionista, são conhecidos como Treinamento, Validação e Teste. Os subconjuntos caracterizam as fases do processo de validação, conhecido como validação cruzada com parada antecipada. Nesse tipo de validação, temos que:

1. O Treinamento é a fase na qual o primeiro subconjunto de exemplos é lido e processado. Além disso, os pesos são armazenados nesta fase.

2. A Validação é a fase na qual se verifica o andamento do aprendizado, sendo um novo subconjunto de exemplos utilizado para se monitorar o treinamento.

3. O Teste é a verificação final do aprendizado da RNA, ocorrendo após a obtenção de um resultado satisfatório na Validação.

O aprendizado inicia com o treinamento da RNA durante várias épocas (ditas épocas de treinamento), até que o erro na RNA esteja em um patamar mínimo. Após várias épocas de Treinamento, ocorre uma de Validação. Se o erro computado na Validação não for satisfatório, a RNA inicia uma nova época de Treinamento.

A fase de teste expressa o número de acertos resultantes da RNA, baseado no subconjunto de exemplos restante; assim, a eficiência da RNA é medida estatisticamente.

A monitoração do erro é a métrica de avaliação do processo de aprendizagem da RNA. Portanto, para cada uma das fases no desenvolvimento da RNA, deve-se monitorar o erro periodicamente. Uma das maneiras de medir o erro global da RNA é através do Erro Médio Quadrado ("*mean square error*") ou a Raiz do Erro Médio Quadrado (RMS - "*root mean square error*"), utilizando a equação 2.8, obtemos essa medida, [19, 23, 39].

$$Erro_{rms} = \frac{1}{NEx * NEpo} * \sqrt{\sum (saída_{desejada} - saída_{obtida})^2} \tag{2.8}$$

Na equação 2.8, $NEx$ indica o número de exemplos que estão sendo lidos pela RNA, e $NEpo$ é o número de épocas utilizadas no aprendizado.

A Figura 2.4 mostra o diagrama em blocos dos processos de validação cruzada com parada antecipada. O andamento dos processos é acompanhado, seguindo-se os fluxos apresentados. A maioria dos programas desenvolvidos durante o trabalho utilizaram uma estrutura igual à do diagrama para executar o aprendizado.

## 2.3 Redes Neurais Recorrentes

Para entendimento de alguns pontos apresentados no capítulo 5, é necessária uma visão de redes neurais recorrentes. Portanto, nesta seção, abordaremos alguns aspectos desse tipo de rede neural, em especial as redes recorrentes auto-regressivas.

As RNAs Recorrentes já foram estudadas para o emprego em máquinas de estado finitas, e as atuais pesquisas demonstram que, no mínimo, se podem obter as potencialidades de máquinas de Turing [21]. Exemplos de redes recorrentes são as redes propostas por Hopfield, as Redes Bidirecionais, as Redes de Jordan e as Redes Recorrentes de Tempo Real (apud Barreto [34]). Um dos principais tipos de RNAs recorrentes é a NARX ("*Nonlinear AutoRegressive with eXogenous inputs.*").

As RNAs Recorrentes empregam, como princípio de funcionamento, o retorno da resposta obtida em um tempo anterior "(t-1)". Em cada novo ciclo de aprendizagem, a RNA fará a leitura, tendo como uma de suas entradas a saída obtida no ciclo precedente, tal como é mostrado na Figura 2.5. Os ciclos do processo de aprendizagem, baseados na realimentação da RNA, se tornarão cada vez mais efetivos à medida que os exemplos são lidos pela RNA, como também acontece na aprendizagem sem realimentação. A diferença é que a realimentação age como viés da aprendizagem.

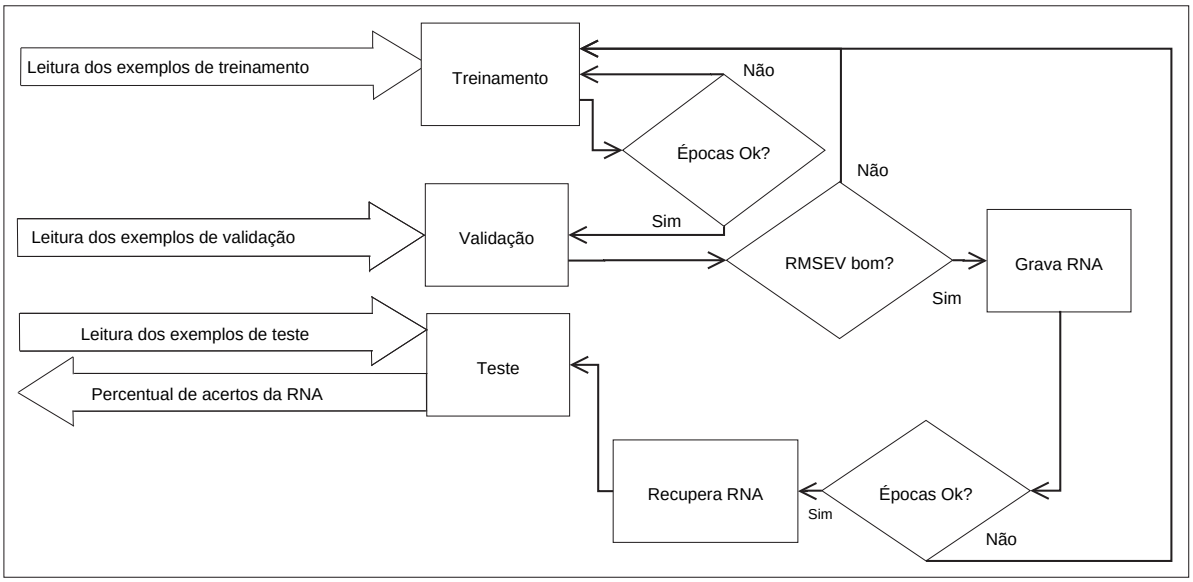


Figura 2.4: Processo de validação cruzada com parada antecipada.

Segundo [34], as RNAs recorrentes permitem a modelagem de sistemas reativos e instintivos, pois levam em conta, além das entradas normais, as de controle baseado em estados anteriores (um tipo de "memória"), enquanto que a capacidade de modelagem das RNAs sem realimentação é própria para sistemas reflexivos (funcionam segundo estímulo-resposta).

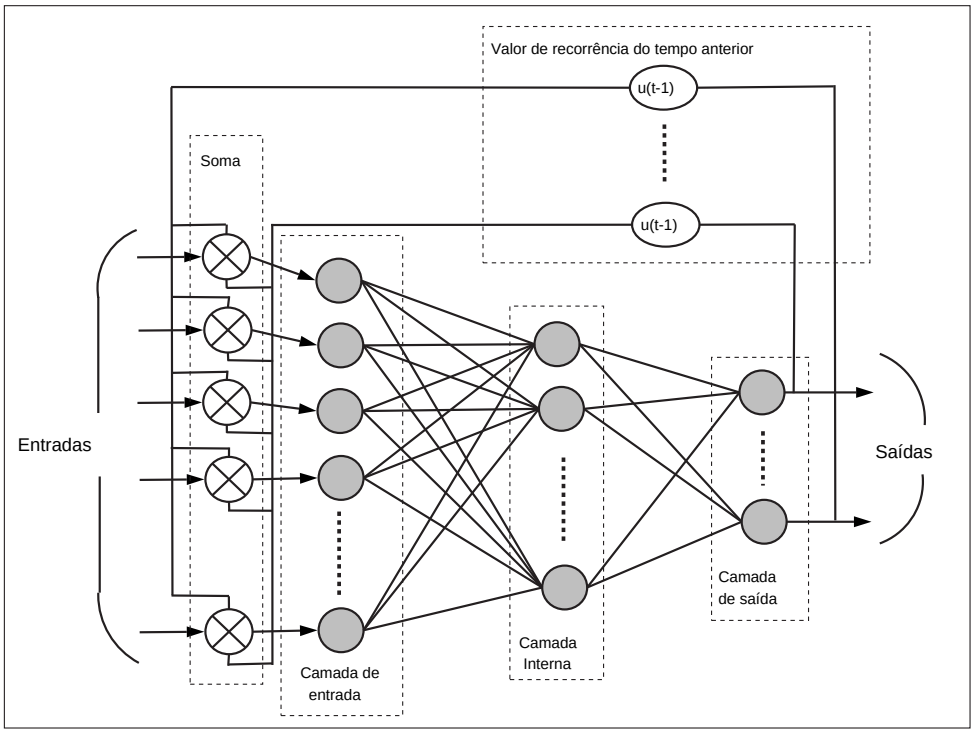


Figura 2.5: Representação de RNA recorrente.

# Capítulo 3

# Aspectos relacionados à Inteligência Artificial Simbólica

"*As palavras do mundo estão loucas para formar sentenças.*"
Gaston Bachelard (1884 - 1962) - Filósofo francês

A Inteligência Artificial Simbólica (IAS) emprega a representação de conceitos através de elementos simbólicos, compatíveis com o processamento humano de informações. A IAS é o paradigma da IA que aborda a descrição literal, por símbolos gráficos, dos domínios e suas relações, formalizando assim as idéias. Isso possibilita o desenvolvimento de estruturas passíveis do tratamento computacional, nas quais um dos objetivos é a representação do conhecimento no computador para fins de sua manipulação. O conhecimento, na maioria das vezes, é estruturado seguindo uma seqüência lógica. Um outro ponto positivo é que esse paradigma tem alta potencialidade na explicação dos resultados obtidos.

O presente capítulo apresenta introduções aos tipos de lógicas que comumente são utilizadas para a formalização do conhecimento. Iniciamos, estudando dois tipos de lógica: a proposicional na Seção 3.1, e a modal, na Seção 3.2, que contribuem para a formação de uma terceira lógica, a Lógica Epistêmica, apresentada na Seção 3.3. Essa lógica é utilizada para a estruturação do raciocínio, na modelagem de sistemas.

## 3.1 Linguagens estruturadas pela lógica

Com a Lógica Clássica de Primeira Ordem, podemos representar domínios específicos, contendo objetos, entidades, seus atributos e as relações existentes nesse domínio [30]. Utilizamos a lógica no contexto da IAS, para a estruturação semântica e fundamentação das idéias. O conjunto dessas estruturas forma a linguagem chamada de Linguagem Lógica. A Linguagem Lógica também pode ser descrita sob forma de estruturas textuais, conforme a sua sintaxe.

O trabalho, aqui apresentado, utilizou arquivos-texto, contendo estruturas baseadas nas cláusulas lógicas. O corpo da cláusula, ou antecedente, é composto por literais. Os literais são, por sua vez, separáveis em átomos e/ou suas negações. Um átomo é uma variável proposicional. A cabeça da cláusula, ou conseqüente, equivale ao resultado lógico do antecedente. A expressão

abaixo é um exemplo genérico de cláusula lógica.

$$A_1 + a_2 * a_3 + ... + A_n \rightarrow B \tag{3.1}$$

Nesse exemplo, os literais do antecedente iniciam em "$A_1$" e prosseguem até o último literal "$A_n$". O literal "$A_2$" foi subdividido em dois átomos "$a_2 * a_3$", para exemplificar uma possível composição. O resultado lógico é apresentado no conseqüente pelo literal "$B$". A operação lógica de disjunção, representada por "+", tem a finalidade de fornecer o sentido lógico "`ou`", significando que basta que apenas um dos literais exista e seja verdadeiro, para que tenhamos o conseqüente $B$ também verdadeiro.

A operação lógica de conjunção é representada, na expressão, por "$*$" entre os átomos "$a_2$" e "$a_3$". Seu significado lógico é o de atribuir o valor verdadeiro ao literal "$A_2$", somente se seus dois átomos forem verdadeiros. O operador lógico de negação "$\neg$" também pode ser utilizado nas expressões e atribui um sentido lógico verdadeiro ao que é falso e vice-versa.

Um programa que utiliza a Linguagem Lógica, digamos $P$, é formado por um conjunto de cláusulas lógicas, tal qual a mostrada no exemplo acima (3.1). Um programa possui várias *interpretações*. Cada interpretação corresponde ao valor lógico atribuído ao conjunto de átomos do antecedente simultaneamente. Chama-se avaliação do programa $P$ o mapeamento entre interpretações e os respectivos resultados (verdadeiro ou falso). O modelo do programa é aquele no qual as interpretações conduzem $P$ a uma resposta verdadeira.

## 3.2 Lógica Modal

A Lógica Modal [8] trabalha com o conceito de mundos. Um mundo representa um contexto no qual algumas regras lógicas são válidas. A Lógica Modal caracteriza-se pelo condicionamento de mundos artificiais de forma bastante peculiar, pois vários mundos podem ser criados relacionando-se ou não entre si. No emprego da Lógica Modal, mantém-se como base a Lógica Clássica Proposicional; porém, impõem-se restrições ou adicionam-se extensões aos seus valores, segundo o conjunto de regras do mundo onde se trabalha. Saul Kripke [24, 27] introduziu as idéias de necessidade e possibilidade (representadas pelos operadores lógicos $\Box$ e $\Diamond$, respectivamente) nesses mundos, gerando, assim, dois novos conectivos, além dos existentes na lógica clássica.

A semântica dos mundos de Kripke é dada por uma tupla de quatro elementos, representando estruturas na Lógica Modal, dando a noção sobre os mundos, mundos possíveis, mundos acessíveis, mundo atual e função de interpretação. A tupla está sob a seguinte forma:

*(U, R, mi, Fi)*

O primeiro elemento da tupla representa um conjunto de mundos $U$, para o qual será feita a relação binária $R$ com o mundo atual $m_i \in$ U. O mundo atual é o elemento em que a tupla é empregada. O quarto elemento é a função $F_i$, que fornece os valores-verdade constantes em cada mundo. As tuplas e a forma como as empregamos possibilitam a construção de raciocínios baseados na lógica, que servem para a representação dos casos de uso inseridos na programação lógica empregada pelo trabalho.

Um exemplo de desenvolvimento de raciocínio lógico, segundo a Lógica Modal, pode ser dado pela seguinte proposição aplicada ao contexto do esporte:

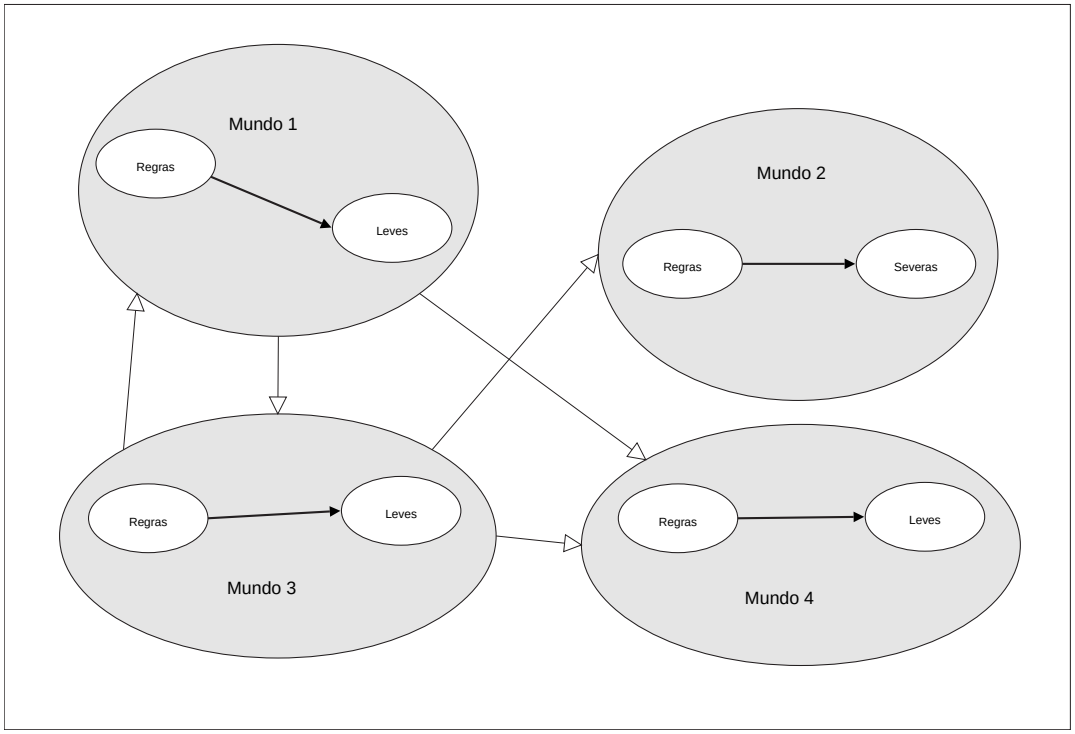


Figura 3.1: Exemplo de Lógica Modal.

*"As regras (*esportivas*) são leves."* - Que pode ser substituída por um símbolo proposicional tal como "$R$".

Para a proposição e para os mundos da Figura 3.1 podemos dizer que:

No Mundo 1: "$R \rightarrow \top$"
No Mundo 2: "$R \rightarrow \bot$"
No Mundo 3: "$R \rightarrow \top$"
No Mundo 4: "$R \rightarrow \top$"

A partir do Mundo 1, podemos verificar que os mundos relacionados com ele (excetuando-se o Mundo 2) possuem $R$ verdadeira ($\top$). Isso indica que *necessariamente* a proposição será verdadeira para todos os mundos desse conjunto, que compõem um universo em particular. Definimos, então, que: $\Box R$ para o Mundo 1.

Nessa situação, recorrendo a analogia esportiva, podemos entender que os Mundos um, três e quatro são jogos, os quais fazem parte de uma competição (dada pelas relações), na qual todas as regras desses jogos são leves.

Uma outra situação, que pode acontecer, ocorre quando a partir do Mundo 3 existe uma das relações que aponta para o Mundo 2, na qual $R$ é falsa ($\bot$). Com isso, a partir do Mundo 3, percebemos que existe a *possibilidade* de $R$ ser falsa. Definimos, então, que: $\Diamond R$ para o Mundo 3.

Novamente pela analogia, nessa situação, existe a possibilidade de que um dos jogos da competição tenha as suas regras mais rigorosas, que a dos outros jogos.

### 3.2.1 Inserção e extração de operadores modais

O sistema em Lógica Modal baseia-se principalmente nas relações existentes entre os mundos, ou quando aplicada ao nosso contexto, nas conexões entre as sub-RNAs criadas. A existência de uma conexão depende de duas operações: a inserção de operador e a extração de operador, como explicitado a seguir [11]:

- `Inserção de operadores`

A inserção de operador aplica-se tanto ao operador de necessidade quanto ao de possibilidade. Isso só depende da programação lógica inserida. A inserção é representada pela conexão entre o mundo de origem e os de destino, normalmente com peso unitário.

A inserção para a necessidade é empregada quando um mundo ($\omega_1$) específico se relaciona ($R(.)$) com outros mundos ($\omega_i$), onde haja necessariamente a existência do mesmo componente de regra ($\alpha$) em todos esses mundos ($\omega_i$). A inserção de operador necessidade ( I) se caracteriza pela seguinte regra modal:

$$\frac{[\omega_i : \alpha] \ldots R(\omega_1, \omega_i)}{\omega_1 : \Box\alpha}$$

O operador possibilidade somente será inserido se existirem as duas condições seguintes:

1. Ao menos um dos mundos ($\omega_2$) com o qual o mundo de origem ($\omega_1$) se relaciona possuir um componente de regra ($\alpha$).
2. O componente da regra é similar ao existente no mundo de origem.

A regra modal para a inserção do operador possibilidade ($\Diamond$I)é expressa a seguir:

$$\frac{\omega_2 : \alpha \ ; \ R(\omega_1, \omega_2)}{\omega_1 : \Diamond\alpha}$$

- `Extração de operadores`

A extração de operadores modais se processa inversamente à inserção, no que se refere ao sentido das conexões entre os mundos. O mundo de destino da inserção passa a ser o mundo de origem na extração. Os casos nos quais se aplica a extração de operadores modais dependem do tipo de operador (necessidade ou possibilidade), da existência da relação entre os mundos e da existência do mesmo componente de regra nos mundos.

Como exemplo temos: Se, em um mundo ($\omega_1$), existe o operador necessidade antecedendo um componente de regra ($\Box\alpha$) e este mundo se relaciona com outro mundo ($\omega_2$), então, haverá a conexão relativa à extração do operador necessidade ( E), tendo $\omega_2$ como origem e $\omega_1$ como destino. A regra lógica para esse caso é:

$$\frac{\omega_1 : \Box\alpha \ ; \ R(\omega_1, \omega_2)}{\omega_2 : \alpha}$$

Quando, em um mundo ($\omega_1$), existe o operador possibilidade agregado a algum componente de regra ($\Diamond\alpha$), em outro mundo ($\omega_2$), existe esse mesmo componente de regra e também a

relação entre esses dois mundos, então, haverá a conexão de retorno ligando o $\omega_2$ a $\omega_1$. Nesse caso, aplica-se a extração do operador possibilidade ($\Diamond$E), como é expresso pela regra abaixo:

$$\frac{\omega_1 : \Diamond\alpha}{\omega_2 : \alpha \ ; \ R(\omega_1, \omega_2)}$$

## 3.3 Lógica Epistêmica

A Lógica Epistêmica é a subárea da lógica que analisa formalmente os processos de raciocínio sobre o conhecimento. Mais sucintamente é a lógica do conhecimento [15]. A Lógica Epistêmica em conjunto com a Lógica Modal, tornam possível a modelagem de estudos de caso típicos da IA, mais precisamente quando desejamos trabalhar com agentes cognitivos.

J. Hintikka foi um dos precursores do estudo formal da Lógica Epistêmica com o livro chamado *Conhecimento e Crença* [20]. Esta área estuda principalmente a conexão entre o conhecimento e a ação, quais informações de um conjunto de processos desencadeiam ações em outro conjunto de processos e como isso acontece. Os processos podem ser vistos como um mundo em particular ou um modelo de agente, então, o desencadeamento das ações desse agente dependem do seu conhecimento sobre outros agentes em uma rede de dependências.

O conhecimento ($K$) de um agente qualquer (índice $i$) sobre um fato específico (digamos: $\varphi$) é representado da seguinte forma:

$K_i\varphi$, onde (0<i<n-ésimo) e (i$\in \aleph$)

Quando, por exemplo, o agente 1 conhece o fato "$\varphi$", ou seja:

$K_1\varphi$

E se "$\varphi$" refere-se ao conhecimento do agente 2 sobre outro fato ($\phi$), temos:

$\varphi = K_2\phi$

A representação completa é:

$K_1K_2\phi$.

Simplificadamente o conhecimento em um grupo de agentes pode apresentar-se de três maneiras diferentes:

1. O conhecimento sobre determinado fato pode ser comum ao grupo. - Todos conhecem um fato e todos sabem que todos conhecem aquele fato.
2. Todos os agentes possuem o conhecimento sobre determinado fato. - Todos conhecem um fato, mas não existe conhecimento sobre o que o outro agente sabe.
3. O conhecimento existe no grupo, mas está distribuído no grupo. Isso significa que cada agente contém alguma informação sobre o conhecimento.

O conhecimento e a Lógica Modal podem ser relacionados através de cinco propriedades sobre o conhecimento, que foram expressas abaixo, baseadas em [15]:

1. Se, em uma estrutura de mundos, existe o conhecimento sobre um fato $A$ e desse fato implica o fato $B$, então existe o conhecimento sobre o fato $B$.

2. Se, em uma estrutura de mundos, existe algum fato, sabe-se que os agentes que pertencem a esse mundo conhecem aquele fato.
3. Se os agentes de algum mundo conhecem algum fato, então, este fato existe no mundo.
4. Se algum agente conhece algum fato, então, ele sabe que ele conhece o fato.
5. Se algum agente não conhece algum fato, então, esse agente têm ciência disso.

Observa-se que a primeira propriedade é uma regra de inferência, também conhecida na lógica como "modus ponens". As propriedades ou mesmo o conhecimento em si são os componentes de situações diversas que compõem um sistema inteligente e dinâmico. A Lógica Epistêmica nos possibilita a estruturação do raciocínio. A Seção 4.4, do próximo capítulo, mostra alguns estudos de caso nos quais é possível esta estruturação.

# Capítulo 4

# A Integração Neuro-Simbólica

"*Para resolvermos problemas realmente difíceis, temos que usar muitos tipos diferentes de representações.*"
Marvin Minsky (1927) - Cientista da Computação

A Integração Neuro-Simbólica (INS) é o terceiro paradigma abordado neste trabalho. Ele surge pela união dos dois paradigmas apresentados nos Capítulos 2 e 3. Alguns motivos para o surgimento da INS são apresentados na Seção 4.1. As duas seções subseqüentes, 4.2 e 4.3, contêm os algoritmos originais propostos à implementação [12, 11]. No final deste capítulo, Seção 4.4, algumas propostas de aplicação dos algoritmos são mostradas, considerando o apresentado pelos autores dos algoritmos originais [10].

## 4.1 Por que integrar?

A união entre as áreas conexionista e simbólica é conveniente para obter-se um sistema de IA com melhor desempenho, pois ambos os paradigmas são complementares um ao outro [25, 26]. A Tabela 4.1 sintetiza as características dos paradigmas. Parte do estudo da INS se justifica pelas vantagens trazidas em agregar o fácil aprendizado do conexionismo com a explicabilidade do simbolismo [35].

Uma outra razão para o desenvolvimento de um sistema construído segundo o paradigma Neuro-Simbólico é a possibilidade de se trabalhar com o conhecimento, integrando características diferentes apresentadas pela Tabela 4.1. Sistemas que utilizam a INS podem obter um resultado melhor nas aplicações baseadas em conhecimento do que o obtido por sistemas puros. Um exemplo de INS pode ser um Sistema Especialista [2, 33], produto conhecido da IA Simbólica, que pode ser modelado com as RNAs.

A modelagem de um sistema na INS pode acontecer de várias formas conforme a proporção de seus componentes conexionistas e simbólicos, nos chamados Sistemas Híbridos Neuro-Simbólicos. Um sistema Neuro-Simbólico bem conhecido é o *KBANN* ("*Knowledge Based Artificial Neural Networks*") [36], que utiliza três algoritmos para executar a aprendizagem do conhecimento simbólico em RNAs: o algoritmo de inserção de regras simbólicas, o de treinamento por retropropagação e o algoritmo de extração das regras geradas. Outro exemplo de sistema Neuro-Simbólico

Tabela 4.1: Comparação entre dois paradigmas da Inteligência Artificial.

| **Característica** | **Simbólico** | **Conexionista** |
|---|---|---|
| Aprendizagem | Difícil | Fácil |
| Explicabilidade | Fácil | Difícil |
| Degradabilidade | Rápida | Lenta |
| Algoritmo | Seqüencial | Paralelo |
| Generalização | Difícil | Fácil |
| Resistência ao Ruído | Fraca | Boa |
| Estruturação dos dados | Fácil | Difícil |

é o C-IL$^2$P, que é um dos focos deste trabalho.

O C-IL$^2$P ("*Connectionist Inductive Learning and Logic Programming*") é um sistema paralelo para a computação massiva. A utilização de RNAs propicia a integração do aprendizado indutivo por exemplos com o conhecimento adquirido no aprendizado dedutivo da programação lógica [12].

Uma representação de como o sistema completo funciona é mostrada na Figura 4.1. A seqüência de passos da INS que aparecem nessa figura, conforme [12], é dada dessa forma: (1) Os conhecimentos já adquiridos, representados por proposições de uma linguagem lógica, são inseridos na RNA. (2) Ocorre o treinamento da RNA com o uso de exemplos. (3) A RNA irá processar a programação inserida com o que foi aprendido pelo uso dos exemplos, por meio de um sistema de computação em paralelo. Em (4), o sistema irá consistir o conhecimento extraído da rede neural treinada. Esse passo faz a revisão automática da programação obtida. No estágio final (5), irá ocorrer a extração do conhecimento, que é analisado por um especialista (humano), responsável pela realimentação do sistema, fechando assim o ciclo [4, 9, 12].

O trabalho descrito por esta dissertação não ambicionou a implementação do sistema C-IL$^2$P completo, devido ao escopo desse sistema ser demasiadamente amplo. As partes relativas aos algoritmos de conversão para inserção do conhecimento e o refinamento do conhecimento por meio de aprendizagem foram implementadas. Comparativamente ao sistema apresentado as implementações avançaram no sentido apontado pelos passos (1), (2) e (3) da Figura 4.1.

## 4.2 O algoritmo C-IL$^2$P

A execução do algoritmo de conversão do sistema C-IL$^2$P parte de um programa em linguagem lógica **P**, convertendo-o na RNA **R**. Após o processo de aprendizagem, conforme descrito anteriormente, um novo programa em linguagem lógica é obtido **P'**, possuindo novos conhecimentos simbólicos. O algoritmo de conversão C-IL$^2$P em um nível de abstração mais alto[1] é dado por[2]:

1. Calcular o maior número entre a quantidade de literais e a quantidade de cláusulas com o mesmo conseqüente, o valor de ativação mínimo dos neurônios e os pesos das conexões dos neurônios.

[1] As expressões formais foram representadas verbalmente e resumidamente.

[2] A Seção A.1 do apêndice possui o algoritmo detalhado.

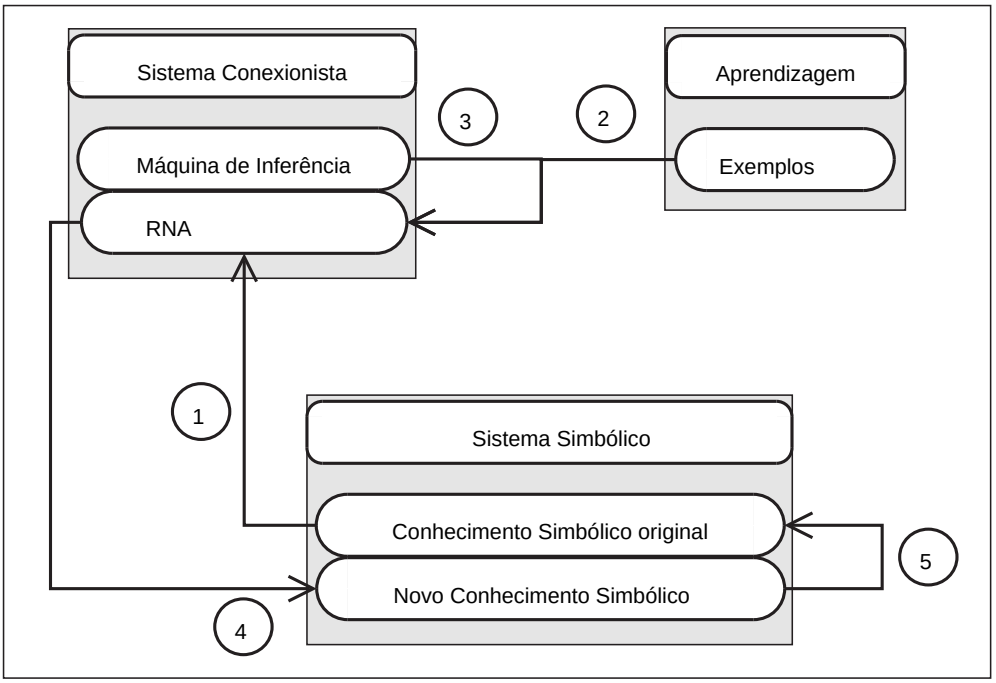


Figura 4.1: O sistema $C - IL^2P$ [12].

2. Converter cada cláusula do programa lógico em um neurônio na camada interna. Conectar cada neurônio da camada de entrada ao neurônio da camada interna criado. Conectar o neurônio criado com o neurônio de saída. Definir os limiares dos neurônios.

3. Definir as funções de ativação dos neurônios da camada de entrada. Dessa maneira, a interpretação do programa lógico será feita pela ativação conforme o vetor de entrada.

4. Definir as funções de ativação dos neurônios das camadas internas e de saída. Após isso, um algoritmo de aprendizagem pode ser aplicado.

5. A RNA deve estar totalmente conectada; atribui-se valor zero às demais conexões.

Esse algoritmo constitui a base para o estabelecimento do algoritmo O C-IL$^2$P Modal através da sua adaptação aos conceitos da Lógica Modal.

## 4.3 O algoritmo C-IL$^2$P Modal

Esta seção apresenta o algoritmo C-IL$^2$P Modal em alto nível[3]. Nesse algoritmo, existem as analogias entre várias RNAs e os mundos da Lógica Modal. As várias RNAs (sub-RNAs) são sub-redes que compõem uma RNA maior, que também pode representar um programa em linguagem lógica. O algoritmo original [11] foi expresso em linhas gerais segundo os passos abaixo:

1. Calcular os pesos modais a partir dos valores dos limiares de ativação e dos pesos, conforme calculados pelo algoritmo $C - IL^2P$ original (não-estendido).

[3]A Seção A.2 do apêndice contém o algoritmo detalhado.

2. Renomear os literais pela sua modalidade e aplicar o algoritmo $C - IL^2P$ original.

3. Criar os neurônios equivalentes aos literais possibilidade, estabelecer as respectivas conexões da RNA e os parâmetros internos ao neurônio.

4. Criar os neurônios equivalentes aos literais necessidade, estabelecer as respectivas conexões da RNA e os parâmetros internos ao neurônio.

5. Adicionar neurônios do tipo $\vee$ à sub-RNA, que contém o operador possibilidade, segundo a regra de inserção do operador possibilidade.

6. Adicionar neurônios do tipo $\wedge$ à sub-RNA, que contém o operador necessidade, segundo a sua regra de inserção.

## 4.4 Aplicando o algoritmo C-IL$^2$P Modal

A Figura 4.2 ilustra uma aplicação das regras, das relações entre mundos e a RNA gerada pelo algoritmo do C-IL$^2$P Modal, conforme exemplo apresentado no artigo original [11], transcrito como:

mundos: $\omega_1$, $\omega_2$ e $\omega_3$; Relações: $R(\omega_1, \omega_2)$ e $R(\omega_1, \omega_3)$.

$\omega_1$ : r $\rightarrow \Box$ q, $\quad \Diamond$s $\rightarrow$ r;
$\omega_2$ : s;
$\omega_3$ : q $\rightarrow \Diamond$p;

Essas regras podem ser lidas conforme segue:

- No mundo 1 ($\omega_1$): se existe "r", então existe a necessidade da existência de "q" e conseqüentemente "q" existe nos mundos com os quais o mundo 1 tem relação ($\omega_2$ e $\omega_3$).

- No mundo 1: a possibilidade da existência de "s" nos mundos com os quais o mundo 1 tem relação implica a existência de "r".

- No mundo 2 ($\omega_2$): "s" existe, é um fato.

- No mundo 3 ($\omega_3$): a existência de "q" implica a possibilidade da existência de "p".

Os mundos representam um escopo onde as regras são válidas. A quantidade de regras dentro desse escopo é variável, conforme a representação que se deseja. Um mundo somente irá influenciar em outro mundo, se existir alguma relação entre eles. A relação indica que um ou mais átomos são comuns aos mundos relacionados.

Novamente na Figura 4.2, as regras são convertidas em neurônios (círculos não-sombreados) diretamente pela aplicação do algoritmo C-IL$^2$P e os demais neurônios (círculos sombreados) são gerados pela aplicação do algoritmo de conversão do C-IL$^2$P Modal.

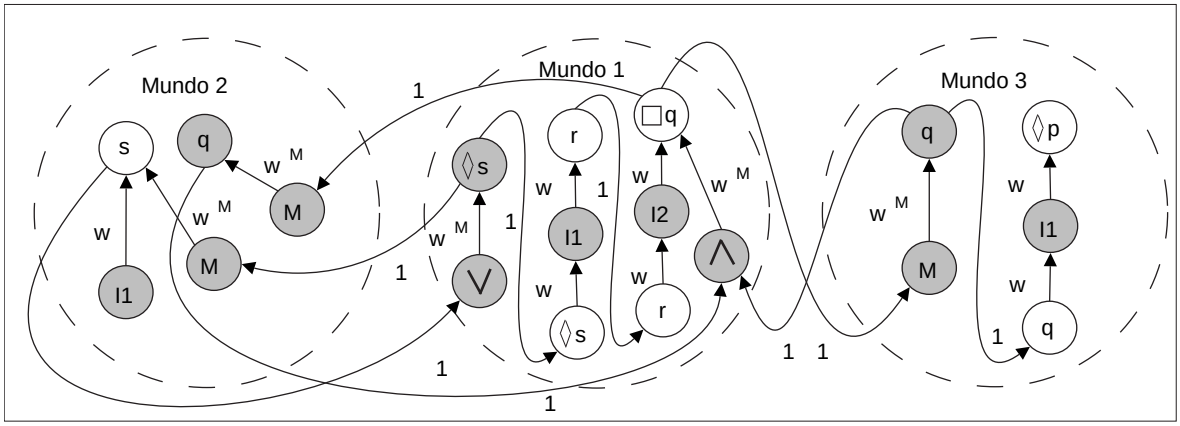


Figura 4.2: Exemplo de aplicação do algoritmo Modal.

### 4.4.1 Aplicação modal passo a passo

O exemplo apresentado na Figura 4.2 foi detalhado por passos, conforme é mostrado pela seqüência de figuras a seguir (da Figura 4.3 até a Figura 4.10). Percebemos o funcionamento do algoritmo do sistema C-IL$^2$P Modal, quando executamos passo a passo as etapas que o compõem, pela implementação feita neste trabalho. As setas pontilhadas que aparecem nas figuras identificam as ligações no passo que está sendo representado; as setas com linha cheia identificam as ligações criadas em passos anteriores. A mesma forma de representação é válida para os neurônios.

O algoritmo apresentado está em um nível de abstração mais baixo do que o mostrado na seção anterior (Seção 4.3), na qual o detalhamento objetivou a apresentação dos passos genéricos. Houve também uma melhora na organização, tornando sua forma mais apropriada à implementação deste algoritmo modal.

Algumas pequenas mudanças na ordem da execução de certas etapas do algoritmo foram feitas durante a implementação, devido a restrições, conveniências e otimizações, tais como: (1) os cálculos dos pesos foram feitos posteriormente à instanciação das sub-RNAs; (2) a aplicação dos procedimentos do operador necessidade precedem aos do operador possibilidade e (3) as conexões internas às sub-RNAs foram feitas no último passo.

1. Para cada cláusula da programação fazer (Figura 4.3):
   (a) Instanciar a(s) sub-RNA(s) equivalente(s).
   (b) Instanciar os neurônios correspondentes aos componentes das regras.
   (c) Identificar os neurônios segundo as modalidades lógicas.
   (d) Calcular e atribuir os menores valores de ativação ($A_{min}$'s), os valores dos pesos ($W$) e também dos limiares de ativação ($\theta$'s), segundo o algoritmo C-IL$^2$P (Figura 4.4).

2. Para cada neurônio possibilidade "de saída[4]" ($\Diamond$), que haja equivalência com outra sub-rede neural, segundo as relações existentes ($R$(sub-RNA$_i$;sub-RNA$_j$)), (Figura 4.5) fazer:

[4]"De saída" porque está relacionado ao neurônio de onde partirá a conexão para a outra sub-RNA.

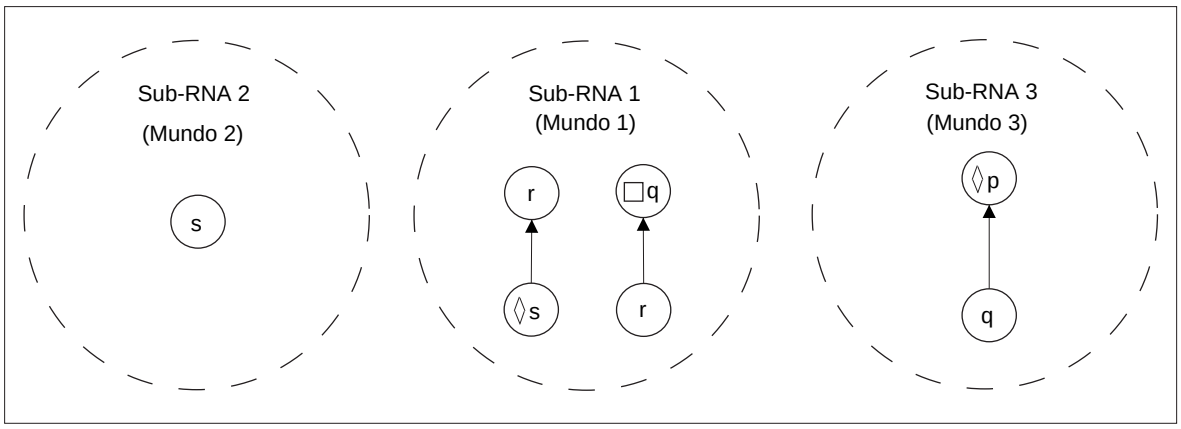


Figura 4.3: Aplicando o algoritmo C-IL$^2$P Modal - passo 1.

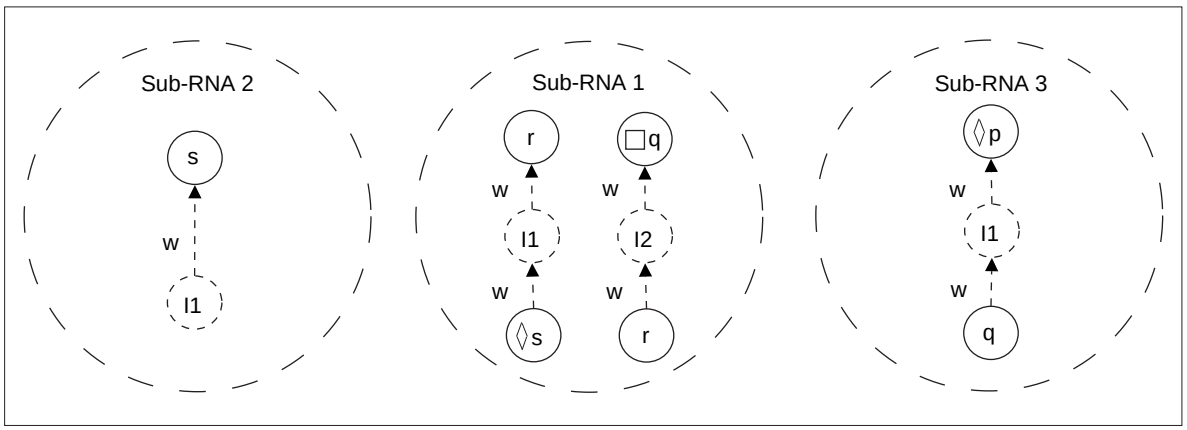


Figura 4.4: Aplicando o algoritmo C-IL$^2$P Modal - passo 2.

(a) Atender à regra de extração do operador possibilidade (◊E).

(b) Adicionar um Neurônio Modal (M) na sub-RNA de destino (sub-RNA$_j$).

(c) Calcular o Peso Modal ($W^M$), sendo que: $W^M > h^{-1}(A_{min}) + \mu * W + \theta_A$.

(d) Conectar, com peso de valor "1", o neurônio possibilidade ao neurônio modal e, com peso "$W^M$", o neurônio modal ao átomo correspondente.

(e) Atribuir os limiares dos neurônios modais ($\theta$'s com valores entre "-1" e $A_{min}$).

(f) Destinar a função degrau à função de ativação do neurônio modal.

3. Para cada neurônio necessidade "de saída" (□), onde exista a relação entre as sub-RNAs, (Figura 4.6) fazer:

(a) Atender à regra de extração do operador necessidade ( E), implica a adição de novos neurônios onde eles não existam.

(b) Adicionar neurônios modais (M) que satisfaçam a relação.

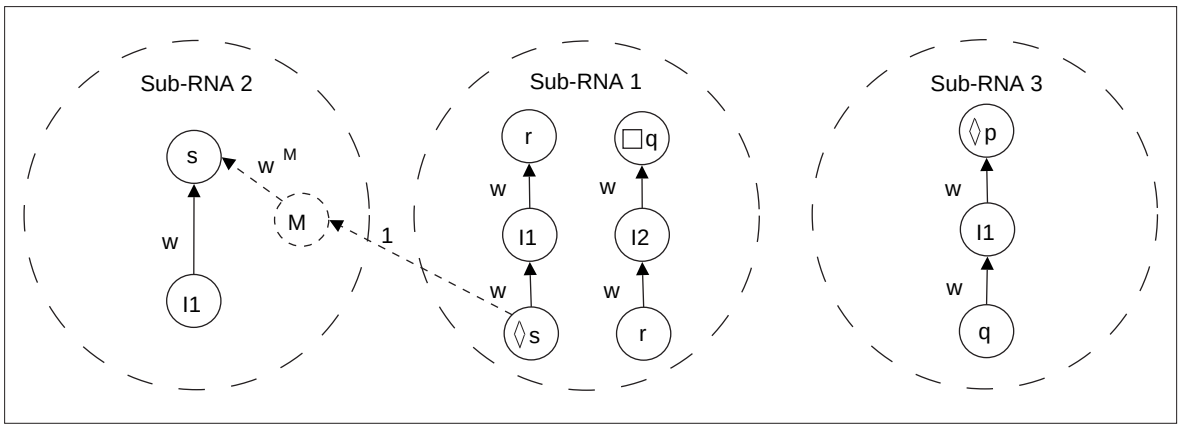


Figura 4.5: Aplicando o algoritmo C-IL$^2$P Modal - passo 3.

(c) Atribuir peso "1" nas conexões entre as sub-RNAs e peso "$W^M$" nas conexões internas as sub-RNAs.

(d) Atribuir os limiares dos neurônios modais ($\theta$'s com valores entre "-1" e "$A_{min}$)".

(e) Destinar a função degrau à função de ativação do neurônio modal.

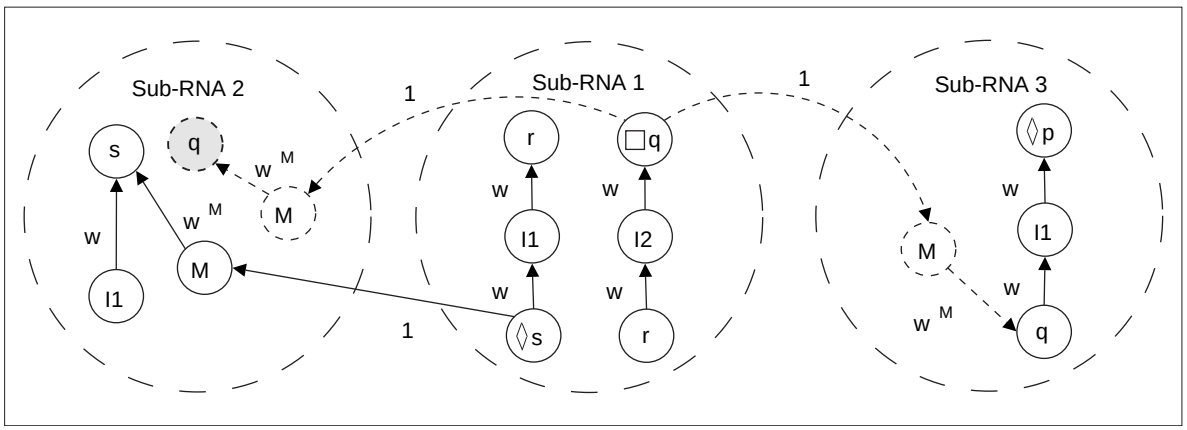


Figura 4.6: Aplicando o algoritmo C-IL$^2$P Modal - passo 4.

4. Para cada neurônio das sub-RNAs onde haja relação e não ocorreu a extração de operadores conforme os passos anteriores, (Figura 4.7) fazer:

(a) Atender à regra de inserção do operador necessidade ( I).

(b) Adicionar neurônios modais ($\wedge$) que satisfaçam a relação e a regra.

(c) Atribuir à função de ativação do(s) neurônio(s) "$\wedge$" a função degrau.

(d) Conectar as sub-RNAs com peso de valor "1". As conexões internas às sub-RNAs possuem pesos de valor "$W^M$".

(e) Atribuir o(s) limiar(es) do(s) neurônio(s) "$\wedge$" ($\theta$'s com valores entre "$(n-(1+A_{min}))$" e "$n * A_{min}$").

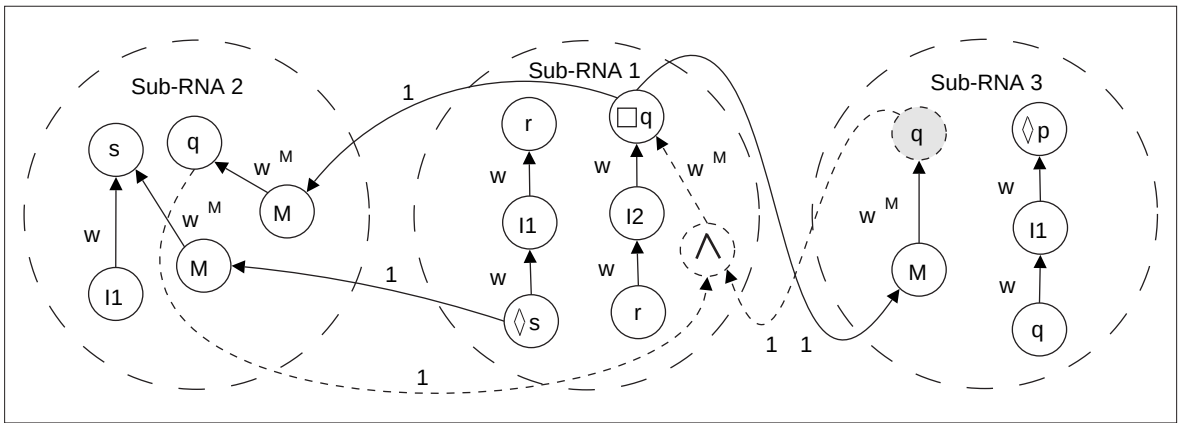


Figura 4.7: Aplicando o algoritmo C-IL$^2$P Modal - passo 5.

5. Para cada neurônio das sub-RNAs onde haja relação e não ocorreu a extração de operadores e nem a inserção do operador necessidade, conforme os passos anteriores, (sub-RNA 2 para a sub-RNA 1, na Figura 4.8) fazer:

   (a) Atender à regra de inserção do operador possibilidade ($\Diamond$I).

   (b) Adicionar neurônios modais ($\vee$) que satisfaçam a relação e a regra.

   (c) Atribuir à função degrau a função de ativação do(s) neurônio(s) "$\vee$".

   (d) Conectar, com peso de valor "1", as sub-RNAs e peso de valor "$W^M$", as conexões internas às sub-RNAs.

   (e) Atribuir o(s) limiar(es) do(s) neurônio(s) "$\vee$" ($\theta$'s com valores entre "$-n * A_{min}$" e "$(n * A_{min} - (n-1))$").

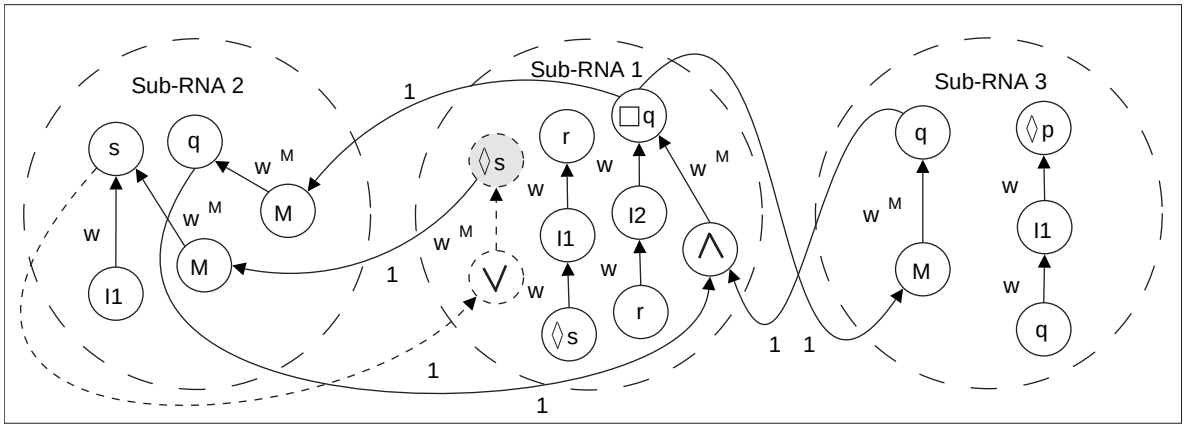


Figura 4.8: Aplicando o algoritmo C-IL$^2$P Modal - passo 6.

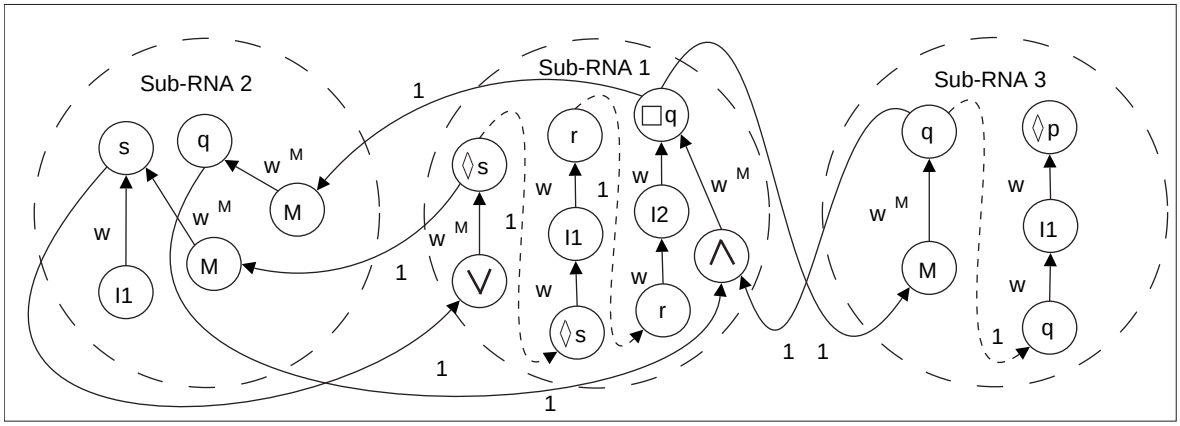


Figura 4.9: Aplicando o algoritmo C-IL$^2$P Modal - passo 7.

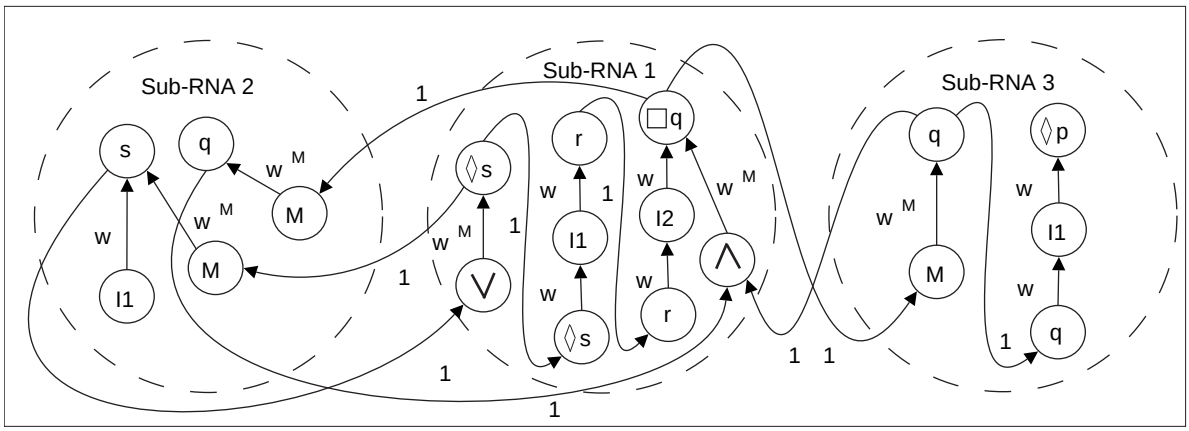


Figura 4.10: RNA gerada pelo algoritmo C-IL$^2$P Modal.

O sistema das três sub-RNAs (Figura 4.9) é finalizado pelo acréscimo dos laços internos de realimentação, que, por definição, recebem o valor de peso unitário (peso igual a "1"). Obtemos, então, o conjunto de sub-RNAs da Figura 4.10. Isso significa que são o mesmo neurônio, pois, se sua entrada for ativada, a sua saída também estará ativa. Observa-se ainda que um dos efeitos dessa separação estrutural é a propagação de um impulso em tempos diferentes pela RNA.

### 4.4.2 Outros exemplos de aplicações do algoritmo C-IL$^2$P Modal

Exemplos de aplicação do algoritmo C-IL$^2$P Modal são encontrados na continuação dos trabalhos de Artur S. d'Avila Garcez, Luís C. Lamb, Krysia Broda e Dov M. Gabbay [10]. Nessa continuação dos trabalhos, os conceitos da INS são aplicados em sistemas de agentes cognitivos interagindo entre si.

Esses sistemas podem ser modelados pela Lógica Epistêmica, com evolução temporal das situações representadas pelo sistema. Nos apêndices B.1, B.2 e B.3, existe uma explicação resumida das aplicações citadas [10, 15], assim como uma análise de suas evoluções temporais.

# Capítulo 5

# Apresentação do trabalho desenvolvido

"*Escolha um trabalho que você ame e não terá que trabalhar um único dia de sua vida.*"
Confúcio (551 a.C. - 479 a.C.) - Filósofo chinês

Este capítulo apresenta a síntese do trabalho, segundo os softwares desenvolvidos, fornecendo um roteiro das diretrizes seguidas. O Sistema Computacional proposto está esquematizado na Seção 5.2, onde são mostradas a sua divisão em blocos e as inter-relações existentes entre eles. A Seção 5.3 contém algumas peculiaridades relevantes das implementações.

## 5.1 Estudo e desenvolvimento inicial

### 5.1.1 Recursos e estratégias utilizadas no desenvolvimento

O sistema "Windows" foi adotado como sistema operacional para o desenvolvimento dos aplicativos, por ser um sistema bastante difundido e familiar, no entanto, nada impede que haja portabilidade dos aplicativos para outros sistemas operacionais. A plataforma de desenvolvimento é o software "Dev-C++" da "Bloodshed" [22]. A linguagem de programação C++ foi a escolhida para as implementações dos programas desenvolvidos, devido à facilidade na orientação a objeto, o que possibilitou a replicação de blocos (sub-RNAs) nas RNAs.

A estratégia utilizada na implementação dos algoritmos distingue a abordagem segundo a Inteligência Artificial Simbólica (implementações de conversões simbólicas baseadas na Lógica) e a Conexionista (implementações enfocavam as RNAs). Como último ponto da estratégia, houve a integração dos dois paradigmas, através das implementações dos algoritmos Neuro-Simbólicos.

O método utilizado para a implementação foi baseado em testes, primeiramente focando problemas específicos, para, logo em seguida, desenvolver programas que satisfaçam os requisitos da classe desses problemas. O último passo foi a ampliação do programa para o atendimento de outras classes de problemas, tornando-o mais genérico.

### 5.1.2 Etapas do desenvolvimento

Inicialmente, objetivou-se a implementação do algoritmo empregado pelo sistema C-IL$^2$P, dividindo-a em duas subetapas distintas: A implementação relativa ao programa em linguagem

lógica (IAS) e a implementação da Rede Neural Artificial (IAC) correspondente à linguagem lógica.

Iniciou-se com programas para RNAs aplicadas a problemas simples e específicos, para, logo em seguida generalizá-los. Na seqüência, ocorreu o desenvolvimento de programas para a conversão de Linguagens Lógicas, apresentadas sob a forma simbólica. Após a integração entre as duas partes, ocorre a aplicação a um problema mais extenso e específico que os anteriores e o acompanhamento e avaliação do desempenho obtido pela RNA final.

Na fase final do desenvolvimento, foi criado um banco de dados artificial para ser utilizado nos aprendizados das RNAs. Nessa fase, também foram implementados alguns exemplos em Lógica Modal utilizando-se o algoritmo C-IL$^2$P Modal.

O desenvolvimento do trabalho pode ser acompanhado pelo enfoque existente nas etapas que o dividiram. Foram relacionadas 12 etapas pela sua ordem cronológica no desenvolvimento, nas quais se destacaram os respectivos enfoques e descrição, conforme segue:

- `Etapa 1 (RNA específica):` - Implementações de RNAs simples, permitindo a primeira definição das técnicas de implementação de RNA a serem empregadas.
- `Etapa 2 (Função logística):` - Estudo das funções matemáticas sigmóides e suas derivadas.
- `Etapa 3 (RNA genérica):` - Versões mais aprimorada do programas de RNAs específicas, com refinamento das técnicas.
- `Etapa 4 (Interface gráfica):` - Estudo preliminar para integrar todos os programas executáveis, para o gerenciamento do sistema C-IL$^2$P em um ambiente computacional.
- `Etapa 5 (Algoritmo CIL`$^2$`P):` - Estruturação das classes do sistema C-IL$^2$P e desenvolvimento do método que engloba rotinas para o processamento da sintaxe de textos.
- `Etapa 6 (Conversão de dados):` - Formatação dos dados a serem aplicados em uma RNA específica, definição dos formatos dos arquivos.
- `Etapa 7 (Algoritmo CIL`$^2$`P):` - Aprimoramento dos programas anteriores permitindo a inserção das regras aplicáveis de forma genérica, segundo o algoritmo C-IL$^2$P.
- `Etapa 8 (Algoritmo CIL`$^2$`P):` - Aplicação do algoritmo C-IL$^2$P a exemplos específicos.
- `Etapa 9 (Base de dados):` - Criação da base de dados baseada em uma regra lógica.
- `Etapa 10 (Algoritmo CIL`$^2$`P):` - Comparação entre RNAs com e sem o emprego do algoritmo C-IL$^2$P.
- `Etapa 11 (Algoritmo CIL`$^2$`P):` - Testes das RNAs com utilização do algoritmo C-IL$^2$P.
- `Etapa 12 (Algoritmo CIL`$^2$`P Modal):` - Desenvolvimento do ambiente computacional que contém os recursos estudados nos programas precedentes, incluindo a implementação do C-IL$^2$P Modal.

As etapas 1 e 5 foram importantes para a fundamentação dos seus conceitos. A partir delas, podemos destacar duas implementações preliminares importantes para a continuidade dos trabalhos, vistas na subseção 5.1.3. Nas etapas 2 e 4 foram realizados estudos que definiram a função matemática mais apropriada e como a interface deveria ser implementada. As etapas 3, 6 e 9 relacionam as implementações aplicadas a exemplos específicos. As etapas 7, 8, 10 e 11 focam o algoritmo CIL$^2$P e suas aplicações. Finalizamos com a etapa 12, que contém o estudo e aplicação do CIL$^2$P Modal.

### 5.1.3 Implementações preliminares

Algumas implementações básicas foram necessárias para se adquirir experiência, servem como exemplo da aplicação distinta dos paradigmas IAC e IAS. O conhecimento adquirido com essas implementações ajudou a composição dos blocos das implementações subseqüentes. Podemos destacar abaixo as duas implementações iniciais importantes:

(a) Utilização de exemplos variados para o treinamento das RNAs.
(b) Emprego de regras lógicas simplificadas.

#### Redes Neurais Artificiais simples

Os primeiros programas desenvolvidos neste trabalho utilizavam a topologia básica da RNA com três camadas e com poucos neurônios por camada (por exemplo: 5x3x1 ou 2x2x1). Nesse caso, pretendia-se fazer somente o treinamento da RNA, verificação de cálculos e teste estrutural da arquitetura.

Alguns exemplos simples foram utilizados, como o da função booleana XOR [29] e o de reconhecimento e classificação de formas (padrões sensíveis à RNA). Na classificação de formas, um "*string*", ou conjunto de caracteres, é passado como entrada aos neurônios da primeira camada da RNA. Um padrão sensível à RNA pôde ser definido a partir de uma imagem básica proveniente de uma matriz de caracteres. Como exemplo do reconhecimento de imagens formadas por "strings", temos a assinatura (ou modelo) de um peixe apresentado na Figura 5.1, definido pelo caracter "#".

A figura é composta por uma matriz contendo 11 linhas com 25 caracteres em cada linha, constituindo um modelo de peixe para a RNA. Variações desse modelo poderiam servir de exemplo para o treinamento da RNA. Um exemplo positivo decorre de uma forma que não é muito diferente da assinatura do peixe, como um peixe mais magro ou menor. Por outro lado, se um outro formato fosse lido pela RNA, tal como uma esfera, essa seria classificada como um exemplo negativo pelo modelo.

As estruturas matriciais foram construídas somente no sentido direto da RNA. As matrizes da RNA continham os valores dos neurônios e das conexões entre eles. Os valores matriciais de retropropagação que existem nos demais programas, nesse caso, devido à simplicidade, foram calculados através de variáveis contendo valores originados pela aplicação direta da Regra Delta (equação 2.5).

A criação de uma interface para entrada de dados fez-se necessária com a migração para programas mais genéricos. A solução mais imediata foi a adoção de arquivos formatados contendo um conjunto de definições, que são alteráveis no programa antes da sua execução.

#### Leitura de programas lógicos

A leitura de programas lógicos foi útil para o estudo inicial da implementação do algoritmo de conversão C-IL$^2$P e da computação de sistemas convergentes. A execução, o acompanhamento e os testes dessa implementação permitiram verificar a tradução de um programa em linguagem lógica (Programa Lógico) em valores calculados segundo o C-IL$^2$P e também verificar a estabilidade do Programa Lógico.

A Figura 5.2 representa duas regras lógicas (regras 5.1 e 5.2). A partir dessas duas regras lógicas, foram obtidas as estruturas das RNAs similares às da Figura 5.2, aplicando-se a elas os cálculos requeridos pelo algoritmo, tais como a determinação do limiar de ativação mínimo e do

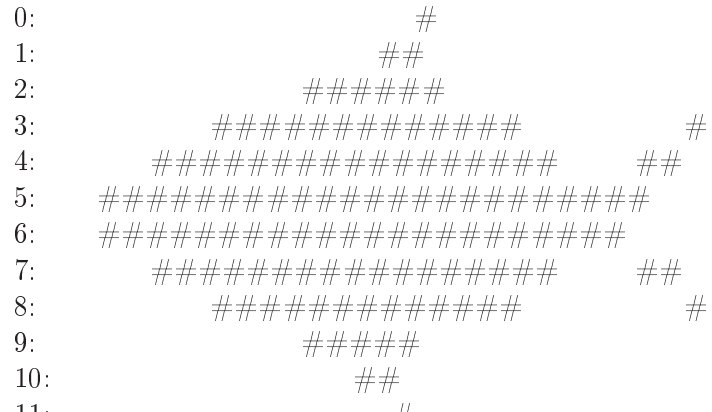


Figura 5.1: *Strings* que representam a assinatura de um peixe.

valor mínimo de peso de rede aceitável. Durante a convergência, as saídas das regras ("e" e "j") atingiram a estabilidade em poucas iterações. As entradas da RNA, nesse caso, eram valores binários atribuídos a "a, b, c, d, f, g, h e i". Os valores de convergência foram apresentados em ponto flutuante, resultado dos cálculos do algoritmo.

O arquivo-texto utilizado continha as cláusulas lógicas descritas abaixo:

$$a * b + c * d \rightarrow e \tag{5.1}$$

$$f * g + h * i \rightarrow j \tag{5.2}$$

As duas cláusulas contêm cinco átomos cada uma, quatro dos quais estão distribuídos em dois literais, nos seus antecedentes. O conseqüente de cada cláusula possui apenas um literal contendo um átomo. Durante a execução dessa implementação, foi possível alterar os números de átomos, de literais e também suas posições, quando se adicionou recursividade entre os átomos, por exemplo, o conseqüente "e" substituiu o antecedente "f".

## 5.2 O sistema desenvolvido

O enfoque principal do sistema desenvolvido foi a implementação do algoritmo de conversão unidirecional do sistema C-IL$^2$P estendido por modalidades. A modelagem da interface utilizada iniciou-se a partir do desenvolvimento de um editor de arquivos, com o objetivo de edição e salvamento de arquivos utilizados pelo programa. Essa interface também permite que se criem novos arquivos de informações.

### 5.2.1 Aspectos internos do sistema

#### Classes que compõem o programa

A Figura 5.3 apresenta um esquema das classes nas quais o ambiente computacional teve o seu desenvolvido baseado. Utilizamos uma classe-suporte para o trabalho com a plataforma "Windows", com atributos e métodos padronizados. A classe "Windows" instancia a classe para

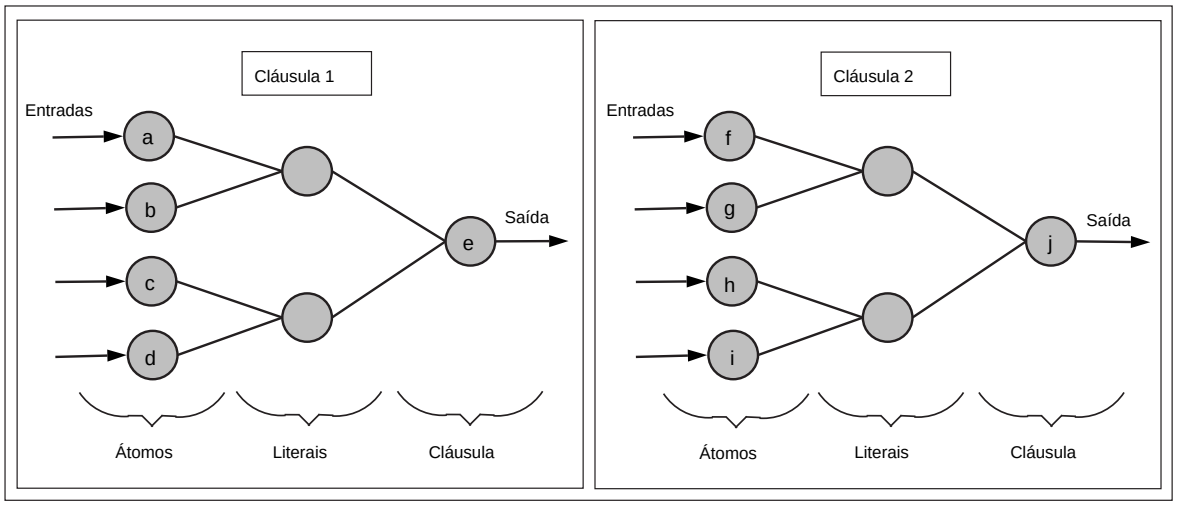


Figura 5.2: Representação da RNA gerada por cláusulas lógicas.

tratamento da programação em linguagem lógica (Classe da Programação Lógica), conforme as opções que lhe são passadas.

A classe da Programação Lógica tem como dependentes as classes de instanciação das Redes Neurais Artificiais e da Aprendizagem dessa. Por sua vez, a classe que trata da RNA pode ser derivada em RNAs menores, ou sub-RNAs, que possuem seus métodos agregados aos de aprendizagem.

### Casos de Uso previstos

A Figura 5.4 contém alguns casos mais comuns de uso do ambiente computacional. Um especialista do conhecimento pode valer-se desse ambiente para aplicá-lo a um problema específico. O objetivo de aplicação pode ser: o aumento na abrangência de regras associadas ao conhecimento inicial do especialista. Cada um dos casos de uso se refere a uma etapa existente neste ciclo, proposto para o aprimoramento das regras provenientes do conhecimento.

### Seqüência de eventos

O diagrama da Figura 5.5 aponta para a seqüência de operações iniciadas pelo usuário, que basicamente, interage através da configuração contida nos arquivos e na interface ou dando início aos processos. O software interno faz as consultas ao banco de dados ou a outros arquivos contendo as informações necessárias ao desenvolvimento de seus processos.

Como ilustra a Figura 5.5, o software interno atua na modificação da estrutura da RNA geral, quer seja pelo estabelecimento da sua topologia, quer na alteração dessa em conseqüência da aprendizagem. Uma vez que as operações estejam encerradas, o usuário pode consultar os arquivos produzidos e verificar os resultados.

Os passos à execução do programa podem ser resumidos desta forma:

1. Seleção dos algoritmos, das operações, dos arquivos de dados.

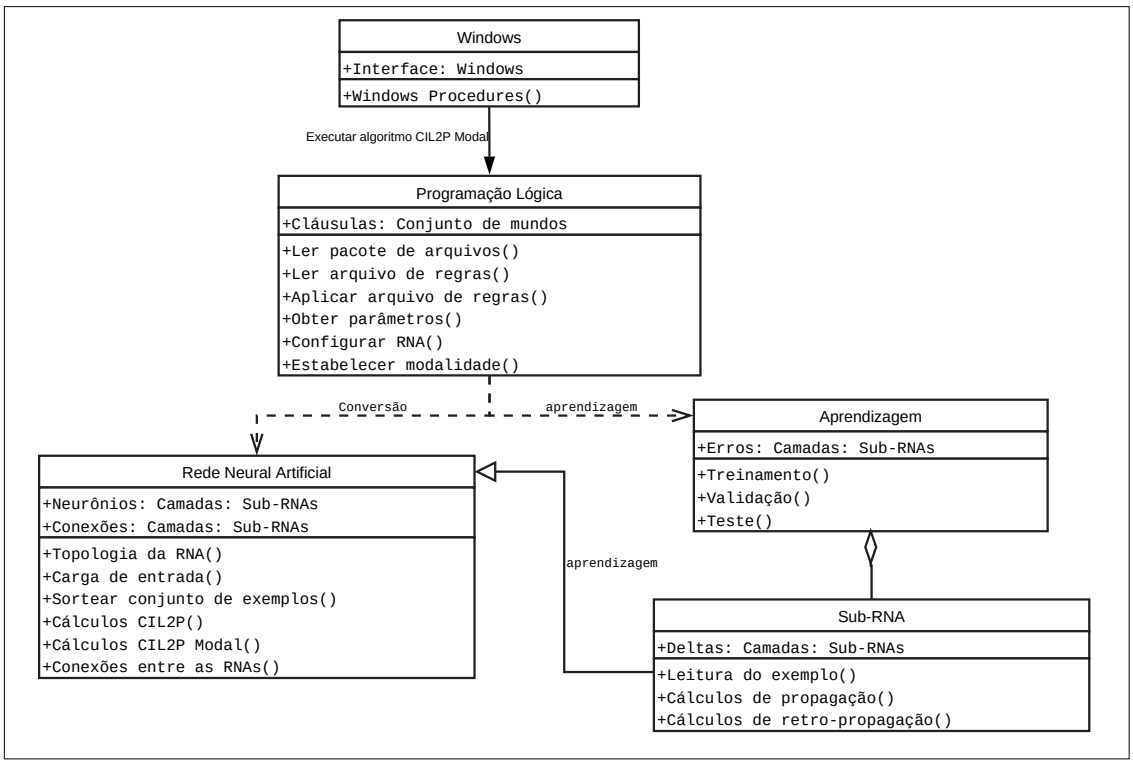


Figura 5.3: Diagrama de Classes do sistema.

2. A partir dos arquivos de informações inseridos, um conjunto de dados internos de trabalho é produzido pelo programa. Nesse conjunto de dados, podem ser lidos átomos, os mundos aos quais eles pertencem, as devidas posições e recursividades entre eles.

3. Os cálculos são executados baseados nas informações do passo 2 e se estabelecem a topologia inicial da RNA e suas sub-RNAs, havendo o seu acompanhamento.

4. Opcionalmente, novas informações podem ser obtidas a partir da leitura de exemplos, durante o aprendizado, alterando e preparando as RNAs aos processos de inferências.

5. Inicia-se a etapa de dedução lógica. Busca-se a computação convergente do programa, que se reflete na procura da estabilidade do sistema.

### Processos do sistema

Na Figura 5.6, podemos acompanhar, de outra forma, as etapas apresentadas nos casos de uso do sistema (Figura 5.4). Também temos a visão interna da seqüência de processos e a atividade de inter-relação entre as classes (Figura 5.3). Primeiramente, um programa em linguagem lógica será inserido e, conforme a configuração, ocorre o seu processamento inicial.

Existem dois processos relativos à estrutura que entram em operação após a leitura dos dados iniciais: as RNAs são instanciadas e as suas topologias iniciais são fixadas. Logo após, iniciam-se

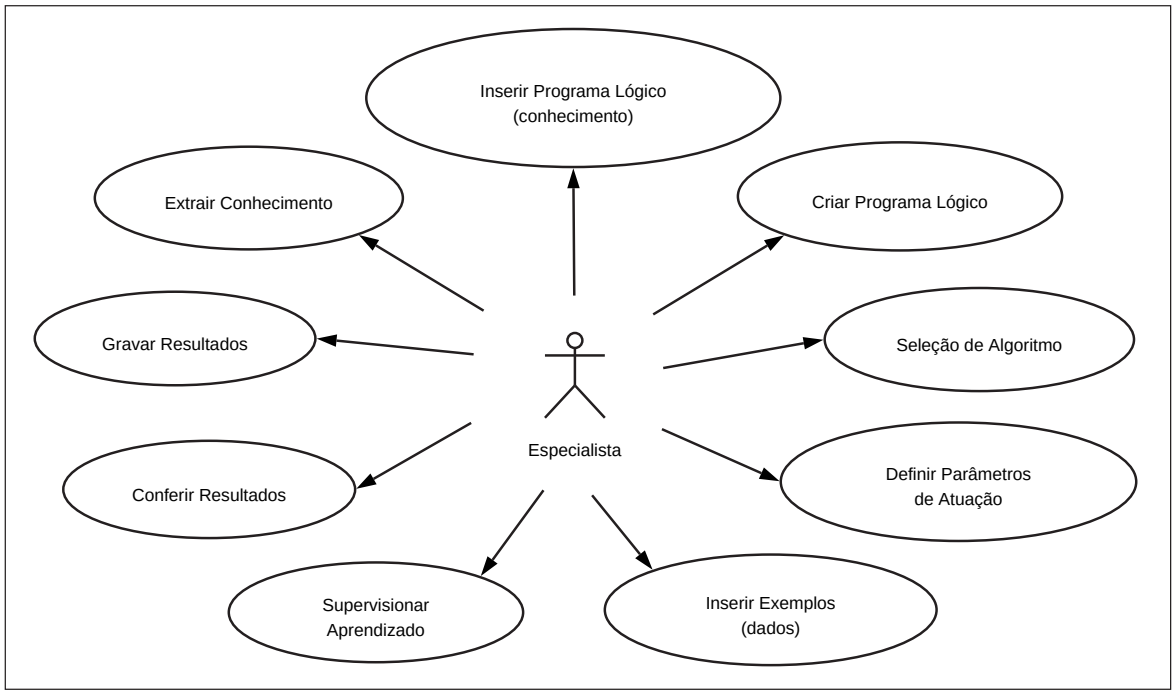


Figura 5.4: Diagrama de Casos de Uso do sistema.

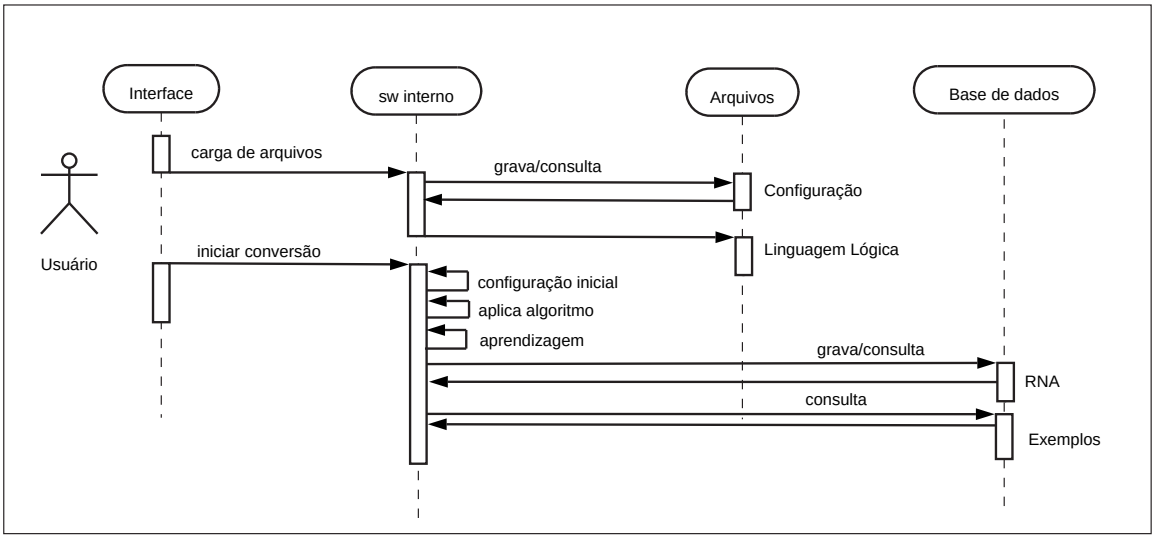


Figura 5.5: Diagrama de Seqüência do sistema.

os processos de aprendizagem das RNAs, que podem alterar a forma original da estrutura da rede geral.

Após o processamento da programação, instanciam-se as RNAs correspondentes, assim como os cálculos relativos ao algoritmo em operação. A etapa de aprendizagem das RNAs é executada logo em seguida, finalizando pela obtenção de resultados. Os resultados gerados pelo programa

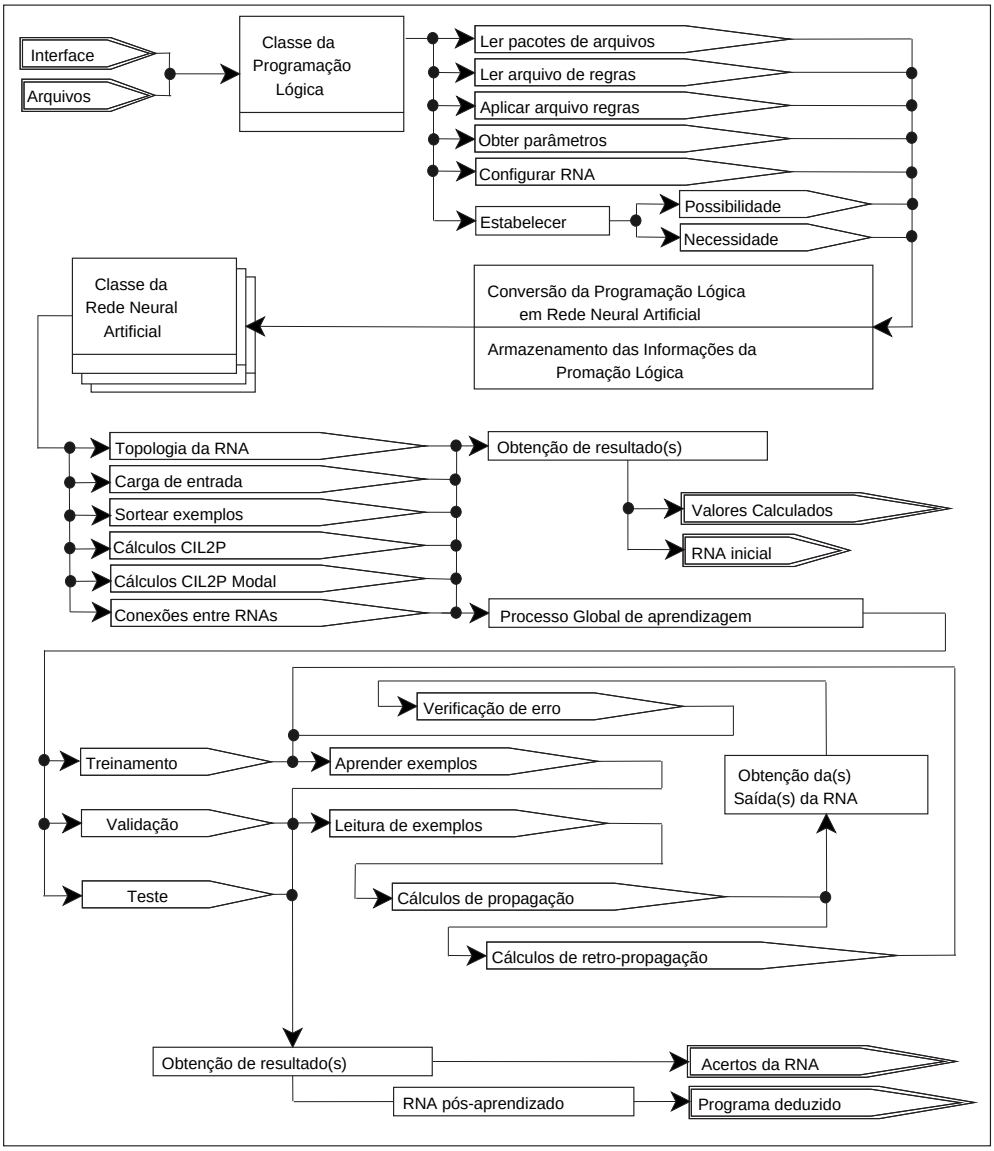


Figura 5.6: Diagrama de Processos do sistema.

são as RNAs após o processos de aprendizagem e o percentual de acerto obtido individualmente pelas RNAs.

A extração do conhecimento obtido, originando novas regras, fugiu ao escopo deste trabalho, embora isso possa ser feito de modo não-automático pelo especialista. Observa-se que um sistema automático completo agrega um refinamento mais aprimorado, segundo critérios de um especialista, com a extração do conhecimento.

### 5.2.2 Descrição da interface

A interface também contém um menu para seleção do algoritmo a ser usado, assim como as suas opções de operação dos algoritmos. A aparência da interface está ilustrada pela Figura 5.7. A

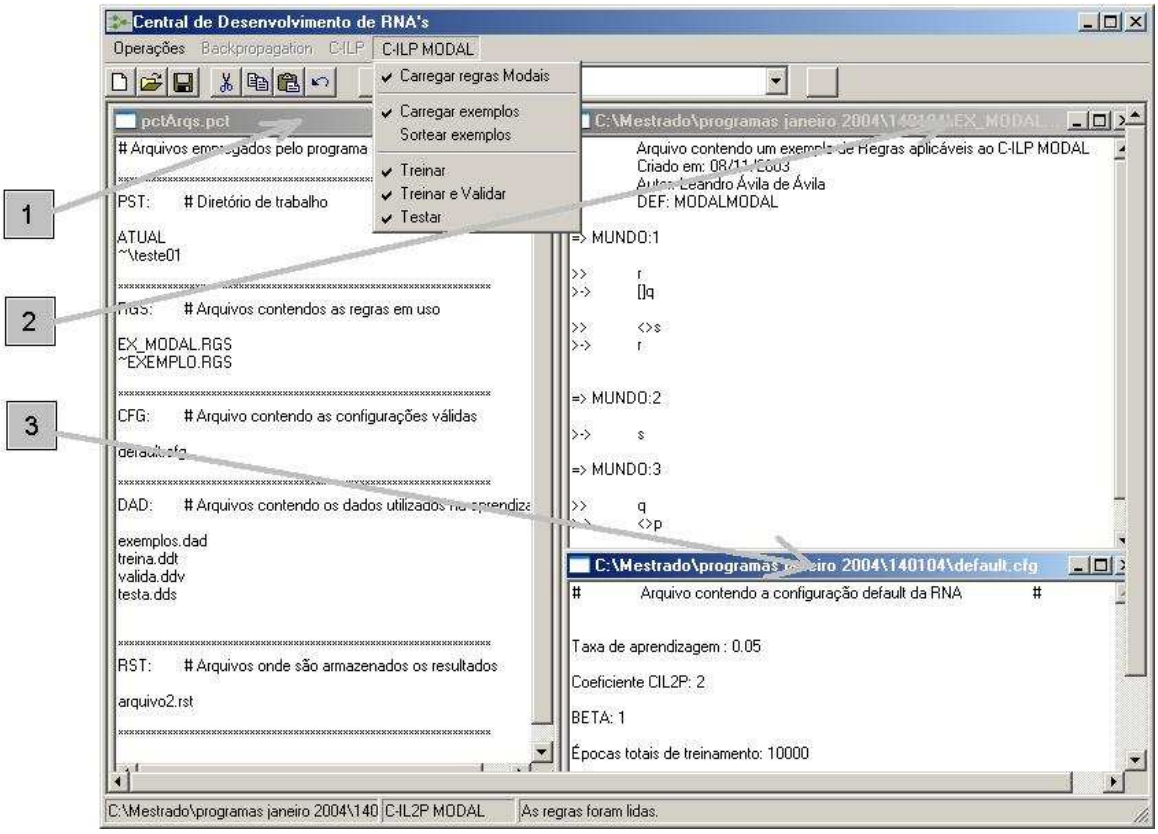


Figura 5.7: Interface do Ambiente Computacional.

descrição da interface segue nesta seção.

### Painel de edição

Os três principais arquivos, conforme círculos na Figura 5.7, editáveis pela interface são:

1. O arquivo que contém os nomes dos arquivos de trabalho do ambiente computacional (pacote de arquivos).
2. O arquivo que contém o program lógico a ser trabalhado.
3. O arquivo de configuração, que contém as informações sobre como a RNA deve se comportar durante o aprendizado.

Esses três arquivos são abertos e acessados diretamente no painel interno da interface. Não existe restrição quanto ao tipo de arquivo a ser editado. Arquivos como os de dados a serem utilizados pela RNA também podem ser trabalhados.

### Barra do menu

Na barra de menu, temos as opções descritas na Tabela 5.1. As subopções relacionadas ao tipo de algoritmo escolhido funcionam como pontos de checagem durante a execução do programa.

Haverá a execução da rotina correspondente no programa sempre que a subopção for selecionada e os processos de aprendizagem forem acionados.

Tabela 5.1: Menu de opções da interface.

| **Opção** | **Subopção** | **Descrição** |
|---|---|---|
| Operações | Pacote de Arquivos...Definir | Abre o lote de arquivos de trabalho para a edição. |
| | Pacote de Arquivos...Carregar | Carrega no ambiente os arquivos de trabalho. |
| | Pacote de Arquivos...Salvar | Salva o pacote de arquivos em uso. |
| | Editar Arquivos...Regras | Abre arquivo para edição da programação lógica. |
| | Editar Arquivos...Configuração | Abre arquivo para editar a configuração da RNA. |
| | Editar Arquivos...Exemplos | Abre arquivo de exemplos para a RNA. |
| | Algoritmo...Backpropagation | Aprendizagem tradicional da RNA. |
| | Algoritmo...C-IL2P | Aprendizagem aplicando-se o algoritmo C-IL$^2$P. |
| | Algoritmo...C-IL2P MODAL | Aprendizagem aplicando-se o algoritmo C-IL$^2$P Modal. |
| Back-propagation | Criar nova Rede Neural | Os pesos da RNA são sorteados aleatoriamente. |
| | Carregar Rede Neural | Ao invés de sortear os pesos, ele os lê em arquivo. |
| | Salvar Rede Neural | O arquivo atual de pesos é salvo. |
| | Carregar exemplos | Os exemplos utilizados no treinamento, na validação e no teste são carregados no ambiente computacional. |
| | Sortear exemplos | Os exemplos são distribuídos aleatoriamente entre os processos de treinamento, validação e teste. |
| | Treinar | Executa o processo de treinamento da RNA. |
| | Treinar & Validar | Executa o processo de validação da RNA após o teste. |
| | Testar | Executa o processo de teste da RNA final. |
| C-IL2P | Carregar regras | O arquivo contendo a programação lógica é lido. |
| | Carregar exemplos | Os exemplos utilizados no treinamento, na validação e no teste são carregados no ambiente computacional. |
| | Sortear exemplos | Os exemplos são distribuídos aleatoriamente entre os processos de treinamento, validação e teste. |
| | Treinar | Executa o processo de treinamento da RNA. |
| | Treinar & Validar | Executa o processo de validação da RNA após o teste. |
| | Testar | Executa o processo de teste da RNA final. |
| C-IL2P MODAL | Carregar regras | A programação em Lógica Modal é lida. |
| | Carregar exemplos | Os exemplos utilizados no treinamento, na validação e no teste são carregados no ambiente computacional. |
| | Sortear exemplos | Os exemplos são distribuídos aleatoriamente entre os processos de treinamento, validação e teste. |
| | Treinar | Executa o processo de treinamento da RNA. |
| | Treinar & Validar | Executa o processo de validação da RNA após o teste. |
| | Testar | Executa o processo de teste da RNA final. |

### Barra de botões

A barra de botões da interface facilita algumas operações de edição e execução dos programas. Os espaços (botões ou janela) da barra de botões foram descritos na Tabela 5.2.

Tabela 5.2: Barra de botões da interface.

| Grupo | Botão | Descrição |
|---|---|---|
| Arquivo | Cria | Cria um novo arquivo. |
| | Abre | Abre um arquivo específico. |
| | Salva | Salva o arquivo da janela ativa. |
| Edição | Corta | Corta um texto selecionado no arquivo. |
| | Copia | Copia o texto selecionado. |
| | Cola | Cola o texto que foi copiado. |
| | Desfaz | Desfaz a operação anterior. |
| Gráfico | Exibir | Exibi janelas gráficas. |
| Execução | Executar | Inicia o processo selecionado na janela. |

**Barra de "status"**

A interface também conta com uma barra localizada abaixo da janela principal que serve para o acompanhamento do andamento das operações. A chamada barra de *status* foi dividida em três regiões: a primeira contém os arquivos que foram carregados e estão sendo trabalhados, a segunda, o algoritmo em execução, e a terceira salienta as etapas do processamento.

## 5.3 Considerações sobre as implementações

### 5.3.1 RNAs

A maioria dos programas desenvolvidos empregaram as RNAs com o seu aprendizado executado por meio do algoritmo de retropropagação. A escolha desse algoritmo deveu-se à sua utilização no sistema C-IL$^2$P [12], possibilitando a comparação de resultados. Os algoritmos que definiram as topologias de RNAs empregadas foram conseqüência da evolução natural das técnicas de programação.

Os valores neurais e as sinapses da RNA foram implementados por matrizes de ponteiros. Quando desejamos acessar o valor contido em um neurônio específico, indexamos os ponteiros correspondentes, pois os índices de cada ponteiro possibilitam o acesso à camada e aos neurônios dessa camada. Na Figura 5.8, vemos a representação por estruturas matriciais com as quais podemos acessar valores da RNA por ponteiros, durante a leitura dos $n$ exemplos. Os dois tipos de ponteiros são descritos abaixo:

→ `valor do neurônio[k][i]`
→ `valor do peso[k][i][j]`

Neles, $k$, $i$ e $j$ representam, respectivamente, o número da camada, o número do neurônio da camada indicada e o número do neurônio na próxima camada (neurônio de destino da conexão).

Através das matrizes representaram-se as regras lógicas fornecidas pela programação simbólica, regras essas que equivalem as RNAs. É possível o compartilhamento da mesma estrutura de dados entre as classes às quais cada estrutura pertence e também entre os valores de propagação e retropropagação.

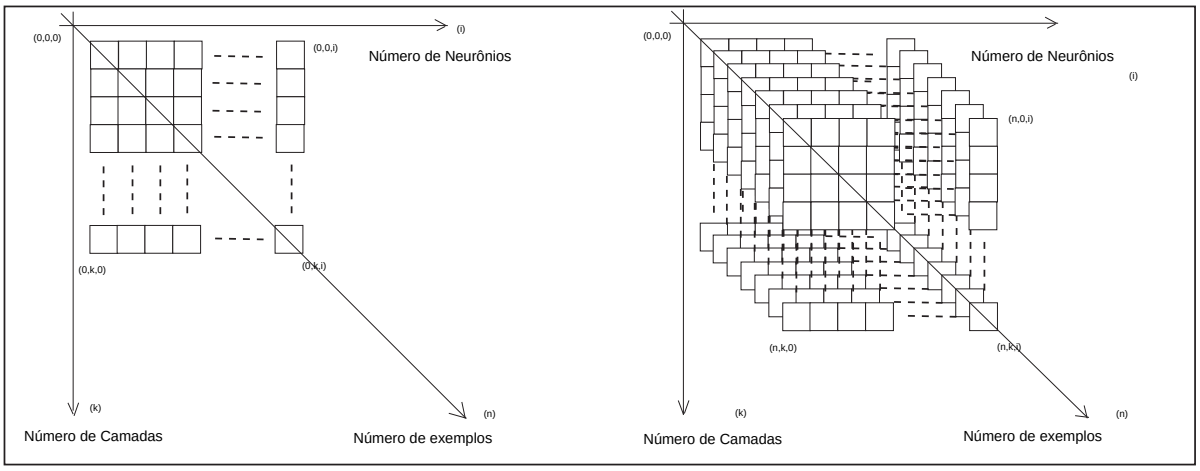


Figura 5.8: Representação matricial das camadas e conexões.

### 5.3.2 Entradas do Sistema

#### Arquivos

Os arquivos que normalmente são utilizados pelos programas dividem-se em cinco grupos:

1. Arquivos contendo a configuração da RNA e informações de controle do programa.
2. Arquivos contendo os dados de entrada para a RNA (treinamento, validação e teste).
3. Arquivos com as informações sobre a arquitetura da RNA, normalmente os valores dos pesos e as conexões ativas.
4. Os arquivos com o programa lógico.
5. Outros arquivos gerados durante a execução dos programas, e.g., arquivos de avaliação de resultados e de registro (".log").

#### Programação lógica

Os resultados dependem diretamente da programação lógica de entrada ao sistema. Por isso, esse item é dedicado à descrição de como foi definida a programação. A programação lógica de entrada pode ser expressa pelo conjunto de caracteres que representam os átomos e os operadores lógicos, formando uma sintaxe e se relacionando de acordo com a semântica lógica a ser descrita.

Nós nos valeremos de um programa simples somente para expor o processo sobre uma programação básica, com um mundo apenas e duas cláusulas. Sendo duas cláusulas quaisquer, como exemplo 5.3 e 5.4, as quais são processadas pelo programa, resultam em uma programação lógica indexada segundo a ordem de ocorrência de cada átomo.

$$a_1 + a_2 * a_3 \rightarrow A \tag{5.3}$$

$$b_1 + b_2 + b_3 \rightarrow B \tag{5.4}$$

A partir da cláusula 5.3, temos que: "$a_1$", "$a_2$" e "$a_3$" se conectam indiretamente a "A" com o auxílio de outros átomos internos, para haver um correto direcionamento e respeito à regra.

O mesmo procedimento é executado para todas as outras cláusulas, atendendo, a priori, as condições necessárias ao algoritmo CIL$^2$P.

Representamos as conexões entre os átomos, nesse caso, conforme:

*Para a cláusula 5.3.*

(0,0,0) - O zero à esquerda representa o índice de "$a_1$", o zero central é o índice do "átomo oculto" (átomo interno com índice 0), e o da direita representa o índice de "A".
(0,1,0) - O zero à esquerda representa o índice de "$a_2$", o zero central é o índice 1 do átomo interno, e o da direita representa o índice de "A".
(1,1,0) - O zero à esquerda representa o índice de "$a_3$", o zero central é o índice 1 do átomo interno, e o da direita representa o índice de "A".

Observa-se que, na disjunção, ocorre a troca de índice do átomo interno, enquanto que, na conjunção, o índice permanece o mesmo para esse átomo. O primeiro nível da estrutura atua como a conjunção (" * "), e o segundo, como a disjunção ("+").

*Para a cláusula 5.4.*

(0,0,1) - O zero à esquerda representa o índice de "$b_1$", o zero central é o índice do átomo interno, e o da direita representa o índice de "B".
(0,1,1) - O zero à esquerda representa o índice de "$b_2$", o zero central é o índice do átomo interno, e o da direita representa o índice de "B".
(0,2,1) - O zero à esquerda representa o índice de "$b_3$", o zero central é o índice do átomo interno, e o da direita representa o índice de "B".

Nessa descrição, observamos que ocorreram três disjunções; com isso, o índice do átomo interno atingiu o valor "2".

Os operadores lógicos modais afetam somente as conexões entre mundos ou conjuntos de cláusulas. No processamento inicial da programação lógica inserida, temos o armazenamento dos átomos ao qual o operador modal está relacionado. Essa informação é utilizada na criação da RNA que representa todos os mundos.

### 5.3.3 Constatações importantes

Algumas observações de ordem técnica ocorreram durante o trabalho, fazendo com que o curso do desenvolvimento e a aplicação se ramificassem ou seguissem outros caminhos. Abaixo, foram listadas as observações consideradas mais importantes.

- Uma boa escolha no valor do $A_{min}$ é muito importante, já que dele dependem diretamente o peso inicial da rede e demais valores dados pelos algoritmos estudados. Valores muito baixos para o $A_{min}$ implicam pesos com valores muito altos, o que pode produzir efeitos indesejáveis na RNA.

- O viés negado (-$\theta$) da RNA influencia diretamente o cálculo do delta-peso (regra delta, equação 2.5) durante o aprendizado da RNA. Se utilizarmos um viés positivo, estaremos conduzindo erroneamente o aprendizado, dificultando-o.

- O cálculo do Erro Médio Quadrado sofreu adaptações, conforme o seu emprego no processo de aprendizagem. Esse cálculo depende do número de épocas em operação, tornando-a variável ou constante, apenas exprimem um indicativo quanto ao processo de aprendizagem.

- Percebeu-se que o aprendizado era otimizado quando se utilizava uma taxa de aprendizagem variável conforme o número de épocas.

### 5.3.4 Fórmulas

As principais fórmulas existentes nos algoritmos implementados foram relacionadas na seqüência. No apêndice A.3, temos o detalhamento de cada uma delas através do algoritmo pelo qual foram implementadas.

#### Cálculo do $A_{min}$

O $A_{min}$ é o menor valor aceitável para a ativação do neurônio. O $A_{min}$ situa-se normalmente entre (-1 e 0) ou (0 e 1).

$$A_{min} > \frac{MAX_p(\vec{\kappa}, \vec{\mu}, n) - 1}{MAX_p(\vec{\kappa}, \vec{\mu}, n) + 1} \tag{5.5}$$

Para a fórmula acima, "n" indica a quantidade de mundos (sub-RNAs) considerados. Temos também que $\vec{\kappa}$ representa o número de literais em uma cláusula, $\vec{\mu}$ representa o número de cláusulas com o mesmo conseqüente, e $MAX_p$ é o maior dos valores entre $\kappa$ e $\mu$, isso quando consideramos todos os mundos.

O cálculo do $A_{min}$ é importante porque dele dependem os cálculos dos demais valores gerados pela aplicação do algoritmo C-IL$^2$P, nos quais a RNA irá se basear. Ainda não foi realizado nenhum estudo para determinação com precisão de um valor ótimo para $A_{min}$. Durante o trabalho, foi feita uma estimativa através do emprego do "Coeficiente de regulação do C-IL$^2$P", que fornecia um valor intermediário entre $A_{min}$ e o $A_{max}$ (o $A_{max}$ ou é -1 ou 1).

#### Cálculo do peso de uma sinapse segundo o C-IL$^2$P

As conexões entre os neurônios de uma RNA possuem um peso $W$ atribuído a elas. Esse peso pode ser calculado a partir da fórmula abaixo, relativa a um mundo em particular.

$$W \geq \frac{2}{\beta} \frac{ln(1 + A_{min}) - ln(1 - A_{min})}{MAX_p(\vec{\kappa}, \vec{\mu})(A_{min} - 1) + A_{min} + 1} \tag{5.6}$$

O peso $W$ é um dos principais valores fornecidos pelo algoritmo C-IL$^2$P, por ser determinado por uma inequação. A sua correta escolha influencia o desempenho do aprendizado da RNA.

#### Cálculo dos valores de limiares ($\theta's$)

Existem dois tipos de limiares a serem calculados segundo o algoritmo C-IL$^2$P: os limiares dos neurônios internos $\theta_l$ e os dos neurônios de saída $\theta_A$. Estes são calculados por:

$$\theta_l = \frac{(1 + A_{min})(\kappa_l - 1)}{2} W \tag{5.7}$$

$$\theta_A = \frac{(1 + A_{min})(1 - \mu_l)}{2} W \tag{5.8}$$

A razão da existência de dois tipos de limiares é o fato de o algoritmo C-IL$^2$P possibilitar a montagem de RNAs a partir de sub-RNAs com três camadas (entrada-escondida-saída). O primeiro nível de camadas (entrada-escondida) configura a disjunção lógica, subentende-se o "e". Enquanto que o próximo nível de camadas (escondida-saída) configura a conjunção lógica, subentende-se o "ou".

### Cálculo do peso Modal por RNA segundo o C-IL$^2$P Modal

Novos neurônios são gerados após a aplicação do algoritmo C-IL$^2$P Modal. Em conseqüência, novas conexões passam a existir na RNA. O peso dessas novas conexões é calculado por:

$$W^M > h^{-1}(A_{min}) + \mu W + \theta \tag{5.9}$$

O peso $W^M$ é empregado internamente na RNA. Ele conecta os neurônios que fazem o interfaceamento entre RNAs. É um parâmetro com possibilidades de variação. Ainda não houve um estudo que determinasse as suas influências no aprendizado.

# Capítulo 6

# Testes realizados

"*Experiência é o nome que cada um dá a seus erros.*"
Oscar Wilde (1854-1900) - Escritor irlandês

Os casos nos quais os algoritmos Neuro-Simbólicos foram aplicados e testados são agrupados genericamente em três, segundo o foco do estudo. São eles: (a) Uso de um problema clássico, o reconhecimento de promotores, Seção 6.2. (b) Estudo sobre a influência do domínio do conhecimento nos algoritmos e vice-versa, Seção 6.3 e (c) Aplicação específica modelada pela Lógica Modal (teste do CIL$^2$P Modal aplicado a um exemplo específico), Seção 6.4.

## 6.1 Roteiro geral dos testes

Foram realizados cinco testes com o CIL$^2$P, aplicados a um domínio incompleto do conhecimento, cujos resultados se referem ao exemplo do reconhecimento de promotores de DNA. Os testes foram ordenados conforme a seqüência abaixo:

1. Teste da implementação.
2. CIL$^2$P versus regras aplicadas diretamente.
3. RNA com retropropagação e CIL$^2$P sem retropropagação.
4. RNA com retropropagação e CIL$^2$P com retropropagação.
5. CIL$^2$P sem e com retropropagação.

Foram realizados seis testes do CIL$^2$P em domínio completo do conhecimento, cujos resultados servem para o exemplo da base de dados completa. Os teste foram executados na seguinte seqüência:

1. Aplicando a retropropagação.
2. Simulação no JavaNNS.
3. Aplicando o CIL$^2$P.

4. Aplicando a retropropagação.
5. Aplicando o CIL$^2$P.
6. CIL$^2$P com retropropagação.

O último dos testes aplica o algoritmo CIL$^2$P Modal à modelagem de uma situação específica, cujos passos testados para a correta obtenção do resultado foram: (1) A inserção do programa em Lógica Modal. (2) A conversão do programa em RNA. (3) A aprendizagem da RNA e (4) A obtenção de um sistema estável.

## 6.2 Reconhecimento de Promotores

### 6.2.1 Contextualização

Um Promotor é uma seqüência curta[1] de DNA que normalmente precede o início de um Gene. A seqüência de DNA é composta por quatro nucleotídeos (adenina-"A"; citosina-"C"; guanina-"G" e timina-"T"). Os nucleotídeos que compõem uma seqüência de DNA fornecem sentido biológico a essa seqüência.

O presente trabalho empenha-se na busca dos promotores da *Escherichia coli* (E. coli). A E. coli é uma bactéria responsável pela gastroenterite, cuja principal forma de contágio é através do consumo de carne bovina. O código genético da E. coli é conhecido e amplamente utilizado em pesquisas. A base de dados empregada foi obtida através do banco de dados da Universidade da Carolina Irvine [28].

Para o desenvolvimento do trabalho, interessa-nos a verificação de certas regras. Sabe-se que os promotores possuem uma região onde uma proteína (polimerase do RNA) deve contatar com a hélice da seqüência de DNA (que possui uma conformação válida). Quando as duas regiões de contato estão alinhadas, ocorre o reconhecimento do promotor. Na Tabela 6.1, foram transcritas as regras empíricas, definidas por profissionais, que normalmente são encontradas nos bancos de dados.

Segundo as regras, o promotor é reconhecido quando o contato e a conformação existem. O contato é reconhecido quando as regiões menos$_{10}$ e menos$_{35}$ existem. Porém, para uma região menos$_{10}$ ou menos$_{35}$ existir, basta apenas que, pelo menos, uma de suas regras seja verdadeira, por exemplo:

- **menos$_{10}$** :- p-14=t, p-13=a, p-12=t, p-11=a, p-10=a, p-9=t.

Se ocorrer o reconhecimento posicional da seqüência genética, segundo a Tabela 6.2, então, a região menos$_{10}$ é verdadeira. O reconhecimento posicional, nesse caso, acontece quando, na posição "14", existir o nucleotídeo "t", na posição "13", houver o nucleotídeo "a" e assim por diante, até a posição "9", que contém o nucleotídeo "t".

### 6.2.2 Formatação dos dados

As regras para o reconhecimento de promotores foram formatadas (Tabela 6.3) e, desta forma, inseridas ao arquivo de linguagem lógica. A formatação das regras facilita a aplicação do algoritmo de conversão por terem sua reestruturação segundo a arquitetura de uma RNA, com uma camada de entrada, três internas e uma de saída. Na Figura 6.1, as regras foram agrupadas permitindo uma visualização estrutural.

[1] Uma seqüência curta de DNA é número em torno de 57 nucleotídeos.

Tabela 6.1: Regras para o reconhecimento de promotores de DNA.

| | |
|---|---|
| **promotor** | contato, conformação. |
| **contato** | $\text{menos}_{35}$, $\text{menos}_{10}$. |
| **menos$_{35}$** | p-37=c, p-36=t, p-35=t, p-34=g, p-33=a, p-32=c. |
| **menos$_{35}$** | p-36=t, p-35=t, p-34=g, p-32=c, p-31=a. |
| **menos$_{35}$** | p-36=t, p-35=t, p-34=g, p-33=a, p-32=c, p-31=a. |
| **menos$_{35}$** | p-36=t, p-35=t, p-34=g, p-33=a, p-32=c. |
| **menos$_{10}$** | p-14 t, p-13 a, p-12=t, p-11=a, p-10=a, p-9=t. |
| **menos$_{10}$** | p-13 t, p-12=a, p-10=a, p-8=t. |
| **menos$_{10}$** | p-13 t, p-12=a, p-11=t, p-10=a, p-9=a, p-8=t. |
| **menos$_{10}$** | p-12=t, p-11=a, p-7=t. |
| **conformação** | p-47=c, p-46=a, p-45=a, p-43=t, p-42=t, p-40=a, p-39=c, p-22=g, p-18=t, p-16=c, p-8=g, p-7=c, p-6=g, p-5=c, p-4=c, p-2=c, p-1=c. |
| **conformação** | p-45=a, p-44=a, p-41=a. |
| **conformação** | p-49=a, p-44=t, p-27=t, p-22=a, p-18=t, p-16=t, p-15=g, p-1=a. |
| **conformação** | p-45=a, p-41=a, p-28=t, p-27=t, p-23=t, p-21=a, p-20=a, p-17=t, p-15=t, p-4=t. |

Tabela 6.2: Expandindo uma das regras.

| **Posição** | 14 | 13 | 12 | 11 | 10 | 9 |
|---|---|---|---|---|---|---|
| **Nucleotídeo** | t | a | t | a | a | t |

Tabela 6.3: Quadro com as regras dos promotores formatadas.

| | | | | | |
|---|---|---|---|---|---|
| $37@t*a*t*a*a*t$ | $\Rightarrow$ | $b11$ | $14@c*t*t*g*a*c$ | $\Rightarrow$ | $b21$ |
| $38@t*a*t*a*a*t$ | $\Rightarrow$ | $b12$ | $15@t*t*g*a*c*a$ | $\Rightarrow$ | $b22$ |
| $38@t*a*41@a*43@t$ | $\Rightarrow$ | $b13$ | $15@t*t*g*a*c$ | $\Rightarrow$ | $b23$ |
| $39@t*a*44@t$ | $\Rightarrow$ | $b14$ | $15@t*t*g*19@c*a$ | $\Rightarrow$ | $b24$ |
| $6@a*a*10@a$ | $\Rightarrow$ | $c11$ | $6@a*10@a$ | $\Rightarrow$ | $c21$ |
| | | | $23@t*27@t*29@a*a*33@t$ | $\Rightarrow$ | $c22$ |
| | | | $47@t$ | $\Rightarrow$ | $c23$ |
| $2@a*7@t$ | $\Rightarrow$ | $c31$ | $4@c*a*a*8@t*t*11@a*c$ | $\Rightarrow$ | $c41$ |
| $24@t*29@a*32@t*34@a$ | $\Rightarrow$ | $c32$ | $29@g*33@t*35@c$ | $\Rightarrow$ | $c42$ |
| $50@a$ | $\Rightarrow$ | $c33$ | $43@g*c*g*c*c*49@c*c$ | $\Rightarrow$ | $c43$ |
| $b11+b12+b13+b14$ | $\Rightarrow$ | $b1$ | $b21+b22+b23+b24$ | $\Rightarrow$ | $b2$ |
| $b1*b2$ | $\Rightarrow$ | $b$ | | | |
| $c11$ | $\Rightarrow$ | $c1$ | $c31*c32*c33$ | $\Rightarrow$ | $c3$ |
| $c21*c22*c23$ | $\Rightarrow$ | $c2$ | $c41*c42*c43$ | $\Rightarrow$ | $c4$ |
| $c1+c2+c3+c4$ | $\Rightarrow$ | $c$ | | | |
| $b*c$ | $\Rightarrow$ | $(a)$ | — | — | — |

O primeiro nível apresenta os nucleotídeos. Nele cada agrupamento pode ser visto como uma única entrada, representada pelos bXX e cXX. Os bXX's são unidos por disjunções em "b1" e "b2", que integram b por conjunção. Os cXX's se unem primeiramente por conjunção nos c1, c2, c3 e c4; na seqüência, resultam em "c". Observamos que há um desnível entre os caminhos das regras que resultam no reconhecimento ou não dos promotores de DNA (representado por "a").

Os 57 nucleotídeos de cada exemplo do arquivo foram separados em grupos de quatro entradas binárias, correspondendo aos nucleotídeos "A, G, T e C", nesta ordem, (1000), (0100), (0010) e (0001), totalizando 228 entradas binárias. A multiplicação do conjunto de exemplos (106 exemplos) pelo número de entradas binárias gerou uma matriz de entrada de dados para a RNA

com 24168 bits.

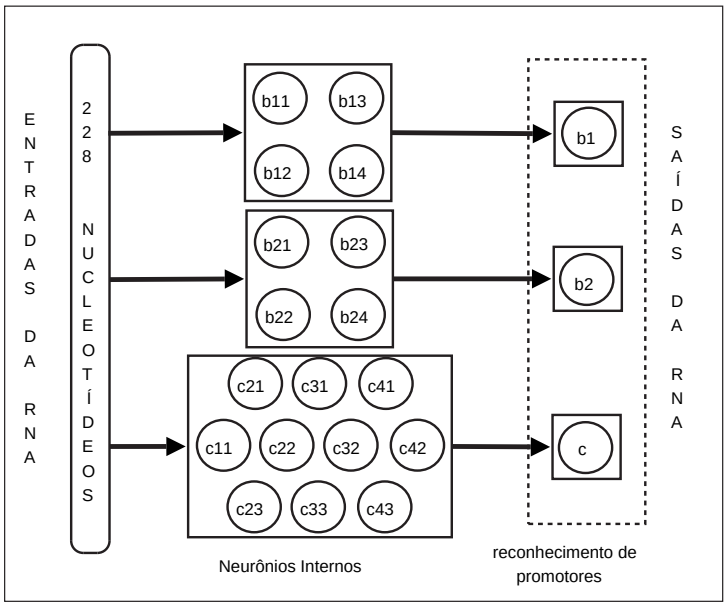


Figura 6.1: Representação da RNA gerada para o reconhecimento de promotores.

Alguns dos testes aplicados utilizaram a RNA contendo três camadas (228x18x1) (ver Figura 6.1), mas essa distribuição dos neurônios variou durante os testes. O número de conexões entre a primeira e a segunda camada totalizou 4104 sinapses . O número de neurônios internos "18" equivale ao número de regras constituídas diretamente por nucleotídeos, conforme a Tabela 6.3. As conexões entre a camada interna e a camada de saída são 18, visto que o resultado final é representado apenas por um neurônio na última camada (promotor ou não).

A maioria dos testes manteve a topologia fornecida pela regra, mas outros parâmetros relacionados à RNA variaram durante os testes. A Tabela 6.4 descreve os parâmetros mais importantes utilizados durante os testes.

Tabela 6.4: Descrição dos parâmetros utilizados.

| **Parâmetro** | **Descrição** |
|---|---|
| Taxa de aprendizagem ($\eta$) | Multiplicador da Regra Delta, utilizado no cálculo do $\Delta W$. |
| Beta ($\beta$) | Valor que determina a declividade da função de ativação. |
| Número de camadas | A quantidade total de camadas existentes na RNA. |
| Distribuição dos neurônios | Indica o número de neurônios por camada. |
| Maior quantidade de épocas | Variação máxima do número de épocas utilizada no teste. |
| Coeficiente de regulação do C-IL$^2$P | Valor que fixa os cálculos das inequações ($\geq$) de $A_{min}$ e $W$. |
| Número total de exemplos | Conjunto dos exemplos existentes no banco de dados. |
| Número de exemplos de treinamento | Quantidade de exemplos utilizados para o treinamento da RNA. |
| Número de exemplos de validação | Quantidade de exemplos utilizados para a validação da RNA. |
| Número de exemplos de teste | Quantidade de exemplos utilizados para o teste específico da RNA. |

### 6.2.3 Testes realizados para o reconhecimento de promotores de DNA

Os cinco testes executados no trabalho inicial e detalhados no texto desta Subseção são: (1) Teste da implementação. (2) CIL$^2$P versus regras aplicadas diretamente. (3) RNA com retropropagação e CIL$^2$P sem retropropagação. (4) RNA com retropropagação e CIL$^2$P com retropropagação e (5) CIL$^2$P sem e com retropropagação. Conforme essa seqüência de testes, encontraremos o objetivo, a descrição, os parâmetros utilizados, os resultados obtidos, assim como a avaliação, referente a cada teste.

**Primeiro teste: Teste da implementação**

**Objetivo:** Verificar se a implementação de RNAs está funcionando corretamente.

**Descrição:** Para esse teste, os exemplos dos arquivos de promotores foram divididos em três arquivos:

- Arquivos de treinamento, contendo 60 exemplos.
- Arquivos de validação, contendo 26 exemplos.
- Arquivos de teste, contendo os 20 exemplos restantes.

Os arquivos de treinamento e validação foram empregados no aprendizado das RNAs. Foram Utilizados os resultados do simulador JavaNNS (*Java Neural Networks Simulator*) [16, 39] como referência para fins de comparação com os resultados obtidos pela implementação. Um exemplo de estrutura de RNA gerada pelo simulador JavaNNS aparece na Figura 6.2. O algoritmo utilizado na aprendizagem foi o de retropropagação para os dois casos. O programa foi executado 50 vezes. A média dos resultados foi calculada e comparada com os resultados do simulador.

**Parâmetros:** Os parâmetros empregados para esse teste são apresentados na Tabela 6.5. Os demais parâmetros requeridos pelo JavaNNS foram deixados no seu modo *default*.

Tabela 6.5: Parâmetros utilizados no primeiro teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0.05 | $\beta$ | 1 |
| Nº de camadas | 4 | Distribuição dos neurônios | (228 x 12 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | | |
| Nº total de exemplos | 106 | Nº de exemplos de treinamento | 60 |
| Nº de exemplos de validação | 26 | Nº de exemplos de teste | 20 |

**Resultados:** O gráfico da Figura 6.3 representa a diminuição do erro, consistindo em uma amostra do que ocorria para os dois casos testados, em que a simulação no JavaNNS refletiu os resultados do programa implementado. A Tabela 6.6 contém os resultados utilizados para validar a implementação.

**Avaliação:** Observou-se que a implementação feita para a RNA e a simulação feita no JavaNNS convergiam para a mesma resposta. Embora, no simulador, possamos

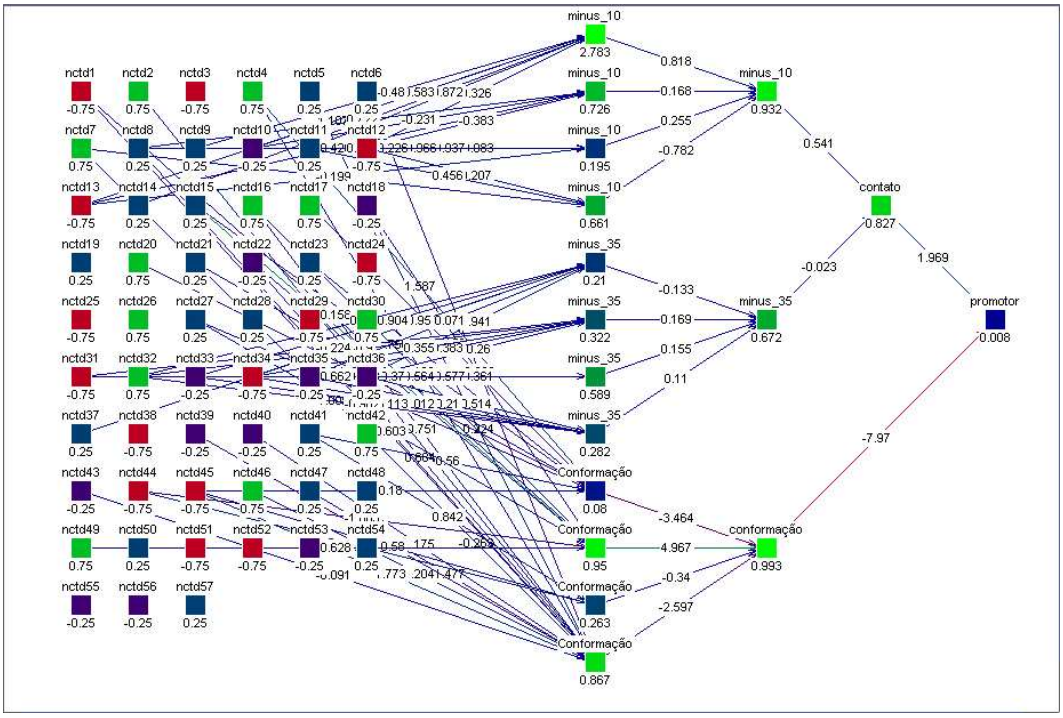


Figura 6.2: RNA dos promotores de DNA simulada no JavaNNS [16].

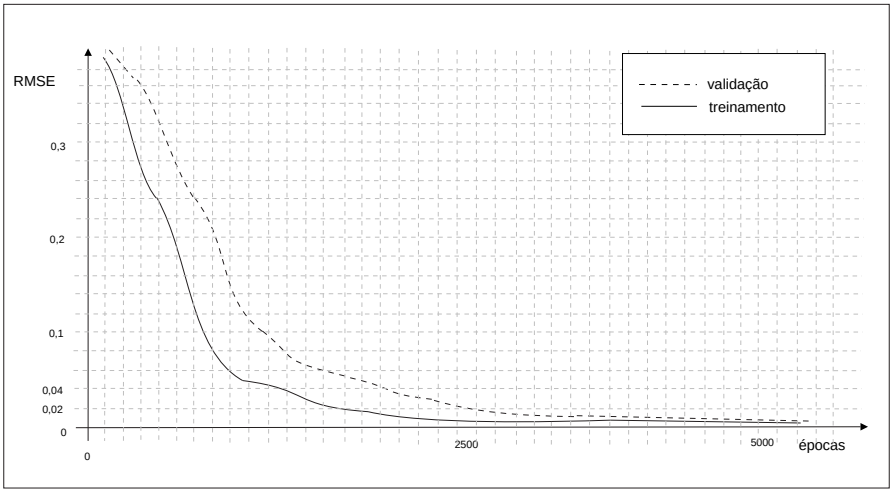


Figura 6.3: Gráfico de diminuição do erro da RNA por épocas.

contar com refinamentos que não foram implementados no programa utilizado, tais como "momentum" da RNA[2], a comparação entre os resultados dos dois métodos foi

[2]Similar ao momentum, ou inércia da física, é uma resistência à mudança do sentido do movimento; mais especificamente, trata-se de um percentual que é multiplicado à última variação do peso e somado à variação atual.

Tabela 6.6: Resultados comparativos do primeiro teste.

| Técnica | Quantidade de testes | Média de acertos | Desvio padrão |
|---|---|---|---|
| retropropagação | 50 | 83,33% | 8,26% |
| JavaNNS | 10 | 85,38% | 1,13% |

satisfatória, caracterizando o aprendizado da RNA.

### Segundo teste: Teste do C-IL$^2$P e aplicação direta das regras

**Objetivo:** Verificar se o C-IL$^2$P consegue representar um conjunto de regras corretamente.

**Descrição:** Primeiramente, implementou-se um programa que percorria o conjunto dos 106 exemplos de promotores. As seqüências de DNA existentes nos exemplos de promotores correspondentes às existentes nas regras eram separadas conforme a posição esperada. O passo seguinte foi conferir quais das seqüências dos exemplos, quando aplicadas às regras, ativavam o reconhecimento do promotor. Iniciando a outra etapa do teste, as regras foram lidas pelo programa que implementa o C-IL$^2$P, convertendo-as em uma RNA. A verificação dos resultados foi feita através da simples leitura de cada exemplo na camada de entrada da RNA e a verificação da respectiva resposta na saída da mesma estrutura (promotor ou não).

**Parâmetros:** Os parâmetros empregados para esse teste estão de acordo com a Tabela 6.7. A exceção, neste caso, é a de que o conjunto de teste foi composto pelo conjunto total de exemplos, os 106 exemplos, não havendo treinamento, nem validação.

Tabela 6.7: Parâmetros utilizados no segundo teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0.05 | $\beta$ | 1 |
| Nº de camadas | 4 | Distribuição dos neurônios | (228 x 12 x 3 x 1) |
| Coeficiente de regulação do C-IL$^2$P | 2 | | |
| Nº total de exemplos | 106 | Nº de exemplos de treinamento | 60 |
| Nº de exemplos de validação | 26 | Nº de exemplos de teste | 20 |

**Resultados:** A Tabela 6.8 resume os resultados pela quantidade de acertos obtidos.

Tabela 6.8: Resultados obtidos no segundo teste.

| Técnica | Número de execuções | Acertos |
|---|---|---|
| Análise das regras | 1 | 50% |
| C-IL$^2$P sem aprendizagem | 1 | 72% |

**Avaliação:** Quando aplicamos "rigidamente" as regras ao conjunto de exemplos, verificamos que elas não são capazes de reconhecer nenhum promotor. As regras

fornecem sempre valor falso para os promotores do banco de dados. Os 50% de acertos apresentados devem-se às seqüências genéticas diferentes de promotores. No entanto, quando o C-IL$^2$P é aplicado ao mesmo conjunto de exemplos, obtêm-se 72% de acertos no reconhecimento dos promotores de DNA. Isso se deve ao fato de que as regras são inflexíveis aos dados, enquanto que as RNAs têm a propriedade de generalizar. Porém, necessitamos de um conjunto de regras *completo* para podermos avaliar se o C-IL$^2$P é capaz de representar corretamente um conjunto de regras. Isso será mostrado na Seção 6.3.

**Terceiro teste: RNA com retropropagação e C-IL$^2$P sem aprendizagem**

**Objetivo:** Verificar qual dos dois métodos é o melhor em termos de acertos: a RNA, cuja aprendizagem é determinada pelo algoritmo de retropropagação ou a RNA, gerada apenas pelo algoritmo de conversão do sistema C-IL$^2$P.

**Descrição:** Iniciou-se pela definição da topologia da RNA apenas fixando a quantidade de neurônios por camada (228 x 12 x 3 x 1). Os valores dos pesos das sinapses foram atribuídos inicialmente de forma aleatória, para, logo em seguida, serem alterados pelo processo de aprendizagem. O outro método não utilizou pesos aleatórios, mas eles foram definidos fornecidos pelo C-IL$^2$P, assim como os limiares da rede. Para a RNA com aprendizagem, procedeu-se da seguinte forma: após a estabilização do erro (RMSE), para o treinamento e a validação do processo de aprendizado da RNA[3], foi aplicado o conjunto de teste definido nos parâmetros.

**Parâmetros:** Os parâmetros empregados para esse teste estão de acordo com a Tabela 6.9.

Tabela 6.9: Parâmetros utilizados no terceiro teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0.05 | $\beta$ | 1 |
| Nº de camadas | 4 | Distribuição dos neurônios | (228 x 12 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | Coeficiente de regulação do C-IL$^2$P | 2 |
| Nº total de exemplos | 106 | Nº de exemplos de treinamento | 60 |
| Nº de exemplos de validação | 26 | Nº de exemplos de teste | 20 |

**Resultados:** A Tabela 6.10 resume os resultados obtidos no terceiro teste.

Tabela 6.10: Resultados obtidos no terceiro teste.

| **Técnica** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|
| retropropagação | 50 | 83,33% | 8,26% |
| C-IL$^2$P sem aprendizagem | 1 | 72% | - |

**Avaliação:** A RNA treinada por retropropagação e validada obteve, em média, um percentual de acertos de 83,33%. Enquanto que a RNA, gerada a partir do

[3]Normalmente abaixo de 0,02.

algoritmo C-IL$^2$P obteve 72% de acertos. Percebeu-se que o aprendizado age de maneira efetiva. Obtemos um bom resultado mesmo para uma RNA que não tenha nenhuma informação inicial sobre o domínio do conhecimento ao qual ela é aplicada.

**Quarto teste: C-IL$^2$P com aprendizado e C-IL$^2$P sem aprendizado**

**Objetivo:** Verificar o ganho, representado pelo aumento do número de acertos de teste, que se obtém pela aplicação do algoritmo de aprendizagem (retropropagação) na estrutura da RNA gerada pelo algoritmo de conversão do sistema C-IL$^2$P.

**Descrição:** A partir das regras, gerou-se uma RNA inicial (como feito no segundo teste), através da qual se pôde observar melhor o percentual de acertos resultante do aprendizado, que acrescenta as fases de treinamento e validação antes da fase de teste da RNA. O processo de monitoração utilizado foi o mesmo apresentado na Seção 2.2, visando a um erro mínimo.

**Parâmetros:** Os parâmetros empregados para esse teste estão de acordo com a Tabela 6.11.

Tabela 6.11: Parâmetros utilizados no quarto teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0.05 | $\beta$ | 1 |
| Nº de camadas | 4 | Distribuição dos neurônios | (228 x 12 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | Coeficiente de regulação do C-IL$^2$P | 2 |
| Nº total de exemplos | 106 | Nº de exemplos de treinamento | 60 |
| Nº de exemplos de validação | 26 | Nº de exemplos de teste | 20 |

**Resultados:** A Tabela 6.12 apresenta a comparação de resultados para este teste.

Tabela 6.12: Resultados obtidos no quarto teste.

| Técnica | Quantidade de testes | Média de acertos | Desvio padrão |
|---|---|---|---|
| C-IL$^2$P sem aprendizagem | 1 | 72% | - |
| C-IL$^2$P com aprendizagem | 50 | 88,22% | 6,14% |

**Avaliação:** Obteve-se o maior percentual de acertos entre todos os testes pela aplicação do aprendizado na RNA gerada a partir do C-IL$^2$P: em média, 88,22%. Em casos particulares, obteve-se RNAs com percentual de acerto superior a 94%. Os melhores resultados são obtidos quando a aprendizagem é orientada pelas regras iniciais. Isso comprova o quanto o conhecimento inicial (regras) é importante para se obter um bom desempenho na aprendizagem.

**Quinto teste: RNA com retropropagação e C-IL$^2$P com aprendizagem**

**Objetivo:** Verificar se a RNA, utilizando o C-IL$^2$P com posterior treinamento, consegue resultados iguais ou melhores que uma RNA treinada por retropropagação.

**Descrição:** Comparar os resultados, em termos de acertos, entre RNAs que iniciam com os pesos das conexões aleatoriamente e as que iniciam com os pesos das conexões dados pelo algoritmo C-IL$^2$P.

**Parâmetros:** Os parâmetros empregados para esse teste estão de acordo com a Tabela 6.13.

Tabela 6.13: Parâmetros utilizados no quinto teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0.05 | $\beta$ | 1 |
| Nº de camadas | 4 | Distribuição dos neurônios | (228 x 12 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | Coeficiente de regulação do C-IL$^2$P | 2 |
| Nº total de exemplos | 106 | Nº de exemplos de treinamento | 60 |
| Nº de exemplos de validação | 26 | Nº de exemplos de teste | 20 |

**Resultados:** A Tabela 6.14 apresenta a comparação das duas técnicas: a com pesos aleatórios e com o C-IL$^2$P.

Tabela 6.14: Resultados obtidos no quinto teste.

| **Técnica** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|
| retropropagação | 50 | 83,33% | 8,26% |
| C-IL$^2$P com aprendizagem | 50 | 88,22% | 6,14% |

**Avaliação:** Conforme verificado na Tabela 6.14, a RNA que aprende por retropropagação obtém 83,33% de média de acertos, contra os 88,22% da RNA com C-IL$^2$P e com aprendizagem. Houve uma diferença significativa, o que vem a confirmar que as condições iniciais estabelecidas pelo C-IL$^2$P propiciam um melhor aprendizado. Percebe-se que há um direcionamento pela utilização do C-IL$^2$P voltado à melhor aprendizagem da RNA.

### 6.2.4 Análise geral de resultados

O primeiro teste apenas teve o efeito de confirmação da implementação feita: se os algoritmos e suas aplicações estavam corretos. A partir dos resultados verificados no segundo teste, quando aplicamos as regras diretamente aos exemplos da base de dados sem sucesso no reconhecimento de promotores, podemos fazer as seguintes análises. Os trechos de código genético existem na seqüência dos exemplos e estão deslocados. Ao contrário do que se imaginava anteriormente a esse teste, as regras não são precisas a ponto de apontarem se o exemplo é ou não um promotor de DNA. As regras da Tabela 6.3, devido à sua origem empírica, apenas dão a indicação inicial, constituindo assim um domínio de conhecimento incompleto. A necessidade de mais informações para completar o conhecimento faz com que esse seja um banco de dados amplamente utilizado pelos pesquisadores da área de RNAs [28], no qual o uso de RNA tem se destacado. A Tabela 6.15 apresenta um resumo dos resultados baseado na média de acertos das RNAs finais, em que o destaque é o C-IL$^2$P com aprendizagem, que apresentou o melhor resultado.

Tabela 6.15: Resultados obtidos nos testes.

| Técnica | Quantidade de testes | Média de acertos | Desvio padrão |
|---|---|---|---|
| retropropagação | 50 | 83,33% | 8,26% |
| C-IL$^2$P sem aprendizagem | 1 | 72% | - |
| C-IL$^2$P com aprendizagem | 50 | 88,22% | 6,14% |

Para algumas RNAs, foram obtidos ótimos resultados, estando no patamar esperado, quando comparado com outros algoritmos de treinamento. Os resultados obtidos pelas RNAs simuladas no programa JavaNNS coincidiram com os apresentados pela Tabela 6.15.

Observou-se que os resultados apresentados refletem as características do conjunto de dados escolhido e a técnica empregada para seu tratamento. Em conseqüência disso, existem e existirão diferentes resultados. Como exemplo, os resultados apresentados pelo artigo que define o C-IL$^2$P [12] possuem valores diferentes dos encontrados neste trabalho, devido à diferença de métodos de teste empregados; porém, os valores são aproximados, e as conclusões são as mesmas.

O C-IL$^2$P sem aprendizagem apresentou 72% de acertos ao invés de 100%, justamente em razão de as regras não serem completas, mas representarem um conhecimento empírico e incompleto. Desse fato, sabemos que os valores calculados para as conexões e para os limiares não são precisos o suficiente para obtermos os 100% de acerto, somente atingidos com o treinamento (que, no caso, apresentou uma melhora para 88,22%).

## 6.3 Utilização de uma base de dados completa

Esta seção mostra os testes em um domínio do conhecimento completo, em que todas as informações que estão contidas em uma base de dados são conhecidas. Essa base de dados completa é originada a partir de uma regra lógica. O uso dessa base de dados nos permitiu testar o algoritmo C-IL$^2$P e perceber o seu comportamento quando aplicado a situações nas quais temos certeza da resposta. Os testes apresentados tiveram como objetivo a comparação dos resultados após a aplicação do algoritmo C-IL$^2$P, verificando a sua validade para os casos que são apresentados na seqüência.

### 6.3.1 Contextualização

O emprego do algoritmo C-IL$^2$P no reconhecimento de promotores mostra a ocorrência do descobrimento de novos conhecimentos a partir de conhecimentos existentes. Nesse caso, consideramos que houve o uso de um domínio de conhecimento incompleto visando a ampliá-lo. No exemplo dos promotores, parte das informações estão contidas nas regras e parte na base de dados trabalhada, as quais devem ser pesquisadas pelos algoritmos na formação dos resultados. Podemos criar artificialmente uma base de dados cujo domínio de conhecimento identifiquemos. Nessa base de dados artificial, todas as informações podem ser convertidas em regras e vice-versa.

Uma base de dados artificial pode ser gerada a partir de uma regra lógica *matriz*, a qual produz um valor lógico a partir de variáveis ou átomos. A união de todos os valores lógicos produzidos e as respectivas variáveis que os produziram formam a base de dados. Uma regra contendo 11 átomos (de "a" até "k") foi escolhida para este exemplo, conforme abaixo:

$$((a*b)+(c*d)')'+((e*f)'+(g*h))'+ (i*j*k)$$

Fez-se a separação da cláusula em níveis para a obtenção do resultado final de cada exemplo interpretado por ela. Percebe-se que, na cláusula apresentada, temos o nível de entrada - composto pelos átomos, um primeiro nível de literais - composto pela união dos átomos, um segundo nível de literais - formado pelo conjunto dos literais do primeiro nível, e o nível final, que representa toda a cláusula. A negação na cláusula foi representada por " ´ ".

### 6.3.2 Descrição geral dos testes

A programação desenvolvida produziu todas as possibilidades de variação lógica da regra apresentada. A produção total dessa regra equivale a 2048 ($2^{11}$) resultados, segundo as entradas binárias aplicadas aos átomos. Os resultados estão divididos em 865 verdadeiros (exemplos positivos) e 1183 falsos (exemplos negativos). A base de dados aplicada na RNA é dividida nos arquivos de treinamento (1536 exemplos), teste (256 exemplos) e validação (256 exemplos).

A Figura 6.4 representa a topologia de RNA, cuja estrutura é construída a partir da regra lógica, conforme aplicação do C-IL$^2$P. Nessa figura, são destacadas quatro sub-RNAs, as quais podem ser agrupadas ou mantidas independentes para o estudo de caso feito nesta seção.

A aplicação básica do C-IL$^2$P se dá sobre três camadas ou dois níveis ("ou" e "e"). Devido a isso, a RNA gerada possibilitou a aplicação dos cálculos do C-IL$^2$P de duas formas diferentes, gerando dois casos para a análise. Conforme o agrupamento de sub-RNAs, temos que: (1) O primeiro caso utiliza as quatro sub-RNAs de forma independente para base dos cálculos. (2) O segundo executa os cálculos por camada, com um grupo contendo as três sub-RNAs de entrada e o outro com a sub-RNA de saída (sub-RNA 4, na figura).

Os três testes descritos na Subseção 6.3.3 referem-se ao primeiro caso, sendo eles: (1) A aplicação da retropropagação na RNA gerada. (2) A simulação da RNA no JavaNNS e (3) A aplicação do CIL$^2$P na RNA. Esses testes são os primeiros de um total de seis, utilizando o exemplo da base de dados completa. No segundo caso (Subseção 6.3.4), houve dois testes: um, na aplicação da retropropagação, e o outro do CIL$^2$P, na RNA gerada. Por fim, a Subseção 6.3.5 apresenta o sexto teste do conjunto, que uniu as técnicas do CIL$^2$P com a retropropagação em uma topologia de RNA diferente das anteriores. A metodologia de análise segue os moldes da anterior, destacando-se o objetivo, a descrição, os parâmetros, os resultados e a avaliação para cada teste.

### 6.3.3 Primeiro caso: quatro sub-RNAs

O primeiro caso é o que possui quatro sub-redes neurais (Figura 6.4), onde as três sub-redes de entrada geram cada qual a sua saída, cada saída serve como entrada para a quarta sub-rede. Cada RNA possui alguns parâmetros próprios aplicáveis à implementação, como demonstrado nas tabelas 6.16 e 6.23.

Pela perspectiva da IA Simbólica, o maior valor entre o número de literais de uma cláusula ($\kappa$) e o número de cláusulas com o mesmo conseqüente ($\mu$) é representado por MAX$_p$ [12]. Ver a Subseção 5.3.4, ou:

$$MAX_p = MAX(\kappa, \mu) \tag{6.1}$$

Segundo uma visão conexionista, temos: ($\kappa$) é a quantidade de neurônios de entrada ligados ao neurônio interno em cada sub-rede e ($\mu$) representa o número de neurônios internos ligados ao neurônio de saída; MAX$_p$ continua sendo o maior valor numérico entre ($\kappa$) e ($\mu$).

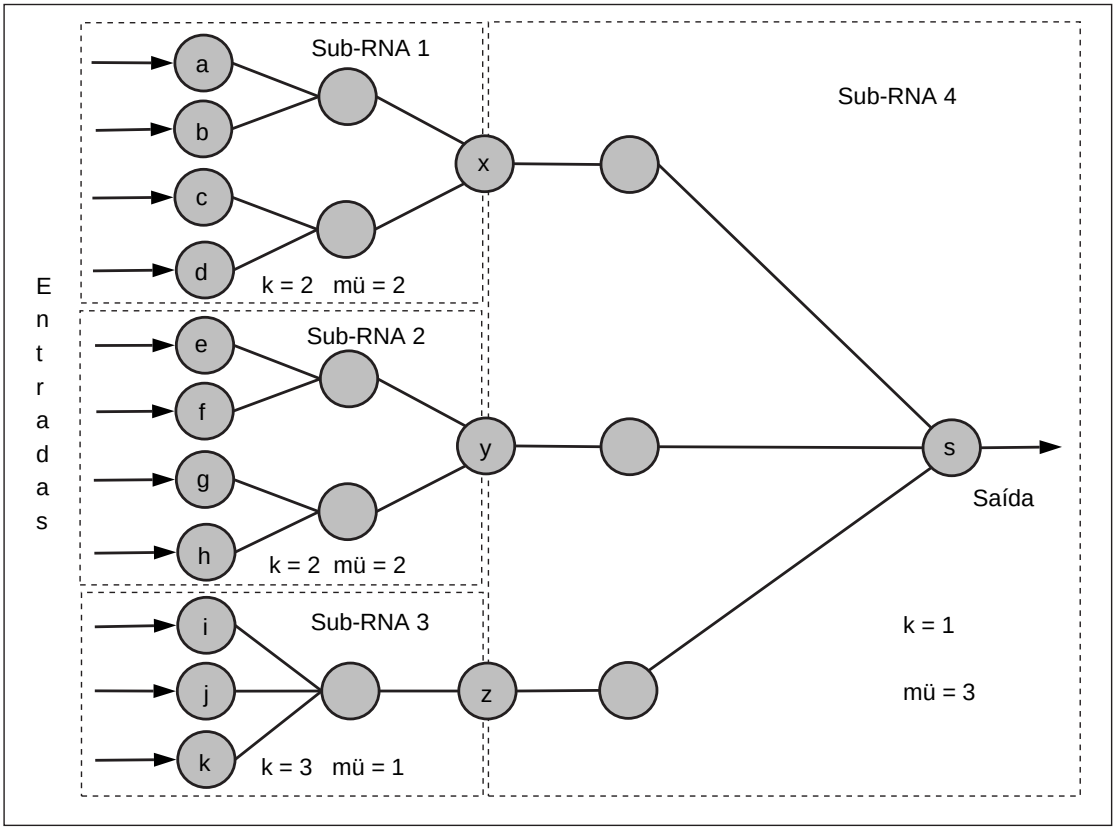


Figura 6.4: Representação topológica da RNA (primeiro caso).

Tabela 6.16: Parâmetros das Sub-RNAs (primeiro caso).

| **Sub-RNA** | $\kappa$ | $\mu$ | $\mathbf{MAX}_p$ |
|---|---|---|---|
| **1** | 2 | 2 | 2 |
| **2** | 2 | 2 | 2 |
| **3** | 3 | 1 | 3 |
| **4** | 1 | 3 | 3 |

O conjunto de testes apresentados a seguir contou com um recurso extra na aprendizagem das RNAs: a taxa de aprendizagem foi variável conforme o número de épocas. Graficamente, tal recurso torna a curva (Erro x Época) mais uniforme. Facilita-se a busca por pontos de mínimos no espaço de aprendizagem. A taxa de aprendizagem é expressa, então, pela equação:

$$T_a = T_i * (1 - \frac{Epo_{Atual}}{NEpo}) \tag{6.2}$$

Na equação 6.2, $T_a$ representa a taxa de aprendizagem, $T_i$, a taxa inicial de aprendizagem (valor que decresce, trabalhou-se com "0,05"), $Epo_{Atual}$, o número da época atual na iteração e $NEpo$, o número total de épocas. Nos testes a seguir, a abreviação "var." indica que se utilizou taxa de aprendizagem variável.

### Primeiro teste: retropropagação

**Objetivo:** Verificar a aprendizagem da RNA por retropropagação para o primeiro caso.
**Descrição:** Primeiramente, gerar a RNA com pesos iniciais de valores aleatórios a partir da estrutura da regra lógica. Logo em seguida, treinar a RNA com o algoritmo de retropropagação. Os arquivos de dados utilizados foram os de treinamento (1536 exemplos), validação (256 exemplos) e teste (256 exemplos).
**Parâmetros:** A Tabela 6.17 contém os parâmetros base para o primeiro teste.

Tabela 6.17: Parâmetros utilizados no primeiro teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | var. | $\beta$ | 1 |
| Nº de camadas | 5 | Distribuição dos neurônios | (11 x 5 x 3 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | | |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

**Resultados:** A Tabela 6.18 apresenta o resultado obtido para este teste.

Tabela 6.18: Resultado obtido no primeiro teste.

| Aprendizagem | Quantidade de testes | Média de acertos | Desvio padrão |
|---|---|---|---|
| retropropagação | 50 | 84,34% | 4,62% |

**Avaliação:** A média de acertos não foi superior devido à inexistência de otimizações na implementação e em razão de o número de épocas de aprendizado empregado nesse teste ser relativamente baixo.

### Segundo teste: simulação no JavaNNS

**Objetivo:** Verificar os resultados obtidos na simulação da RNA provenientes da regra lógica trabalhada.
**Descrição:** Construiu-se a RNA no simulador com pesos aleatórios, aplicou-se, na seqüência, o algoritmo de aprendizagem por retropropagação. A divisão dos exemplos permaneceu a mesma adotada para o teste anterior: 75% dos exemplos para treinamento, 15% para validação e 15% aplicados no teste.
**Parâmetros:** Os parâmetros utilizados nesse teste estão relacionados na Tabela 6.19.

**Resultados:** Esse teste apresentou como resultados o valor contido na Tabela 6.20.

**Avaliação:** O resultado apresentado pelo simulador esteve dentro do esperado: aproximadamente 100% de acertos, representando a facilidade do aprendizado no domínio completo do conhecimento específico trabalhado.

Tabela 6.19: Parâmetros utilizados no segundo teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | 0,05 | $\beta$ | 1 |
| Nº de camadas | 5 | Distribuição dos neurônios | (11 x 5 x 3 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | | |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

Tabela 6.20: Resultado obtido no segundo teste.

| **Aprendizagem** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|
| JavaNNS | 10 | 97,06% | 1,25% |

**Terceiro teste: aplicando o C-IL$^2$P no primeiro caso**

**Objetivo:** Verificar os resultados em um domínio completo do conhecimento, sabendo-se que o algoritmo de conversão do sistema C-IL$^2$P melhora o aprendizado, segundo conclusões dos teste para os promotores.
**Descrição:** A divisão dos arquivos utilizados na aprendizagem permaneceu em 75% para treinamento, 15% para validação e 15% para teste. Primeiramente, instanciou-se a RNA baseada na regra lógica que gerou a base de dados. Os pesos e limiares foram calculados segundo o algoritmo de conversão do C-IL$^2$P. A aprendizagem foi determinada pelo algoritmo de retropropagação.
**Parâmetros:** Os parâmetros utilizados para o terceiro teste seguem o apresentado pela Tabela 6.21.

Tabela 6.21: Parâmetros utilizados no terceiro teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | var. | $\beta$ | 1 |
| Nº de camadas | 5 | Distribuição dos neurônios | (11 x 5 x 3 x 3 x 1) |
| Coeficiente de regulação do C-IL$^2$P | 2 | Quantidade máxima de épocas | 100 |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

**Resultados:** O terceiro teste obteve, como resultado, o conteúdo da Tabela 6.22. A peculiaridade na obtenção dos resultados foi atingi-lo com um número baixo de épocas, aproximadamente em 50 épocas.

Tabela 6.22: Resultados obtidos no terceiro teste.

| **Aprendizagem** | **Quantidade de testes** | **Média de acertos** |
|---|---|---|
| CIL$^2$P | 10 | 100% |

**Avaliação:** Esse teste confirmou que o algoritmo C-IL$^2$P apresenta o melhor percentual de acerto entre todos os testes, repetindo o que foi apresentado nos testes dos promotores.

### 6.3.4 Segundo caso: RNA dividida em dois grupos (por camadas)

O segundo caso, proveniente da regra lógica, possui dois grupos de sub-RNAs. O primeiro grupo de sub-RNA é a união das sub-RNAs 1, 2 e 3 da Figura 6.4, formando o grupo de sub-RNAs de entrada. O segundo grupo de sub-RNA desse caso é a sub-RNA 4 do primeiro caso. Os dois testes aplicados a esse caso utilizaram a mesma topologia de RNA do primeiro caso. A Tabela 6.23 mostra o novo $MAX_p$ para esse segundo caso.

Tabela 6.23: Parâmetros das Sub-RNAs (segundo caso).

| **Sub-RNA** | $\kappa$ | $\mu$ | $\mathbf{MAX}_p$ |
|---|---|---|---|
| **1** | 2 | 2 | |
| **2** | 2 | 2 | 3 |
| **3** | 3 | 1 | |
| **4** | 1 | 3 | 3 |

**Quarto teste: retropropagação no segundo caso**

**Objetivo:** Verificar se os cálculos do $\mu$ e do $\kappa$ influenciam significativamente na média dos resultados finais das RNAs.
**Descrição:** A divisão dos arquivos de aprendizagem foi mantida em 75%, 15% e 15%. De maneira similar à apresentada no primeiro teste, instanciou-se a RNA a partir da regra utilizando os pesos aleatórios nas suas conexões. Posteriormente, houve o treinamento da RNA por retropropagação.
**Parâmetros:** Os parâmetros para o quarto teste estão relacionados na Tabela 6.24.

Tabela 6.24: Parâmetros utilizados no quarto teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | var. | $\beta$ | 1 |
| Nº de camadas | 5 | Distribuição dos neurônios | (11 x 5 x 3 x 3 x 1) |
| Coeficiente de regulação do C-IL$^2$P | 2 | Quantidade máxima de épocas | 10000 |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

**Resultados:** Os resultados para o quarto teste são apresentados na Tabela 6.25.

Tabela 6.25: Resultados obtidos no quarto teste.

| **Técnica** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|
| retropropagação | 50 | 84% | 5,15% |

**Avaliação:** Os 84% de média de acertos obtidos são compatíveis ao percentual de 84,34% obtidos no primeiro teste. Isso significa que não houve diferença considerável devido a forma como os neurônios foram agrupados, na primeira topologia.

**Quinto teste: aplicando o C-IL$^2$P no segundo caso**

**Objetivo:** Verificar se os cálculos do $\mu$ e do $\kappa$ influenciam significativamente na média dos resultados finais das RNAs.
**Descrição:** Instanciou-se a RNA pelo algoritmo C-IL$^2$P, aplicou-se, na seqüência, o algoritmo de aprendizagem por retropropagação. A divisão dos exemplos permaneceu a mesma adotada para o teste anterior: 75% dos exemplos para treinamento, 15% para validação e 15% aplicados no teste.
**Parâmetros:** Os parâmetros utilizados no quinto teste estão em conformidade com a Tabela 6.26.

Tabela 6.26: Parâmetros utilizados no quinto teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | var. | $\beta$ | 1 |
| Nº de camadas | 5 | Distribuição dos neurônios | (11 x 5 x 3 x 3 x 1) |
| Quantidade máxima de épocas | 10000 | Coeficiente de regulação do C-IL$^2$P | 2 |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

**Resultados:** A Tabela 6.27 contém os resultados obtidos pela aplicação do C-IL$^2$P, a segunda topologia utilizada.

Tabela 6.27: Resultados obtidos no quinto teste.

| **Técnica** | **Quantidade de testes** | **Média de acertos** |
|---|---|---|
| CIL$^2$P | 10 | 100% |

**Avaliação:** Assim como no terceiro teste, os resultados refletem que o C-IL$^2$P maximiza a aprendizagem, comprovada para um domínio completo. Os resultados obtidos também estão em conformidade com o quarto teste no que diz respeito à irrelevância nos cálculos de $\mu$ e $\kappa$ para a regra lógica utilizada.

### 6.3.5 RNA sem C-IL$^2$P (três camadas)

Finalizando, podemos fazer uma análise que nos permite comparar as influências da topologia da RNA. Para essa análise, utilizamos uma topologia de RNA não baseada na regra, mas que emprega a mesma base de dados. Essa topologia é constituída por três camadas, nas quais os neurônios da camada interna da primeira topologia são substituídos por uma única camada, contendo um número diferente de neurônios.

**Sexto teste: retropropagação em três camadas (sem C-IL$^2$P)**

**Objetivo:** Verificar qual a resposta de uma RNA com topologia diferente da fornecida pela regra lógica.

**Descrição:** A RNA foi instanciada com pesos iniciais aleatórios. Logo em seguida, foi treinada com o algoritmo de retropropagação. Os arquivos de dados utilizados foram os de treinamento (1536 exemplos), validação (256 exemplos) e teste (256 exemplos).
**Parâmetros:** Os parâmetros utilizados nesse teste estão expostos pela Tabela 6.28.

Tabela 6.28: Parâmetros utilizados no sexto teste.

| | | | |
|---|---|---|---|
| Taxa de aprendizagem | var. | $\beta$ | 1 |
| Nº de camadas | 3 | Distribuição dos neurônios | (11 x 10 x 1) |
| Quantidade máxima de épocas | 10000 | | |
| Nº total de exemplos | 2048 | Nº de exemplos de treinamento | 1536 |
| Nº de exemplos de validação | 256 | Nº de exemplos de teste | 256 |

**Resultados:** Os resultados obtidos com esse teste estão presentes na Tabela 6.29.

Tabela 6.29: Resultados obtidos no sexto teste.

| **Aprendizagem** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|
| retropropagação | 50 | 72,66% | 8,27% |

**Avaliação:** O resultado obtido, por ser baixo se comparado com os demais testes, pode significar que a estrutura topológica desta RNA está distante do resultado obtido pelo C-IL$^2$P.

### 6.3.6 Análise de resultados

Antes de iniciar a análise sobre os resultados, cabe salientar que: Apesar de o número de execuções das implementações não coincidir entre os testes (10 e 50 execuções, Tabela 6.30), o que pode indicar uma invalidade estatística dos resultados, verificamos, para esse caso, que a técnica CIL$^2$P precisa ser aplicada somente uma vez a fim de atingir os 100% de acertos. As 10 execuções apenas garantiram o resultado que já era esperado. Para a simulação no JavaNNS, não foram obtidos os 100%; porém, interessa-nos saber que o simulador também consegue um percentual de acerto alto para o exemplo da base de dados completa.

Tabela 6.30: Resultados obtidos nos testes do base de dados artificial.

| **Técnica** | **Teste** | **Quantidade de testes** | **Média de acertos** | **Desvio padrão** |
|---|---|---|---|---|
| CIL$^2$P | Terceiro | 10 | 100% | - |
| | Quinto | 10 | 100% | - |
| JavaNNS | Segundo | 10 | 97,06% | 1,25% |
| retropropagação | Primeiro | 50 | 84,34% | 4,62% |
| | Quarto | 50 | 84% | 5,15% |
| | Sexto | 50 | 72,66% | 8,27% |

Algumas constatações importantes foram obtidas com a utilização das implementações aplicadas ao base de dados artificial, criado a partir de uma regra lógica. Podemos destacar as constatações conforme segue:

1. Teste do algoritmo de retropropagação: - Para cada um dos casos estudados, observou-se que as curvas do erro possuíam tendência para zero, mas estabilizavam-se antes do erro ficar nulo.

2. Resultados similares para o primeiro e o segundo caso: - A diferença inicial dada pelos cálculos do algoritmo C-IL$^2$P para cada caso não representou significativa mudança nos resultados.

3. Foi possível constatar a eficácia da conversão do algoritmo C-IL$^2$P: - Obtiveram-se exatos 100% de acertos sobre qualquer um dos valores lidos pela RNA provenientes da base de dados, após a aplicação do algoritmo C-IL$^2$P na construção da topologia da RNA.

4. Pode-se reproduzir o banco de dados: - Observou-se que podemos reproduzir todo o banco de dados a partir de uma fração pequena de exemplos (15%). Basta fazer o caminho inverso após a aplicação do algoritmo C-IL$^2$P, extraindo uma nova regra a partir da RNA, após o aprendizado pelos exemplos.

## 6.4 Aplicando a Lógica Modal

### 6.4.1 Contextualização

A implementação do ambiente computacional (Seção 5.2) permitiu automatizar e fazer um acompanhamento sobre a aplicação em Lógica Modal. O programa completo expande as possibilidades de aplicação implementadas anteriormente para o algoritmo C-IL$^2$P, por agregar as características dos operadores modais.

Partimos da análise de um problema genérico em Lógica Modal para testar os conceitos abordados pelo algoritmo C-IL$^2$P Modal. A partir disso, problemas mais complexos que envolvam o desenvolvimento de raciocínio lógico baseado em modalidades (exemplos apresentados nos apêndices B.1 e B.2) podem ser apreciados.

O problema genérico em Lógica Modal, tomado como exemplo para o teste, foi o mesmo apresentado na Seção 4.4, transcrito novamente abaixo:

mundos: $\omega_1$, $\omega_2$ e $\omega_3$; Relações: $R(\omega_1, \omega_2)$ e $R(\omega_1, \omega_3)$.

$\omega_1$ : r $\rightarrow \Box$ q, $\Diamond$s $\rightarrow$ r;
$\omega_2$ : s;
$\omega_3$ : q $\rightarrow \Diamond$p;

O objetivo desse teste foi aplicar o algoritmo de conversão do sistema C-IL$^2$P Modal a uma situação específica. Essa situação modelada apresenta características comuns a outras modelagens de situações. Temos, como exemplo, os fatos, as condições existentes nos mundos e as relações. A estabilidade do sistema pela sua convergência na resposta (seções 2.3 e **??**) nos indica que a modelagem é válida, para esse caso.

### Procedimentos adotados no teste

O primeiro dos procedimentos foi o de estabelecer as topologias pela aplicação dos algoritmos C-IL$^2$P e C-IL$^2$P Modal. Na seqüência dos procedimentos, ocorreram o teste e a validação. O teste, particularmente, conta com algoritmos auxiliares para o treinamento das RNAs geradas e propagação de valores calculados internos às RNAs.

O algoritmo abaixo é uma forma complementar ao algoritmo C-IL$^2$P Modal utilizado na implementação, pois precisa de recursos usados pela aplicação que não fazem parte do algoritmo original.

Algoritmo auxiliar ao C-IL$^2$P Modal

---

1. Identificar os neurônios rotulados por modalidades, duplicando-os quando estão na entrada da sub-RNA.

2. Identificar as relações entre RNAs existentes.

3. Criar novas sub-RNAs, conforme:

   (a) Para os neurônios identificados por *necessidade*:

      i. Estabelecer conexões e neurônio tipo "M".

      ii. Criar neurônios de destino na RNA de destino, caso ele não exista.

   (a) Para os neurônios identificados por *possibilidade*:

      i. Estabelecer conexões e neurônio tipo "M".

4. Treinar as sub-RNAs estabelecidas, obtendo os valores das conexões.

5. Para cada uma das sub-RNAs geradas relacionadas à *necessidade*, criar uma conexão com valor unitário (=1) e outra a ser treinada, criar o neurônio "$\wedge$" .

6. Para cada uma das sub-RNAs geradas relacionadas à *possibilidade*, criar uma conexão com valor unitário (=1) e outra a ser treinada, criar o neurônio "$\vee$" .

---

A Tabela 6.31 relaciona os antecedentes e respectivos conseqüentes das regras, juntamente com o valor de erro esperado para essa relação lógica.

Tabela 6.31: Entradas e respostas para treinamento das RNAs.

| **r**<br>**q**<br>**$\Diamond$s** | **q**<br>**$\Diamond$p**<br>**r** |
|---|---|
| **Entradas** | **Saídas desejadas** |
| -1 | -1 |
| 1 | 1 |

A Figura 6.5 mostra os pesos e conexões para duas sub-RNAs, comuns na RNA gerada, antes e após o treinamento.

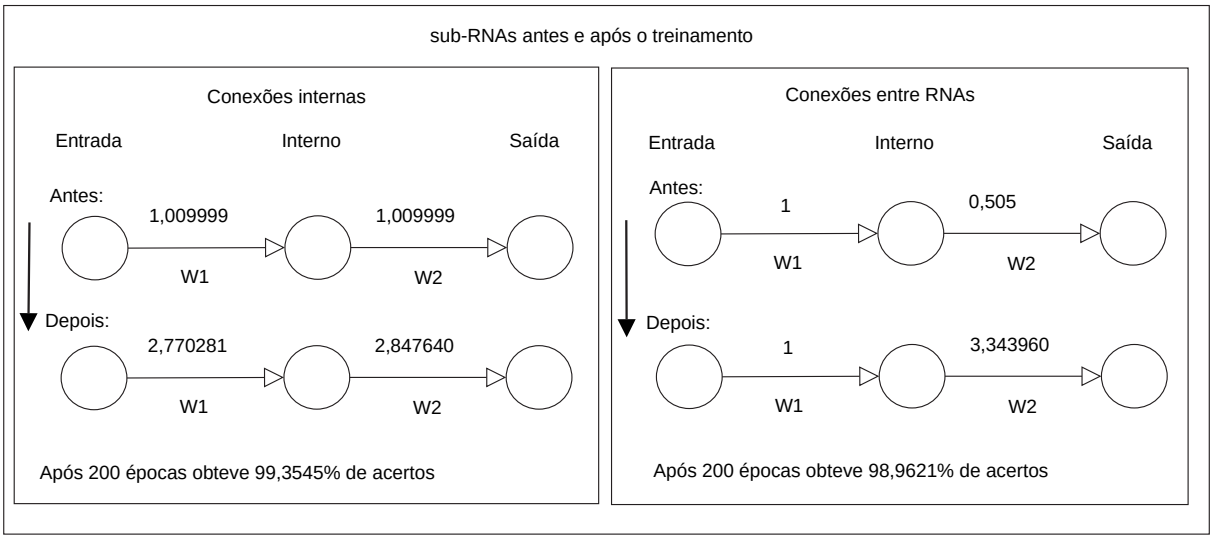


Figura 6.5: Sub-RNAs antes e após o treinamento.

Tabela 6.32: Parâmetros utilizados.

| | |
|---|---|
| Taxa de aprendizagem | variável |
| Taxa de aprendizagem inicial | 0,05 |
| $\beta$ | 1 |
| Coeficiente de regulação do C-IL$^2$P | 2 |
| Número de épocas de treinamento | 100 e 200 |

### Condições do teste

Na Tabela 6.32, são expostos os parâmetros utilizados no teste de aplicação da Lógica Modal. Esses parâmetros foram estabelecidos e seguem a mesma definição dada na Tabela 6.4. Na Tabela 6.33, encontramos os valores calculados pela aplicação dos algoritmos C-IL$^2$P e C-IL$^2$P Modal utilizados nesse teste. Os valores colocados na tabela de parâmetros são mínimos, mas atendem às especificações das fórmulas dos algoritmos.

Tabela 6.33: Parâmetros gerais calculados durante o teste.

| | | | |
|---|---|---|---|
| $A_{min}$ | 0 | $W$ | 1,009999 |
| $\theta_l$ | 0 | $\theta_A$ | 0,505000 |
| $\mu$ | 0 | $\kappa$ | 1 |
| $W^M$ | 0,505000 | | |

Nesse teste, foi atribuído um valor (-1 ou 1) ao átomo "s" pertencente ao mundo 2. Os valores atribuídos ao átomo estão de acordo com a faixa de trabalho da RNA (saídas da função sigmóide). O átomo "s" é o único neurônio de entrada que recebe um valor inicial por ser o único fato do conjunto de mundos.

A Tabela 6.34 mostra o resultado dos átomos (neurônios de saída) desde o início da execução até a estabilidade da rede para cada um dos possíveis valores de "s".

Os neurônios de cada sub-RNA do sistema inicial foram utilizados como medida para verificação dos processos de estabilização, através das iterações. Os valores obtidos são -1, 1 e "x", em

Tabela 6.34: Resultados obtidos pela utilização do fato "s".

| iteração | Átomo | q | r | ♢s | ♢p | q |
|---|---|---|---|---|---|---|
| 1 | **s = 1** | x | 1 | 1 | x | x |
| 2 | **s = 1** | 1 | 1 | 1 | x | 1 |
| 3 | **s = 1** | 1 | 1 | 1 | 1 | 1 |
| 4 | **s = 1** | 1 | 1 | 1 | 1 | 1 |
| 1 | **s = -1** | x | -1 | -1 | x | x |
| 2 | **s = -1** | -1 | -1 | -1 | x | -1 |
| 3 | **s = -1** | -1 | -1 | -1 | -1 | -1 |
| 4 | **s = -1** | -1 | -1 | -1 | -1 | -1 |

que "x" representa um valor instável que será substituído por um outro valor estável durante as iterações.

O teste prosseguiu por várias iterações além da apresentada na Tabela 6.34 para verificar se surgia alguma instabilidade no sistema. Como resultado dessa monitoração, o sistema manteve seus valores estáveis.

### 6.4.2 Análise de resultados

O programa permitiu o acompanhamento passo a passo da aplicação de um exemplo em Lógica Modal, tal qual foi observado nas figuras da Seção 4.4 (Figura 4.3 até a Figura 4.10) e pelos resultados obtidos para esse exemplo (Tabela 6.34).

Após o teste, foi verificado que: (1) Os valores dos limiares de ativação e dos pesos da rede neural calculados pelo programa confirmam os previstos por cálculos manuais. (2) A estabilidade das sub-RNAs ocorreu em conseqüência da convergência ao valor esperado para cada neurônio, atendendo ao definido pela programação lógica.

### 6.4.3 Próximo trabalho

O próximo trabalho é a segunda aplicação do C-IL$^2$P Modal, a qual modela o problema das crianças sujas (*Muddy Children Puzzle*), contido no apêndice B.1. A complexidade desta aplicação é maior, quando comparada com a aplicação anterior. Alguns testes foram executados com esta aplicação, porém como a arquitetura da RNA gerada exige mais pesquisa, principalmente em dois pontos, os resultados obtidos não apresentaram valores consistentes.

Os dois pontos a serem estudados são: (1) A maneira como ocorre a convergência das respostas, em uma RNA com múltiplas camadas. (2) A convergência global do sistema formado pelas três RNAs, assim como a sua inter-relação. A partir desse estudo, poder-se-á reestruturar o programa de forma a atender uma classe mais complexa de problemas.

O aumento da complexidade desta aplicação é percebida nas regras subseqüentes, quando além de termos a lógica proposicional, aparecem os operadores de conhecimento ($K_i$). A Lógica Modal também pode ser admitida na formulação das proposições, dando continuidade a aplicação anterior.

$$r_1^1\colon (K_1 q_1) * (K_1 \neg p_2) * (K_1 \neg p_3) \rightarrow K_1 p_1$$
$$r_2^1\colon (K_1 q_2) * (K_1 \neg p_2) \rightarrow K_1 p_1$$
$$r_3^1\colon (K_1 q_2) * (K_1 \neg p_3) \rightarrow K_1 p_1$$
$$r_4^1\colon (K_1 q_3) \rightarrow K_1 p_1$$

O programa em linguagem lógica tem origem nas regras expressas, segundo [10]. Essas regras se aplicam ao agente[4] 1 (índice "1" na regra $r^1$).

Os átomos "q1, q2 e q3" representam respectivamente que existe uma criança suja, existem duas crianças sujas e todas as três crianças estão sujas. Cada um desses átomos são fatos mutuamente excludentes, estabelecidos antes da execução do programa pela RNA. Os átomos "p1, p2 e p3" representam respectivamente a criança 1, a criança 2 e a criança 3.

Nesse conjunto de regras, a regra:

$$r_1^1\text{: } (K_1 q_1) * (K_1 \neg p_2) * (K_1 \neg p_3) \rightarrow K_1 p_1$$

Significa que: Entre as três crianças existe uma suja, se duas delas não estão sujas ($p_2$ e $p_3$), então a criança que sobrou ($p_1$) só pode estar suja.

As seguintes regras operam em conjunto, pois apenas uma delas não é suficiente para complementar a informação relativa ao conjunto de crianças.

$$r_2^1\text{: } (K_1 q_2) * (K_1 \neg p_2) \rightarrow K_1 p_1$$
$$r_3^1\text{: } (K_1 q_2) * (K_1 \neg p_3) \rightarrow K_1 p_1$$

A criança 1 ($r^1$) sabe que existem duas crianças sujas ($K_1 q_2$); pela Regra $r_2^1$ sabe-se que a criança 2 não está suja ($K_1 \neg p_2$); pela Regra $r_3^1$ a criança 3 não está suja ($K_1 \neg p_3$); isso implica que ela (criança 1) esteja suja ($p_1$).

Por fim, a Regra $r_4^1$ é óbvia. Naturalmente conclui-se que se todas as crianças pertencentes ao conjunto estão sujas ($K_1 q_3$), então a criança 1 também o estará, pois ela faz parte do conjunto.

$$r_4^1\text{: } (K_1 q_3) \rightarrow K_1 p_1$$

Esse programa em linguagem lógica é idêntico para cada agente (criança) e pode ser expresso pela seqüência de regras.

mundos: $\omega_1$, $\omega_2$ e $\omega_3$;
Relações: $R(\omega_1, \omega_2)$; $R(\omega_1, \omega_3)$; $R(\omega_2, \omega_1)$; $R(\omega_2, \omega_3)$; $R(\omega_3, \omega_1)$ e $R(\omega_3, \omega_2)$.

$\omega_1$ :
q1 * ( p2' * ¬p2 * p3' * ¬p3 ) → □p1
q1 * ( (p2' * p3') + (p2' * ¬p3) + (¬p2 * p3') + (¬p2 * ¬p3) ) → ◇p1
q2 * ( p2' * ¬p2 + p3' * ¬p3 ) → □p1
q2 * ( p2' + ¬p2 + p3' + ¬p3 ) → ◇p1
q3 * ( p2 * p3 ) → □p1
q3 * ( (p2 * p3) + (¬p2' * p3) + (p2 * ¬p3') + (¬p2' * ¬p3') ) → ◇p1

¬□p1 → p1
◇p1 + □p1 → p1

Observa-se que: "$\omega_2$" e "$\omega_3$" são equivalentes a "$\omega_1$", quando são feitas as devidas substituições, tais como, no mundo "$\omega_2$", "p1" substitui "p2" e vice-versa, idem para o mundo "$\omega_3$".

[4] Agente é um mundo em particular que representa uma criança do problema.

A Figura 6.6 representa a RNA gerada pela conversão C-IL$^2$P Modal aplicada ao problema das crianças sujas, para primeira criança (Agente 1). A Figura 6.7 mostra todo o conjunto, contendo as três crianças citadas no problema, conforme apêndice B.1.

Espera-se que os passos envolvidos no raciocínio sejam perceptíveis ao observador externo, durante a troca de informação entre os agentes na execução desse exemplo, permitindo seu acompanhamento. Também é esperado que as RNAs e, por conseqüência, o programa lógico atinjam a estabilidade, tal como ocorreu no exemplo anterior.

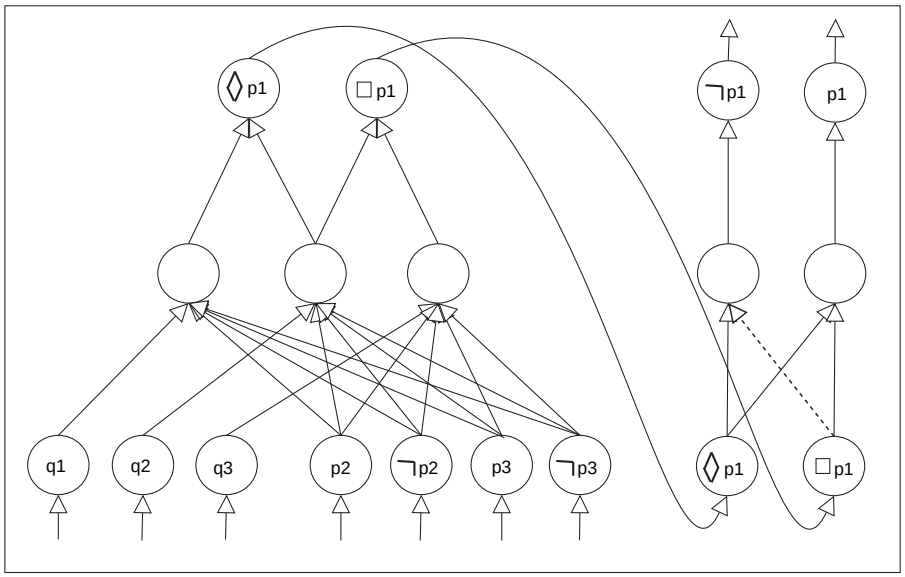


Figura 6.6: RNA modelando um agente para a segunda aplicação Modal.

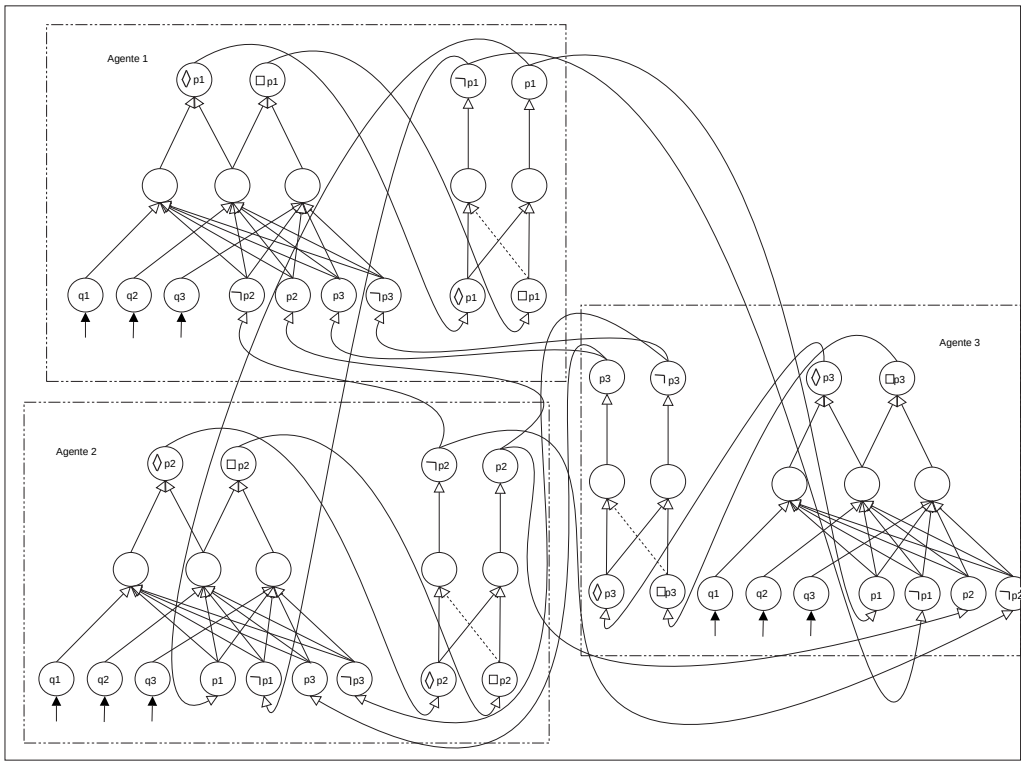


Figura 6.7: Conjunto de RNAs para o problema das crianças sujas.

# Capítulo 7

# Conclusões

"*A meta final de qualquer pesquisa não é a objetividade, mas a verdade.*"
Helene Deutsch (1884 - 1982) - Psiquiatra americana

Este capítulo final foi dividido em duas seções. Primeiramente, na Seção 7.1, são relacionadas as principais contribuições e os resultados mais relevantes do trabalho, juntamente com a análise desses resultados. Algumas reflexões sobre a análise feita, considerações e diretrizes (ou idéias) para próximos trabalhos são apresentadas na Seção 7.2.

## 7.1 Sobre este trabalho

### 7.1.1 Principais contribuições

As contribuições trazidas pela execução do trabalho abordado na dissertação podem ser relacionadas conforme segue:

- Aplicaram-se os algoritmos Neuro-Simbólicos em três casos específicos:
  1. No reconhecimento de promotores de DNA.
  2. No domínio completo gerado por regra lógica conhecida.
  3. No modelamento de uma situação genérica expressa em linguagem lógica, definida por Lógica Modal.
- Exemplificou os domínios incompleto e completo sobre um conhecimento específico.
- Dos testes comparativos entre as formas de se estabelecer as RNAs e os respectivos aprendizados, podemos dizer que:
  - As conclusões apresentadas no artigo original sobre o algoritmo C-IL$^2$P [12] foram reforçadas, estando elas corretas no que diz respeito ao fato de esse algoritmo ser um meio eficaz para se representar um conjunto de regras.
- A primeira implementação do algoritmo C-IL$^2$P Modal foi feita.

- Um protótipo a um ambiente computacional visando a integrar as implementações dos algoritmos Neuro-Simbólicos foi desenvolvido.
- Esta pesquisa serve de base para futuros trabalhos na área de INS.

O estudo e o trabalho realizados resultaram na aplicação dos testes apresentados no capítulo 6. Esses testes enfocaram três tópicos principais dizendo respeito à implementação e ao uso de algoritmos Neuro-Simbólicos, que são: o algoritmo C-IL$^2$P, base de dados artificial completa, e o algoritmo C-IL$^2$P Modal.

### 7.1.2 Análise geral sobre os resultados

Tabela 7.1: Resumo dos resultados dos testes.

| **Tópico do teste** | **Resumo dos resultados** |
|---|---|
| Reconhecimento de promotores | A utilização do C-IL$^2$P melhora o desempenho da aprendizagem. |
| Base de dados completa | O C-IL$^2$P representa as regras com 100% de fidelidade. |
| Aplicando a Lógica Modal | O C-IL$^2$P Modal conduziu à obtenção de um sistema estável. |

A Tabela 7.1 sumariza os resultados dos testes. Como aponta a tabela, o algoritmo de conversão do sistema C-IL$^2$P, aliado ao algoritmo de aprendizagem por retropropagação, produz resultados melhores do que simplesmente aplicar o algoritmo de aprendizagem a uma RNA iniciada com pesos aleatórios.

Outro resultado mostrado na Tabela 7.1 foi o completo domínio da RNA gerada pelo C-IL$^2$P após o processo de aprendizagem sobre o banco de dados gerado pela regra lógica. Os resultados obtidos também nos indicam que, dentro de um domínio de conhecimento com limites definidos, podemos descobrir fatos não-declarados ou não-existentes na informação inicial.

O terceiro resultado sumarizado pela Tabela 7.1 mostra que se obteve confirmação sobre o que era previsto pelo artigo [11] e pelos testes preliminares executados em simuladores [10]. A execução da implementação do C-IL$^2$P Modal foi apenas o passo seguinte nessa linha de desenvolvimento. Ainda existem muitas questões em aberto e as inúmeras pesquisas nessa área ainda estão sendo desenvolvidas (ex.: [10]). Apesar de a Lógica Modal ter sido o último tópico desenvolvido, foi necessário o desenvolvimento dos dois primeiros, pois havia dependência entre eles em uma ordem seqüencial.

Os domínios de conhecimento incompletos (Promotores de DNA) e completos sob restrição de caso (Regra Lógica) foram comparados e também testados para os algoritmos Neuro-Simbólicos. Um banco de dados formado por regras empíricas foi revisado e aprimorado, estabelecendo condições para a formação de um novo banco de dados mais sólido e consistente, no qual se pode aplicar a extração do conhecimento (não tratada pelo trabalho). A implementação do algoritmo C-IL$^2$P Modal permite o modelamento de situações envolvendo agentes, com o detalhamento do raciocínio lógico evolutivo, proposta para trabalhos futuros.

## 7.2 Considerações finais

**Discussões**

A partir do plano de trabalho proposto no mestrado, houve algumas mudanças nas trajetórias de desenvolvimento, devido a opções e a fatores operacionais. Não ocorreram grandes imprevistos nas implementações, somente dificuldades resultantes da inexperiência na área.

Basicamente, aprendeu-se a construir e a treinar RNAs e os conceitos lógicos que constam nas implementações. As RNAs geradas são diferentes da original. Essa mudança indica que houve aquisição de informações, que devem ter sua corretude verificada por processamento simbólico. À medida que os trabalhos eram executados, a compreensão à melhor implementação tornava-se mais adequada, conforme os objetivos do trabalho.

Os resultados obtidos refletem uma parte do trabalho na qual o esforço pode ser medido. Em virtude disso, cada etapa de desenvolvimento foi marcada pelo programa que estava sendo desenvolvido naquele momento. A maior parte do trabalho esteve concentrada no entendimento e na aplicação do algoritmo C-IL$^2$P, razão pela qual nisso se empenhou mais tempo.

As implementações em INS são importantes para expressar as idéias de uma forma dinâmica, porque percebemos, através dos programas desenvolvidos, como os conceitos apresentados nos artigos agem. Através das implementações, percebeu-se a inter-relação dos conceitos tratados. Também, tornou-se possível o estudo do comportamento de parâmetros presentes nos algoritmos e o estudo da integração como um todo.

Percebeu-se a dificuldade que existe na conversão de idéias sob forma de teorias em implementações que apresentassem resultados consideráveis; porém, o tempo empregado na elaboração e detalhamento das idéias é vital ao bom desenvolvimento das implementações. Outro ponto importante é a busca adequada por informações relevantes ao desenvolvimento. Observa-se que se faz necessária uma base sólida nos campos da Inteligência Artificial Simbólica e Conexionista, ao pesquisador que intencionar um trabalho na área Neuro-Simbólica, seguindo a linha de pesquisa desta dissertação.

O trabalho no contexto da IA procurou mostrar a correspondência de conceitos entre os seus paradigmas, o Simbólico, o Conexionista e principalmente pela união dos dois através da Integração Neuro-Simbólica. Um dos pontos importantes foi abordar a conversão do conhecimento expresso em forma simbólica em uma RNA com a finalidade de aprimoramento desse.

Algumas preocupações estiveram presentes durante as implementações, tais como: a definição dos limites de integração entre os dois paradigmas originais; quais as influências de um paradigma sobre o outro; e como contribuir fornecendo resultados que conduzam à justificativa do estabelecimento da INS. A importância desses temas merece um estudo mais aprofundado, dada a sua relevância para as implementações tratadas no trabalho, assim como para os problemas relacionados com o conhecimento de forma geral.

A aplicabilidade da INS pode acontecer nos mais diversos campos do conhecimento. Existem algumas idéias, tais como o fato de a descrição em linguagem natural ser empregada como entrada e saída para a RNA, tornando o produto e as etapas do processamento da RNA perfeitamente compreensíveis ao ser humano.

Outros exemplos estendem-se desde a percepção e sensibilidade em robôs até seu comportamento em grupo. Um robô pode desenvolver raciocínios lógicos mais aprimorados através da utilização de RNAs, após a monitoração humana sobre esses raciocínios, até a aquisição de sua autonomia. Interação entre agentes em uma sociedade através da modelagem BDI ("*Believe, Desire and Intention*") pode possuir componentes neuro-simbólicos.

### Próximos trabalhos

Como pudemos perceber, existem muitos trabalhos a serem feitos. Abaixo, estão relacionados alguns caminhos nos quais as pesquisas podem ser orientadas.

- Execução de outras aplicações para a Lógica Modal, tais como os casos apresentados nos apêndices B.1 e B.2
- Propostas em conjunto com os autores dos algoritmos direcionam ao trabalho com o C-IL$^2$P para a pesquisa dos efeitos do $A_{min}$ sobre os pesos $W$ e $W^M$. Assim, podemos definir padrões ou fórmulas que apontem para valores ótimos para estas variáveis.
- Outra proposta sugerida pelos autores foi a de investigar a discretização de variáveis, em que a Lógica "Fuzzy" [23] pode ser utilizada. Os dados, quando inseridos às RNAs, podem representar um certo tipo de variável conforme o seu valor.
- Existe a possibilidade de empregar variáveis lógicas na programação a ser convertida em RNA. Exemplo: a mesma variável pode assumir, em instantes diferentes, o valor de $\Box$ e $\Diamond$.
- A Lógica Temporal aplicada ao sistema Modal também se faz necessária. Ainda não foram exploradas as potencialidades evolutivas dos sistemas. Uma rápida abordagem aparece no apêndice B.3.
- O emprego de outros tipos de RNAs e algoritmos de aprendizagem podem ser explorados, assim como aumentar a capacidade de expressão das RNAs apresentadas na dissertação.

# Referências Bibliográficas


[1] A. S. Al-homoud and W. W. Tahtamoni. *SARETL: an expert system for probabilistic displacement-based dynamic 3-D slope stability analysis and remediation of earthquake triggered landslides.* Springer-Verlag Heidelberg Editors, 1999.

[2] S. L. Azevedo. Um sistema especialista para escolha do tipo de fundações. *Teoria e Prática na Engenharia Civil*, 1:77–86, 2000.

[3] M. L. Bigge. *Teoria da Aprendizagem para professores.* Editora da Universidade de São Paulo, 1977.

[4] O. Boz. *Knowledge Integration and Rule Extraction in Neural Networks.* PhD thesis, Lehigh University - EECS, 1995.

[5] B. G. Buchanan and E. H. Shortliffe. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Edited by Bruce G. Buchanan and Edward H. Shortliffe, 1984.

[6] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. *Cooperative Mobile Robotics: Antecedents and Directions.* Kluwer Academic Publishers., 1997.

[7] Artificial Intelligence Center. Natural Language Program. http://www.ai.sri.com/natural-language/, acessado em setembro de 2004.

[8] B. F. Chellas. *Modal logic: an introduction.* Cambridge University Press., 1980.

[9] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125(1-2):155–207, 2001.

[10] A. S. d'Avila Garcez, L. C. Lamb, K. Broda, and D. M. Gabbay. Applying connectionist modal logics to distributed knowledge representation problems. *International Journal on Artificial Intelligence Tools*, 13(1):115–139, 2004.

[11] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. A Connectionist Inductive Learning System for Modal Logic Programming. Technical report, Department of Computing, Imperial College, London, 2002.

[12] A. S. d'Avila Garcez and G. Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, 11(1):59–77, 1999.

[13] H.L. Dreyfus and S.E. Dreyfus. Making a mind versus modeling a brain: Artificial intelligence back at a branchpoint. In *Proceedings of the American Academy of Arts and Sciences*, pages 15–43, 1988.

[14] B. Duin. The Pattern Recognition Files. http://www.ph.tn.tudelft.nl/PRInfo/, acessado em setembro de 2004.

[15] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge.* MIT Press., 1995.

[16] I. Fischer. Java Neural Networks Simulator - User Manual, Version 1.1. http://www-ra.informatik.uni-tuebingen.de, acessado em setembro de 2004.

[17] L. M. Fu. Knowledge-based connectionism for revising domain theories. *IEEE transations on System, Man and Cybernetics*, 23:173–182, Jan/Feb 1993.

[18] A. Giacometti. *Modèles Hybrides de l'Expertise.* PhD thesis, IMAG - Lab. LIFIA, Grenoble/ENST Paris - França, 1995.

[19] S. Haykin. *Neural Networks: A comprehensive foundation.* Prentice-Hall Inc., 1999.

[20] J. Hintikka. *Knowledge and Belief.* Cornell University Press, 1962.

[21] B. G. Horne, C. L. Giles, and H. T. Siegelmann. Computational capabilities of recurrent NARX neural networks. Technical report, Institute for Advanced Computer Studies, University of Maryland, 1995.

[22] Free Software Foundation Inc. Plataforma de desenvolvimento Dev-C++. http://www.bloodshed.net, acessado em setembro de 2004.

[23] B. Kosko. *Neural Networks and Fuzzy Systems.* Prentice-Hall Inc., 1992.

[24] S. A. Kripke. *Semantical considerations on modal logic*, pages 63–72. In [27], 1971.

[25] Y. Lallement and F. Alexandre. *Cognitive Aspects of Neurosymbolic Integration*, chapter 4. In [35], 1997.

[26] Y. Lallement, M. Hilario, and F. Alexandre. Neurosymbolic Integration: Cognitive Grounds and Computational Strategies. In *World Conference on the Fundamentals of Artificial Intelligence, Paris, France*, pages 193–203, 1995.

[27] L. Linsky. *Reference and Modality.* Ed. London: Oxford University Press, 1971.

[28] University of California Irvine. Database set from UCI. http://www.ics.uci.edu/∼mlearn/MLSummary.html, acessado em setembro de 2004.

[29] Generation 5 Organization. At the forefront of Artificial Intelligence. http://www.generation5.org/xornet.shtml, acessado em setembro de 2004.

[30] F. Osório e R. Vieira. Tutorial: Sistemas Híbridos Inteligentes. XIX CONGRESSO DA S.B.C. (ENIA'99). 1999. http://www.inf.unisinos.br/∼osorio, acessado em setembro de 2004.

[31] S. Raudys. *Statistical and Neural Classifier.* Springer-Verlag Heidelberg Editors., 2001.

[32] D. E. Rumelhart, G .E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation.* MIT Press, 1986.

[33] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall Inc., 1995.

[34] F. A. Silva, M. Roisenberg, and J. M. Barreto. Redes neurais hierarquicas para implementação de comportamentos em agentes autonômos. *Anais do XIII Congresso Brasileiro de Automática - CBA 2000*, páginas 1473-1478, 2000.

[35] R. Sun and F. Alexandre. *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches.* Lawrence Erlbaum Associates, 1997.

[36] G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence Journal*, 70(1):119–165, 1994.

[37] Carnegie Mellon University. Expert Systems. http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/expert/systems/0.html, acessado em setembro de 2004.

[38] Knowledge System Laboratory – Stanford University. Open Knowledge Base Connectivity, JTP Theorem Prover e World Fact Knowledge Base. http://www-ksl.stanford.edu/sns.shtml, acessado em setembro de 2004.

[39] University of Stuttgart and University of Tubingen. Institute for Parallel and Distributed High Performance Systems and Wilhelm-Schickard-Institute for Computer Science. *Stuttgart Neural Network Simulator - User Manual, Version 4.2*, 1998.

# Apêndice A

# Algoritmos completos

## A.1 O Algoritmo C-IL$^2$P

1. Calcular o maior número entre a quantidade de literais e a quantidade de cláusulas com o mesmo conseqüente, do programa lógico.

$$MAX_p(\vec{\kappa}, \vec{\mu}) \tag{A.1}$$

2. Calcular o valor de ativação mínimo dos neurônios.

$$A_{min} > \frac{MAX_p(\vec{\kappa}, \vec{\mu}) - 1}{MAX_p(\vec{\kappa}, \vec{\mu}) + 1} \tag{A.2}$$

3. Calcular os pesos das conexões dos neurônios.

$$W \geq \frac{2}{\beta} \frac{ln(1 + A_{min}) - ln(1 - A_{min})}{MAX_p(\vec{\kappa}, \vec{\mu})(A_{min} - 1) + A_{min} + 1} \tag{A.3}$$

4. Para cada cláusula do programa lógico,

   (a) Adicione um neurônio à camada interna.

   (b) Conecte cada neurônio da camada de entrada ao neurônio da camada interna criado. Se o literal não estiver negado, o peso será positivo, se o literal for negado, o peso será negativo.

   (c) Conectar o neurônio criado com o neurônio de saída.

   (d) Definir o limiar do neurônio da camada interna.

$$\theta_l = \frac{(1 + A_{min})(\kappa_l - 1)}{2} W \tag{A.4}$$

   (e) Definir o limiar do neurônio da camada de saída.

$$\theta_A = \frac{(1 + A_{min})(1 - \mu_l)}{2} W \tag{A.5}$$

5. Definir as funções de ativação dos neurônios da camada de entrada. Dessa maneira, a interpretação do programa lógico será feita pela ativação conforme o vetor de entrada.

6. Definir as funções de ativação dos neurônios das camadas internas e de saída. Após isso, um algoritmo de aprendizagem pode ser aplicado.

7. Colocar todas os pesos da conexões restantes para zero, se a RNA já estiver totalmente conectada conforme o programa lógico.

## A.2 O Algoritmo C-IL$^2$P Modal

1. Calcular os limiares de ativação ($A_{min}$ e $\theta$) e os pesos ($W$), de forma idêntica à feita no sistema $C - IL^2P$. Os novos Pesos Modais ($W^M$) serão calculados, baseando-se nesses valores.

$$A_{min} > \frac{MAX_p(\vec{\kappa}, \vec{\mu}, n) - 1}{MAX_p(\vec{\kappa}, \vec{\mu}, n) + 1} \tag{A.6}$$

$$W^M > h^{-1}(A_{min}) + \mu W + \theta \tag{A.7}$$

2. Para cada cláusula:

   (a) Renomear os literais segundo sua modalidade.

   (b) Proceder segundo o algoritmo de conversão $C - IL^2P$.

3. Para cada neurônio de saída do literal possibilidade em uma rede neural:

   (a) Adicionar o neurônio equivalente ao literal modal interno em cada sub-rede, de tal maneira que a relação entre o conjunto de cláusulas da rede e da sub-rede persista.

   (b) Integrar a função degrau ao neurônio do literal modal, como função de ativação.

   (c) Conectar com peso "1" o neurônio do literal possibilidade na rede.

   (d) O limiar da modalidade deve variar de "$-1$" até Amin. (Ativação mínima).

   (e) Conectar os literais modais da sub-rede conforme os pesos modais.

4. Para cada neurônio de saída do literal necessidade em uma rede neural:

   (a) Adicionar o neurônio equivalente ao literal modal interno em cada sub-rede, de tal maneira que a relação entre o conjunto de cláusulas da rede e da sub-rede persista.

   (b) Integrar a função degrau ao neurônio do literal modal, como função de ativação.

   (c) Conectar com peso "1" o neurônio do literal necessidade na rede.

   (d) O limiar da modalidade deve variar de "$-1$" até Amin. (Ativação mínima).

   (e) Conectar os literais modais da sub-rede conforme os pesos modais.

5. Para cada neurônio de saída da sub-rede, mantendo a relação entre o conjunto de cláusulas da rede e da sub-rede:

(a) Adicionar um neurônio interno $\vee$ à rede.

(b) Integrar a função degrau (função de ativação) ao neurônio interno $\vee$.

(c) Para cada neurônio da rede, do tipo possibilidade:

i. Conectar com peso "1" os literais da sub-rede aos literais do neurônio interno $\vee$.

ii. Definir os limites de ativação do neurônio interno $\vee$.

iii. Conectar os literais do neurônio interno $\vee$ aos literais de possibilidade da rede conforme os pesos modais.

6. Para cada neurônio de saída da sub-rede, mantendo a relação rede e sub-rede:

(a) Adicionar um neurônio interno $\wedge$ à rede.

(b) Integrar a função degrau (função de ativação) ao neurônio interno $\wedge$.

(c) Para cada neurônio de saída da rede, do tipo necessidade:

i. Conectar com peso "1" os literais da sub-rede aos literais do neurônio interno $\wedge$.

ii. Definir os limites de ativação do neurônio interno $\wedge$.

iii. Conectar os literais do neurônio interno $\wedge$ aos literais de necessidade da rede conforme os pesos modais.

## A.3 Algoritmos para as fórmulas

As fórmulas existentes em cada algoritmo implementado compuseram trechos de código, em que os cálculos são estabelecidos de modo seqüencial. Cada conjunto de cálculos é executado passo a passo para a formação do resultado. As fórmulas estão relacionadas abaixo com o detalhamento nos algoritmos subseqüentes.

### Cálculo do $A_{min}$

O $A_{min}$ é o menor valor aceitável para a ativação do neurônio. O $A_{min}$ situa-se normalmente entre (-1 e 0) ou (0 e 1).

$$A_{min} > \frac{MAX_p(\vec{\kappa}, \vec{\mu}, n) - 1}{MAX_p(\vec{\kappa}, \vec{\mu}, n) + 1} \tag{A.8}$$

Algoritmo para cálculo do $A_{min}$

1. Procurar e armazenar o maior entre os valores ($MAXp$):

(a) Em todos os mundos ($n$)

i. Os literais de uma cláusula ($\kappa$).

ii. As cláusulas com o mesmo conseqüente ($\mu$).

2. Calcular e armazenar:

(a) $MAXp-1$, no numerador.

(b) $MAXp$+1, no denominador.

3. Dividir o numerador pelo denominador.

4. O $A_{min}$ será um valor superior ao calculado.

---

### Cálculo do peso de uma sinapse segundo o C-IL$^2$P

Como já foi dito anteriormente, cada uma das conexões entre os neurônios de uma RNA possui um peso $W$ atribuído a ela. Esse peso pode ser calculado a partir da fórmula abaixo, relativa a um mundo em particular.

$$W \geq \frac{2}{\beta} \frac{ln(1 + A_{min}) - ln(1 - A_{min})}{MAX_p(\vec{\kappa}, \vec{\mu})(A_{min} - 1) + A_{min} + 1} \tag{A.9}$$

Algoritmo para cálculo dos pesos

---

1. Se $A_{min}$ for superior ou igual a "1", então, recalcular $A_{min}$.

2. Dois valores são calculados e armazenados a partir de $A_{min}$.

   (a) $A_{min}$ acrescido de "1".

   (b) $A_{min}$ subtraído de "1".

3. Calcular e armazenar:

   (a) Para o numerador:

      i. A diferença entre o logaritmo natural de 2a e o logaritmo natural do módulo de 2b.

      ii. Multiplicar o valor anterior por dois "2".

   (b) Para o denominador:

      i. Multiplicar o parâmetro de inclinação ($\beta$) da função sigmóide pelo maior dos valores $MAX_p(.)$ entre $\kappa$ e $\mu$ em um mundo.

      ii. Multiplicar o valor calculado no passo anterior por 2b e por "$-1$".

      iii. Somar ao valor calculado a 2a.

4. Dividir o numerador pelo denominador.

5. O valor do peso $W$ deve ser superior ou igual ao calculado.

---

### Calculo dos valores de limiares ($\theta's$)

Existem dois tipos de limiares a serem calculados segundo o algoritmo C-IL$^2$P: os limiares dos neurônios internos $\theta_l$ e os dos neurônios de saída $\theta_A$. Esses são calculados por:

$$\theta_l = \frac{(1 + A_{min})(\kappa_l - 1)}{2} W \tag{A.10}$$

$$\theta_A = \frac{(1 + A_{min})(1 - \mu_l)}{2} W \tag{A.11}$$

Algoritmo para cálculo dos limiares

1. Adicionar o valor "1" ao $A_{min}$.
2. Multiplicar o valor calculado no passo 1 pelo peso $W$ da RNA.
3. Dividir o valor calculado no passo 2 pelo valor 2.
4. Conforme o limiar calculado no passo 3 fazer:
   (a) Se limiar interno: O número de literais da cláusula subtraído em "1".
   (b) Se limiar de saída: "1" subtraído o número de cláusulas com o mesmo conseqüente.
5. Multiplicar o valor calculado no passo 3 pelo valor calculado no passo 4.

### Cálculo do peso Modal por RNA segundo o C-IL$^2$P Modal

Novos neurônios são gerados após a aplicação do algoritmo C-IL$^2$P Modal. Em conseqüência, novas conexões passam a existir na RNA. O peso dessas novas conexões é calculado por:

$$W^M > h^{-1}(A_{min}) + \mu W + \theta \tag{A.12}$$

Algoritmo para cálculo dos pesos adicionais

1. Para cada uma das RNAs calcular:
   (a) O inverso da função sigmóide ($h^{-1}()$) aplicado ao $A_{min}$.
   (b) O número de cláusulas com o mesmo conseqüente ($\mu$) multiplicado pelo peso primário da RNA.
   (c) O limiar do neurônio de saída.
2. Somar todos os valores calculados.
3. O peso Modal $W^M$ será um valor superior ao calculado no passo anterior (passo 2).

# Apêndice B

# Aplicações do algoritmo modal

## B.1 Problema das crianças sujas

### B.1.1 Componentes da situação

Um pai e as crianças que se sujaram.

### B.1.2 Situação

O pai afirma a seus filhos que, ao menos, um deles está com as costas sujas. As crianças não podem ver a sua própria sujeira, somente a dos outros, e nem saber quantas delas estão sujas. O pai pergunta a cada um de seus filhos: "Você saberia dizer se está sujo ou não?".

### B.1.3 Solução

Se o número de crianças for dois, então, cada criança saberá (necessariamente) qual a condição da outra; porém, não sabe responder qual é a sua condição, pois não pode ver a suas próprias costas (existem possibilidades). A segunda criança, na sua vez de responder ao questionamento, sabe que se a primeira a tivesse visto limpa, responderia afirmativamente ao pai, declarando-se suja. Logo, em caso de resposta negativa da primeira criança, a segunda criança resta responder afirmativamente e que está suja, revelando também a condição da primeira, pois a está vendo.

Tabela B.1: Representação do problema das crianças sujas.

| 1ª Criança | 2ª Criança | 3ª Criança | 1ª Vez que perg. | 2ª Vez que perg. | 3ª Vez que perg. |
|---|---|---|---|---|---|
| Limpa | Limpa | Limpa | impossível | impossível | impossível |
| Limpa | Limpa | Suja | ? | ? | possível |
| Limpa | Suja | Limpa | ? | Sim | - |
| Limpa | Suja | Suja | ? | ? | possível |
| Suja | Limpa | Limpa | Sim | - | - |
| Suja | Limpa | Suja | ? | ? | possível |
| Suja | Suja | Limpa | ? | ? | impossível |
| Suja | Suja | Suja | ? | ? | possível |

Seguindo o raciocínio, se o número de crianças for três, somente a terceira criança tem *necessariamente* condições de responder corretamente, através da dedução. As outras crianças somente tem uma *possibilidade* de acertar, entre as sete. A Tabela B.1 esquematiza o raciocínio

desenvolvido para esse caso. "perg.", na tabela, é a abreviatura de "pergunta do pai à criança". O símbolo "?" significa que a criança questionada não sabe a resposta.

A afirmação de que pelo menos uma das crianças está suja, torna impossível a primeira condição da Tabela B.1, que afirma que todas estão limpas. A primeira e a segunda criança somente podem responder se virem as outras crianças limpas, assim, concluirão, que estão sujas. A terceira criança é única que pode deduzir a resposta sem se basear apenas na observação.

Quando é a vez da terceira criança responder, haverá cinco possibilidades, ela saberá que dessas a que indica que ela está limpa é impossível, baseado no raciocínio da segunda criança, que é: "estou vendo que a terceira criança está suja, se eu (segunda criança), estivesse limpa, a primeira criança já teria respondido". Então, a terceira criança sabe que ela só pode estar suja, e basta verificar visualmente a condição das outras, para obter a resposta entre as quatro possibilidades restantes.

Essa modelagem se repete conforme o aumenta do número de crianças. A resposta torná-se de conhecimento comum, logo após a penúltima criança ter respondido. Quando o número de perguntas iguala-se ao número de crianças, todas elas tem condição de avaliar as *possibilidades*, eliminando as que são inválidas; porém, a resposta correta será dada somente pela criança da vez (aquela que ainda não respondeu).

## B.2 Problema dos três homens sábios

### B.2.1 Componentes da situação

Um rei, três homens sábios, dois chapéus brancos e três vermelhos.

### B.2.2 Situação

O rei coloca um chapéu de qualquer uma das cores em cada um dos sábios. Cada um do sábios não consegue ver a cor de seu próprio chapéu somente a cor dos chapéus dos outros dois. O rei faz a seguinte pergunta a cada um dos sábios: Você sabe a cor do chapéu que está na cabeça de cada um?

### B.2.3 Solução

O raciocínio lógico seguido neste caso depende da resposta de cada um dos sábios. As modalidades necessidade e possibilidade integram esse raciocínio. As *possibilidades* diminuem na medida em que as respostas dos sábios passam a ser de conhecimento comum.

- Quando a pergunta é feita ao primeiro sábio, ele consegue ver o que os outros dois estão usando. Se ele tiver sorte, verá dois chapéus brancos e estará apto a responder corretamente ao rei. O sábio *necessita* dessa condição para estar certo.

- Quando a pergunta é feita ao segundo sábio, ele tem a seu favor o fato de conseguir ver a cor do chapéu dos outros sábios e ele sabe a resposta do sábio anterior. Com isso, se tiver sorte, verá o terceiro sábio com chapéu branco; assim, haverá duas *possibilidades* de ele estar certo.

- Quando a pergunta é feita ao terceiro sábio, ele *necessariamente* saberá a resposta, pois tem a seu favor que: - Se os outros dois não responderam, é porque ele não está usando o chapéu branco, e sim, o vermelho

## B.3 Evolução temporal

Notamos, nos exemplos apresentados, a redução das *possibilidades* e a conseqüente apuração da resposta *necessária*, na medida em que as situações evoluem. Na maioria dos casos, existe convergência do sistema para um ponto estável.

Como foi sugerido em [10], além dos algoritmos presentes nesta dissertação, faz ser necessária a existência de um outro algoritmo: o algoritmo Temporal. Esse algoritmo poderia ser desenvolvido pela inclusão de mais uma modalidade: a modalidade para controle temporal, como *próximo passo*.

A Figura B.1 generaliza o que ocorre nas situações apresentadas para os problemas anteriores (B.1 e B.2). A figura mostra as seqüências de interações entre agentes e a evolução temporal de situações.

O agente supervisor, que é o rei em uma situação e o pai em outra, questiona um a um os outros agentes. À medida que os questionamentos vão sendo feitos, o conhecimento comum converge para a solução, excetuando-se o caso em que a resposta é óbvia. Levam-se em conta, em todos os casos, as inferências dos agentes sobre o conhecimento comum e o conhecimento que cada agente tem sobre a situação.

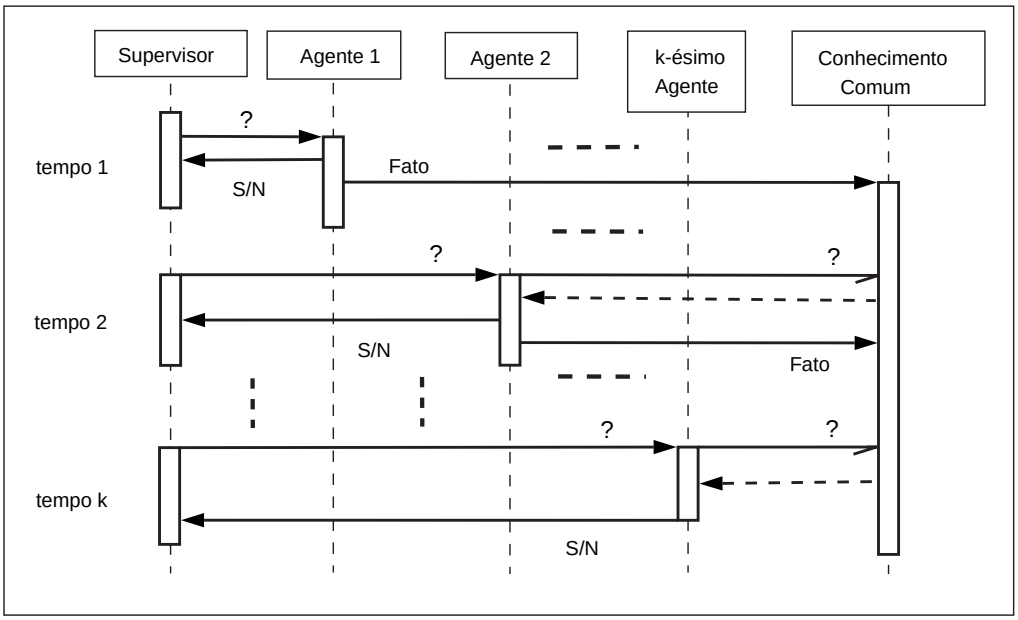


Figura B.1: Diagrama seqüencial e evolutivo.