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

RAFAEL VERGARA BORGES

Investigações sobre raciocínio e aprendizagem temporal em modelos conexionistas

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

Prof. Dr. Luís da Cunha Lamb
Orientador

Porto Alegre, março de 2007

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Borges, Rafael Vergara

Investigações sobre raciocínio e aprendizagem temporal em modelos conexionistas / Rafael Vergara Borges. – Porto Alegre: PPGC da UFRGS, 2007.

77 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Luís da Cunha Lamb.

1. Integração neuro simbólica. 2. Lógica temporal. 3. Aprendizagem empírica. I. Lamb, Luís da Cunha. II. Título.

AGRADECIMENTOS

Primeiramente, agradeço à minha família por tudo o que me proporcionaram, desde o apoio incondicional em todos os momentos às oportunidades de desenvolvimento pessoal que sempre me proveram. Ao meu pai, especialmente pelas lições que, sejam por palavras ou por exemplos, sempre guiaram minha identidade moral. À minha mãe, por todo o zelo, carinho e atenção que sempre dispensou, muitas vezes deixando a si própria de lado pelo bem estar dos filhos. À minha irmã, Raquel, pelo ombro amigo, nunca negando um abraço ou uma palavra bem humorada quando necessários, e ao pequeno Bolívar, cuja imagem sempre serviu como maior motivação para a realização deste trabalho.

Agradeço também a todos amigos, sejam os velhos amigos que ficaram, ou os novos que conheci pelo caminho. Agradeço ao Mário, à Marília, à Carla, à Carolina, à Thaísa, ao Arthur, ao Carlos, ao Pablo e a tantos outros que, nos corredores da UFRGS, dividiram um chimarrão e uma boa companhia em momentos de stress ou de alegria. Aos inúmeros amigos que ficaram mais distantes, e ainda assim sempre confiaram e me apoiaram, bem como à Luciane, à Fátima, à Lutiéllen, à Flávia e outros que estiveram geograficamente mais próximos durante esta trajetória (perdão se esqueci alguém).

Por fim, agradeço ao meu orientador Luís Lamb que, mesmo com todas as diferenças, sempre confiou em meu trabalho, e que foi diretamente responsável pelo crescimento pessoal e profissional nestes anos. Deixo também um agradecimento especial à primeira capital Farroupilha, e aos seus filhos Moser e Lorenzo, que além de companheiros "mistério" para dividir um apartamento, são amigos que ficarão marcados em minha vida por tudo o que vivemos neste período.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADALINE	ADAptive Linear neuron
BPTT	BackPropagation Through Time
CILP	Connectionist Inductive Learning and logic Programming
CML	Connectionist Modal Logic
CTLK	Connectionist Temporal Logic of Knowledge
IA	Inteligência Artificial
KBANN	Knowledge Based Artificial Neural Networks
MLP	Multi Layer Perceptron
NARX	Non-linear Auto Regressive with eXogenous inputs
SCTL	Sequential Connectionist Temporal Logic
TREPAN	TREes PArroting Networks

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A inteligência computacional é considerada por diferentes autores da atualidade como o *destino manifesto* da Ciência da Computação. A modelagem de diversos aspectos da cognição, tais como aprendizagem e raciocínio, tem sido a motivação para o desenvolvimento dos paradigmas simbólico e conexionista da inteligência artificial e, mais recentemente, para a integração de ambos com o intuito de unificar as vantagens de cada abordagem em um modelo único. Para o desenvolvimento de sistemas inteligentes, bem como para diversas outras áreas da Ciência da Computação, o tempo é considerado como um componente essencial, e a integração de uma dimensão temporal nestes sistemas é fundamental para conseguir uma representação melhor do comportamento cognitivo.

Neste trabalho, propomos o SCTL (Sequential Connectionist Temporal Logic), uma abordagem neuro-simbólica para integrar conhecimento temporal, representado na forma de programas em lógica, em redes neurais recorrentes, de forma que a caracterização semântica de ambas representações sejam equivalentes. Além da estratégia para realizar esta conversão entre representações, e da verificação formal da equivalência semântica, também realizamos uma comparação da estratégia proposta com relação a outros sistemas que realizam representação simbólica e temporal em redes neurais.

Por outro lado, também descrevemos, de foma algorítmica, o comportamento desejado para as redes neurais geradas, para realizar tanto inferência quanto aprendizagem sob uma ótica temporal. Este comportamento é analisado em diversos experimentos, buscando comprovar o desempenho de nossa abordagem para a modelagem cognitiva considerando diferentes condições e aplicações.

Palavras-chave: Integração neuro simbólica, Lógica temporal, aprendizagem empírica.

Investigations about temporal reasoning and learning in connectionist models

ABSTRACT

Computational Intelligence is considered, by different authors in present days, the *manifest destiny* of Computer Science. The modelling of different aspects of cognition, such as learning and reasoning, has been a motivation for the integrated development of the symbolic and connectionist paradigms of artificial intelligence. More recently, such integration has led to the construction of models catering for integrated learning and reasoning. The integration of a temporal dimension into such systems is a relevant task as it allows for a richer representation of cognitive behaviour features, since time is considered an essential component in intelligent systems development.

This work introduces SCTL (Sequential Connectionist Temporal Logic), a neural-symbolic approach for integrating temporal knowledge, represented as logic programs, into recurrent neural networks. This integration is done in such a way that the semantic characterization of both representations are equivalent. Besides the strategy to achieve translation from one representation to another, and verification of the semantic equivalence, we also compare the proposed approach to other systems that perform symbolic and temporal representation in neural networks.

Moreover, we describe the intended behaviour of the generated neural networks, for both temporal inference and learning through an algorithmic approach. Such behaviour is then evaluated by means several experiments, in order to analyse the performance of the model in cognitive modelling under different conditions and applications.

Keywords: Neural-symbolic integration, temporal logic, empirical learning.

1 INTRODUÇÃO

A inteligência artificial pode ser considerada uma área de pesquisa com o objetivo de modelar, em sistemas computacionais, capacidades como racionalidade e cognição (RUSSELL; NORVIG, 2003). De acordo com os mesmos autores, podemos descrever racionalidade como a capacidade de decidir pela opção logicamente correta, ou por uma opção ótima, no caso de uma definição mais branda, e podemos definir cognição como sendo a faculdade de processamento de informações intrínseca ao ser humano. Feigenbaum (2003) considera a inteligência artificial como a principal meta da computação para o futuro, bem como os métodos e conceitos da computação como fundamentais para a compreensão da natureza da inteligência.

A consolidação de sistemas inteligentes que apresentam grande capacidade de inferência simbólica (sistemas especialistas e baseados em conhecimento) é impedida, muitas vezes, pela dificuldade de formação da base de conhecimentos necessárias para a inferência, problema denominado *knowledge acquisition bottleneck*. Como estratégia para contornar este problema, a aprendizagem de máquina, mais especificamente a aprendizagem por exemplos (empírica), constitui uma área essencial dentro da inteligência artificial (MICHALSKI, 1987). Valiant (2003) descreve, como um dos maiores problemas da computação na atualidade, a caracterização de uma semântica consistente com dois dos aspectos considerados principais com relação ao comportamento cognitivo: a capacidade de aprender e de raciocinar sobre aquilo que foi aprendido.

Duas abordagens principais podem ser consideradas na modelagem do comportamento cognitivo. Newell e Simon (1976), na chamada 'Hipótese dos Sistemas de Símbolos Físicos' propõem que um sistema simbólico apresenta os meios necessários e suficientes para a implementação do comportamento inteligente. Com base nesta premissa, a chamada IA simbólica se baseia em representação explícita do conhecimento, e manipulação deste, através de recursos fortemente baseados em lógica, estabelecendo forte relação entre os conceitos de cognição e racionalidade. Por outro lado, a abordagem conexionista trata a cognição como um fenômeno emergente da estrutura neuronal do cérebro humano, e tenta representar o comportamento cognitivo através de redes massivamente conectadas de unidades simples de processamento. Smolensky (1988) argumenta que os processos mentais formam um sistema complexo que não pode ser expresso formal e completamente em um nível simbólico.

Entre estes dois extremos, existem diversos modelos que buscam unificar algumas características de ambas abordagens com o intuito de formar um modelo cognitivo mais robusto. Estes sistemas, chamados neuro-simbólicos, buscam superar as limitações que cada paradigma apresenta quando utilizado individualmente, através da integração das principais capacidades de cada modelo individual, como a capacidade de inferência lógica dos sistemas simbólicos e a natural propensão das redes neurais artificiais para realizar

aprendizagem (HILARIO, 1995).

Questões como a inspiração biológica e a natureza paralelamente distribuída e propícia para a aprendizagem empírica, características de redes neurais artificiais, estimularam o desenvolvimento de técnicas para representação de conhecimento simbólico em ambientes conexionistas (BROWNE; SUN, 2001). Um dos grandes obstáculos para esta representação consiste na chamada 'natureza proposicional' de sistemas conexionistas (HITZLER; HÖLLDOBLER; SEDA, 2004), e conseqüente dificuldade para modelagem de linguagens simbólicas mais expressivas. Uma abordagem considerada para contornar este obstáculo consiste na utilização de representação de conhecimento baseada em lógicas não clássicas, tais como lógicas modais, temporais e epistêmicas (D'AVILA GARCEZ; LAMB, 2005).

Dentre as diferentes lógicas não clássicas, podemos destacar a importância das lógicas temporais. A dimensão temporal pode ser considerada como aspecto fundamental para diferentes aspectos cognitivos como aprendizagem (ELMAN, 1990), representação de conhecimento (ALLEN, 1983) e raciocínio inferencial (BARRINGER et al., 1995). Desta forma, o uso de lógicas temporais provê um framework simbólico, formal e não ambíguo para representação do tempo e de seus efeitos sobre o comportamento de sistemas (GABBAY; HODKINSON; REYNOLDS, 1994).

Este trabalho busca integrar o poder de representação das lógicas temporais em um ambiente neuro simbólico, focando em diferentes aspectos desta integração. Sob o ponto de vista simbólico, uma linguagem de programação em lógica para representação de conhecimento temporal é definida, de forma a obter expressividade para sua aplicação em diferentes domínios (modelagem de sistemas dinâmicos, raciocínio temporal em agentes, etc) e ao mesmo tempo garantir a computabilidade em uma estrutura conexionista. Considerando pelo prisma da IA conexionista, diferentes modelos temporais de redes neurais são analisados para a seleção de uma arquitetura que se adeqúe à representação da linguagem lógica selecionada e a aplicação de rotinas de aprendizagem empírica.

Ambos aspectos são, então, integrados em um sistema único, chamado SCTL (Sequential Connectionist Temporal Logic - Lógica Temporal Conexionista e Seqüencial). Tal sistema se baseia na tradução do conhecimento expresso simbolicamente na forma de um programa em lógica temporal para uma arquitetura de rede neural capaz de computar corretamente a semântica deste programa. Esta tradução acontece nos mesmos moldes do modelo CILP (Connectionist Inductive Learning and logic Programming), proposto por D'Avila Garcez e Zaverucha (1999). Além disto, o comportamento da rede gerada pela tradução é definido de forma a permitir a uso de algoritmos de treinamento, buscando o aprimoramento do conhecimento da rede em função da aplicação de exemplos (aprendizagem supervisionada). No decorrer do texto, por abuso de notação, chamaremos a linguagem simbólica definida de "Lógica SCTL", e os modelos neurais gerados de "Redes SCTL".

Nos primeiros capítulos, uma análise preliminar, necessária para a compreensão do trabalho, é realizado. O capítulo 2 reúne informações sobre a lógica na ciência da computação, ilustrando diferentes métodos para representação temporal e mostrando sistemas de programação em lógica. Além disto, neste capítulo é definida a sintaxe e a semântica da lógica SCTL. O capítulo 3 apresenta uma revisão sobre modelos tradicionais e modelos recorrentes de redes neurais, ilustrando suas vantagens e a aplicabilidade de rotinas de aprendizagem. O capítulo 4 apresenta uma revisão sobre as diferentes abordagens para realizar a integração neuro simbólica e o capítulo 5 apresenta o modelo neuro-simbólico (D'AVILA GARCEZ; ZAVERUCHA, 1999), que serve como base para nosso trabalho,

bem como diferentes extensões não-clássicas para este modelo.

No capítulo 6, diferentes aspectos com relação à representação de conhecimento temporal e ao modelo CILP são discutidos, e um algoritmo para tradução de programas lógicos temporais para arquiteturas de redes neurais é desenvolvido. Além disto, a correção semântica desta rede gerada pelo algoritmo, com relação ao programa original, é provada. No capítulo 7, uma estratégia de aprendizagem empírica adaptada a este modelo é descrita, e diversos exemplos são mostrados para ilustrar a importância das etapas de representação e aprendizagem de conhecimento, bem como a integração entre estas. Por fim, o capítulo 8 apresenta as principais conclusões do trabalho, e diferentes rumos a serem seguidos para trabalhos futuros. Parte do conteúdo deste trabalho encontra-se descrito em artigos apresentados em conferências internacionais (BORGES; LAMB; D'AVILA GARCEZ, 2006, 2007).

2 LÓGICAS

2.1 Lógicas para a Ciência da Computação

Diversos autores (HALPERN et al., 2001; ARKOUDAS, 2004) consideram que a lógica desempenha um papel fundamental para diversas áreas da ciência da computação, comparável com o papel desempenhado pelo cálculo nas ciências físicas e disciplinas tradicionais de engenharia. Da mesma forma que o cálculo e equações diferenciais podem ser usados para modelar o comportamento de sistemas físicos contínuos, a linguagem da lógica matemática pode ser usada como uma notação não ambígua para a especificar a estrutura e o comportamento discreto de sistemas. Dentre as diversas áreas da computação em que as lógicas são aplicáveis, podemos citar desenvolvimento de hardware, engenharia de software, linguagens de programação, teoria da computação, inteligência artificial entre outras.

Huth e Ryan (2000) definem, como principal objetivo da lógica na Ciência da Computação, o desenvolvimento de linguagens para modelar diferentes situações, de forma que se possa aplicar um raciocínio formal sobre elas. Dentro da atual visão de sociedade de informação, onde o conhecimento é um bem essencial e crescentemente incorporado em produtos e serviços, a existência de ferramentas formais para sua modelagem e manipulação são de grande importância.

Lloyd (2003) descreve que as lógicas utilizadas pela Ciência da Computação apresentam dois componentes principais: sintaxe e semântica. A sintaxe diz respeito a quais expressões são bem formadas, a quais fórmulas (expressões de tipo booleano) são teoremas e à prova destes teoremas. Já a semântica diz respeito ao significado dos símbolos que compõem os termos e dos termos propriamente ditos, às interpretações que dão este sentido, aos modelos (interpretações que garantem que todos os axiomas sejam verdadeiros) e a quais fórmulas são válidas (verdadeiras em qualquer situação).

Dentre os sistemas lógicos utilizados na ciência da computação, as lógicas modais apresentam grande destaque pela sua capacidade de representação de estruturas relacionais. Blackburn et al. (2001) descrevem as linguagens modais como linguagens simples porém expressivas, que provêem uma perspectiva local, interna para representação das estruturas relacionais.

Huth e Ryan (2000) definem lógicas modais como sistemas que adicionam às lógicas convencionais operadores para expressar um ou mais modos de verdade. Os operadores modais mais comuns são $\Diamond$ (também chamado de possibilidade) e $\Box$ (também chamado de necessidade). Dependendo da natureza da informação a ser representada, estes operadores podem apresentar significados diferentes, como descrito na tabela 2.1.

Tabela 2.1: Diferentes significados dos operadores modais

Lógica	$\Box\varphi$	$\Diamond\varphi$
Temporal	φ sempre é verdadeiro	φ é verdadeiro às vezes
Epistêmica	O agente sabe φ	O agente acredita que φ é possível
Alética	φ é necessariamente verdadeiro	φ é possivelmente verdadeiro
Deôntica	φ é obrigatório	φ é permitido

2.2 Lógica Temporal

De acordo com diversos autores, (PNUELI, 1977; ALLEN, 1983), a representação de tempo pode ser considerada como um dos problemas mais cruciais em sistemas que envolvem representação de conhecimento. Gabbay et al. (1994) descreve a representação e manipulação de conhecimento temporal como essencial para diversas áreas da ciência da computação, citando como exemplos bancos de dados, especificação de software, desenvolvimento de hardware, processamento de linguagem natural, sistemas distribuídos, sistemas em tempo-real entre outros.

O termo lógica temporal tem sido utilizado para cobrir todas as diferentes formas de representar informações temporais dentro de um framework lógico, e mais especificamente para se referir ao tratamento modal introduzido por Arthur Prior com o nome de *Tense Logic*. Esta lógica propõe uma abordagem para relacionar tempo com operadores modais.

Outra proposta de sintaxe para representação lógica do tempo, alternativa a aplicação de operadores modais, é a utilização de lógicas de primeira ordem, através do uso de predicados aumentados com argumentos que indicam a informação temporal associada este predicado. Neste caso, para a representação do fluxo de tempo, torna-se necessário a inserção de predicados e axiomas que modelem suas propriedades.

Dentre as principais diferenças entre as duas abordagens, que são de grande importância para a escolha de uma linguagem para representação temporal, podemos citar a representação do fluxo de tempo, que na abordagem modal se encontra implícita na estrutura relacional entre os mundos possíveis (estados no tempo), enquanto na abordagem baseada em lógica de predicados este fluxo tem que ser descrito explicitamente sob a forma de axiomas.

Galton (2003), define outra diferença importante entre estas abordagens, que diz respeito à distinção entre as chamadas *A-series* e *B-series*. Enquanto as *A-series* caracterizam o tempo em conceitos como presente passado e futuro, as *B-series* não requerem a definição de um ponto de referência como momento atual, apenas aponta as relações "mais cedo" ou "mais tarde" entre os diferentes momentos na linha do tempo. Graças a sua visão local da estrutura relacional, a abordagem modal apresenta uma maior afinidade com a representação de tempo baseada em *A-series*, enquanto a abordagem utilizando predicados apresenta características mais adequadas para representação baseada em *B-series*.

Além da escolha entre lógicas modais e lógicas de primeira ordem, existem outros aspectos que podem receber abordagens diferentes dependendo do sistema lógico a ser utilizado. Um aspecto importante a ser avaliado é a forma de representar os conceitos de fluxo de tempo e mudanças no ambiente (GABBAY; HODKINSON; REYNOLDS, 1994). Um dos tratamentos possíveis é considerar os eventos e mudanças do sistema como conceitos primários na representação. Desta forma, o conceito de fluxo de tempo é definido em função destes conceitos. Seguindo esta linha, diferentes formas podem ser

utilizadas para representar o tempo, como por exemplo definir momentos de tempo como classes de equivalências de estados, indicando que o sistema se mantém num determinado ponto do fluxo de tempo enquanto não ocorre nenhuma alteração em seus estados.

Por outro lado, seguindo a idéia de representação do fluxo de tempo como conceito primário, podemos considerar duas diferentes formas de tratamento do tempo: sob a forma de pontos ou de intervalos. Os principais argumentos para utilização de uma representação intervalar se baseiam no fato que ações e eventos no mundo real apresentam certa duração, não podendo ser definidos meramente sob a forma de instantes.

Ainda assim, um intervalo pode ser definido pelo conjunto dos diversos pontos (instantes) que estejam nele contido, seja um número finito em uma representação discreta ou infinito para uma representação contínua, e este conjunto pode ser descrito através dos pontos que marcam o início e o final do intervalo.

Porém, algumas diferenças precisam ser consideradas no caso de utilizar representação intervalar. Uma delas consiste no aumento do número de relações temporais existentes. Enquanto o tratamento de pontos consideraria como possíveis relações apenas as idéias de "mais cedo", "mais tarde" ou "ao mesmo tempo", a representação de intervalos requer um número maior de relações possíveis. Allen (1983) apresenta 13 relações diferentes entre dois intervalos, sendo sete delas ilustradas na figura 2.1 e as outras seis são definidas pelo inverso das primeiras.

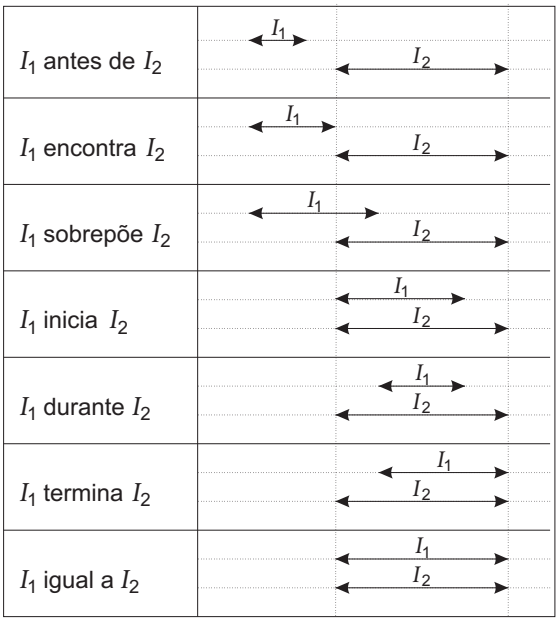

Figura 2.1: Relações entre intervalos.

O conceito de homogeneidade é outro aspecto que deve ser levado em consideração para representações intervalares. Tal conceito indica que, se um fato é verdadeiro em um intervalo, então ele é verdadeiro em qualquer subconjunto deste intervalo. Para estados do mundo, podemos considerar que a homogeneidade é válida, e desta forma tratar isoladamente cada ponto ou subconjunto de pontos dentro de um intervalo. Já no caso de um evento, definido em função de um intervalo como todo, a avaliação de apenas parte do intervalo pode causar uma descaracterização do evento. Se, por exemplo, considerarmos um evento "João viajou de Porto Alegre a Londres", este evento está caracterizado apenas se considerarmos o intervalo desde o momento que ele saiu de sua origem até o ponto no tempo em que ele atingiu o destino. Se avaliarmos apenas parte deste intervalo,

não estará caracterizada uma viagem de Porto Alegre a Londres, mas sim entre pontos intermediários na viagem.

Voltando a considerar uma abordagem modal, para representação baseada no fluxo de tempo como conceito primário, e no uso de pontos em vez de intervalos, utilizaremos uma série de operadores lógicos que contemplam, além dos operadores tradicionais da lógica proposicional, dois operadores unários chamados operadores temporais fracos, e dois chamados operadores temporais fortes. Os operadores temporais fracos $\blacklozenge$ e $\lozenge$ indicam a existência de ao menos um momento em que o operando seja verdadeiro, respectivamente, no passado e no futuro. Já os operadores fortes $\blacksquare$ e $\square$ indicam que para todos os momentos (respectivamente no passado e no futuro) o operando é necessariamente verdadeiro.

Outros operadores também são bastante utilizados para representação de tempo numa abordagem modal, entre eles os operadores binários $\mathbb{S}$ (Since, que indica que um fato vem sendo verdade desde a ocorrência de outro) e $\mathbb{U}$ (Until, que indica que um fato será verdadeiro até a ocorrência de outro). Os operadores $\mathbb{Z}$ (Zince) e $\mathbb{W}$ (Unless) constituem versões fracas destes operadores, ou seja, permitindo também o caso em que o antecedente ocorre "sempre no passado" ($\alpha\mathbb{Z}\beta \equiv \alpha\mathbb{S}\beta \vee \blacksquare\alpha$) ou "sempre no futuro" ($\alpha\mathbb{W}\beta \equiv \alpha\mathbb{U}\beta \vee \square\alpha$). Além destes, em sistemas que tratam o tempo de forma discreta, podemos citar os operadores de próximo momento($\bigcirc$) e momento anterior($\bullet$), que fazem referência direta aos pontos no fluxo de tempo imediatamente posterior ou anterior ao momento atual.

Para a escolha de uma ferramenta lógica para modelar aspectos temporais, um outro aspecto importante de ser observado diz respeito ao determinismo (ou não-determinismo) do fluxo do tempo. Dentro de uma visão determinística, o fluxo de tempo pode ser representado como uma seqüência linear de eventos, enquanto a abordagem não determinística requer uma representação de tempo com ramificações representando as diferentes possibilidades para o futuro.

Enquanto os operadores modais mencionados anteriormente apresentam uma interpretação natural para um fluxo de tempo linear, estes operadores vão apresentar uma semântica diferente quando houver ramificações no tempo. Lamport (1980) cita como exemplo o caso do operador "às vezes" ($\lozenge$), que na abordagem linear seria semanticamente equivalente à definição de "não nunca" ($\neg\square\neg$), enquanto na representação ramificada do tempo estas definições poderiam apresentar significados diferentes, tendo em vista que deveriam considerar também em quais ramificações de tempo estão sendo aplicadas.

Para definição de uma lógica para tratar ramificações no tempo, Ben-Ari et al. (1981) propõem adicionar aos operadores $\square$, $\lozenge$ e $\bigcirc$, um quantificador (universal ou existencial), para indicar se o operador é válido para todos as diferentes possibilidades de futuro ou para ao menos uma. A tabela 2.2 apresenta a descrição semântica dos operadores propostos por Ben-Ari, para ramificações no tempo começando no presente.

Da mesma forma que as ramificações de tempo podem ser consideradas para o futuro, pode se imaginar sua utilização para representação de diferentes possibilidades de passado. Kupferman (1995), por exemplo, descreve esta possibilidade, indicando a importância desta inclusão para enriquecimento da linguagem buscando maior correspondência com descrições naturais do mundo.

2.3 Programação em Lógica

Dentro do paradigma simbólico da Inteligência Artificial, um dos grandes esforços registrados na bibliografia consiste no desenvolvimento de sistema de programação de alto nível, capazes de interpretar conhecimento simbólico expresso de forma descritiva

Tabela 2.2: Operadores para tratar ramificação no tempo

Operador	Semântica
$A\Box P$	Indica que P vai ser sempre verdadeiro em todas as ramificações de tempo
$A\Diamond P$	Indica que P vai ser verdadeiro em pelo menos um momento de todas as diferentes ramificações no tempo
$A \bigcirc P$	Indica que P vai ser todos os momentos possivelmente posteriores ao presente
$E\Box P$	Indica que o fato P vai ser verdadeiro em todos os momentos (sempre) de ao menos uma das ramificações no tempo
$E\Diamond P$	Indica que o fato P vai ser verdadeiro em ao menos um momento de uma ramificação no tempo
$E \bigcirc P$	Indica que o fato P vai ser verdadeiro em pelo menos um dos possíveis momentos imediatamente posterior ao presente

em uma linguagem formal. McCarthy (1959) foi um dos pioneiros a propor o uso de fragmentos da lógica de primeira ordem como linguagem básica para tais sistemas.

A programação em lógica pode ser vista como união de dois conceitos principais: Uma linguagem lógica para a descrição do conhecimento e um mecanismo de execução para interpretar os programas escritos nesta linguagem. A sintaxe usada na programação em lógica é baseada na utilização de cláusulas de Horn:

Definição 2.1 *Um átomo é uma fórmula elementar, que não pode ser dividida em subfórmulas. Um literal pode ser definido como sendo um átomo ou a negação de um átomo. Uma cláusula consiste em uma disjunção de n literais,* $n \geq 1$.

Em uma lógica de primeira ordem, um átomo pode ser considerado uma igualdade entre termos ($t_1 = t_2$) ou uma expressão $p(t_1, t_2..., t_n)$, onde p é um predicado de aridade n e $t_1...t_n$ são termos da lógica ($n \geq 0$). Um termo pode ser uma variável, uma constante ou uma expressão $f(t_1, ..., t_n)$ onde f é uma função de aridade n e $t_1...t_n$ são termos. Já em uma lógica proposicional, um átomo pode ser definido como uma variável proposicional. No decorrer do trabalho, exceto se definido em contrário, consideraremos as definições lógicas sob a ótica proposicional.

Definição 2.2 *Uma cláusula de Horn pode ser definida como uma expressão lógica na forma* $A \leftarrow L_1, L_2, ...L_n$*, onde A é um átomo chamado de cabeça da cláusula e* L_i $(1 \leq i \leq n)$ *são literais que constituem o corpo. Uma cláusula é satisfeita se, e somente se, sempre que a conjunção da interpretação dos literais em seu corpo for verdadeira, o átomo em sua cabeça seja também interpretado como verdadeiro.*

Definição 2.3 *Na lógica proposicional, uma interpretação* I_n *é um mapeamento dos átomos de um programa para valores verdade. Um modelo de um programa em lógica é uma interpretação em que todas as cláusulas do programa são satisfeitas.*

Algumas propriedades de programas lógicos são importantes para o estudo de sua semântica realizado mais adiante neste trabalho:

Definição 2.4 *Um mapeamento de nível | | é definido como todo mapeamento dos literais de um programa lógico para o conjunto dos números naturais tais que* $|\sim A| = |A|$. *Para um átomo A,* $|A|$ *é chamado o nível de A.*

Definição 2.5 *Um programa lógico* $\mathcal{P}$ *é considerado aceitável com relação a um mapeamento de nível | | e um modelo M se, e somente se, para toda cláusula* $A \leftarrow L_1, L_2, ...L_k$ *em* $\mathcal{P}$, *a seguinte implicação é valida para* $1 \leq i \leq k$: *se* $M \vDash \bigwedge_{j=1}^{i-1} L_j$ *então* $|A| > |L_i|$.

Definição 2.6 *Um programa lógico é considerado acíclico com relação a um mapeamento de nível se, e somente se, para todas as cláusulas* $A \leftarrow L_1, ..., Ln$ *do programa, temos que* $|A| > |L_i|$.

Um programa lógico é considerado aceitável se e somente se existir um mapeamento de nível | | e um modelo M tais que $\mathcal{P}$ é aceitável com relação a | | e M. Da mesma forma, $\mathcal{P}$ é considerável acíclico se, e somente se, existir um mapeamento de nível | | tal que $\mathcal{P}$ seja acíclico com relação a | |.

Proposição 2.7 *Todo programa acíclico é também um programa aceitável.*

A semântica denotacional de um programa em lógica clássica pode ser definida através de seu modelo mínimo de Herbrand. Um modelo M é considerado mínimo se, e somente se, não existe nenhum modelo M' diferente de M tal que $M' \subset M$. Por outro lado, podemos definir a semântica de programas aceitáveis de maneira procedimental, através do cálculo do ponto fixo do operador de conseqüência imediata ($\mathcal{T}_\mathcal{P}$).

Definição 2.8 *O operador de consequência imediata* $\mathcal{T}_\mathcal{P}$ *de um programa proposicional* $\mathcal{P}$ *é uma transformação sobre interpretações de* $\mathcal{P}$. *A aplicação de* $\mathcal{T}_\mathcal{P}$ *sobre uma interpretação* $I_\mathcal{P}$ *resulta em uma nova interpretação atribuindo verdadeiro a todos os átomos A que são cabeça de uma cláusula na forma* $A \leftarrow L_1, L_2, ..., L_n$ *tal que* $I_\mathcal{P}(L_1) \wedge I_\mathcal{P}(L_2) \wedge ... \wedge I_\mathcal{P}(L_n)$ *seja verdadeira.*

No livro de D'Avila Garcez, Broda e Gabbay (2002) encontramos uma explicação mais detalhada desta ligação entre a semântica declarativa e a semântica de ponto fixo de programas lógicos proposicionais aceitáveis. Nosso trabalho, por motivos explicados mais adiante, tem um enfoque voltado a utilização de programas acíclicos.

Teorema 2.9 *Para todo programa acíclico* $\mathcal{P}$, *aplicações sucessivas do operador de conseqüência imediata* $\mathcal{T}_\mathcal{P}$ *convergem para um único ponto fixo* $\mathcal{F}_\mathcal{P}$, *tal que* $\mathcal{F}_\mathcal{P}$ *é um modelo mínimo de Herbrand de* $\mathcal{P}$.

Prova: A convergência do operador $\mathcal{T}_\mathcal{P}$ pode ser verificada indutivamente em função da convergência individual da interpretação dos átomos do programa. Para isso, tomamos como base da indução que, para todo átomo que não aparece como cabeça em nenhuma cláusula, bem como para todo fato, aplicações sucessivas de $\mathcal{T}_\mathcal{P}$ não alteram sua interpretação.

Como passo de indução, consideremos para cada átomo A uma constante ν_A tal que, a partir de $\nu_A - 1$ execuções de $\mathcal{T}_\mathcal{P}$, para toda cláusula $A \leftarrow L_1, L_2, ..., L_n$ em $\mathcal{P}$, obtenhamos sempre $\mathcal{T}_\mathcal{P}(I_\mathcal{P})(L_i) = I_\mathcal{P}(L_i)$ para todo $1 \leq i \leq n$. Desta forma, podemos verificar que $\mathcal{T}_\mathcal{P}(I_\mathcal{P})(A) = I_\mathcal{P}(A)$ para toda execução de $\mathcal{T}_\mathcal{P}$ a partir de ν_A.

Como o programa é acíclico, temos que todas as cadeias de dependência (seqüências de átomos A_1, A_2, ..., A_n tais que A_{i+1} aparece no corpo de alguma cláusula em que A_i é cabeça) são finitas, logo o valor de ν_A para cada átomo também é finito.

Além disso, pela definição, temos que se uma interpretação $I_{\mathcal{P}}$ é um modelo de $\mathcal{P}$, então todas as cláusulas de $\mathcal{P}$ são satisfeitas para $I_{\mathcal{P}}$, logo $\mathcal{T}_{\mathcal{P}}(I_{\mathcal{P}})(A) = (I_{\mathcal{P}})(A)$ para todo átomo $A \in \mathcal{P}$. Como verificamos que o ponto fixo de $\mathcal{T}_{\mathcal{P}}$ é único, temos que $\mathcal{F}_{\mathcal{P}}$ é o único modelo de $\mathcal{P}$ e, portanto, um modelo mínimo de Herbrand.

2.4 Lógica SCTL

Mesmo aplicável a uma série de diferentes domínios, a programação em lógica apresenta várias limitações. Buscando contornar estas limitações, diferentes recursos não lógicos foram adicionados aos sistemas de programação em lógica, acarretando em um prejuízo no formalismo destes programas estendidos (ORGUN; MA, 1994). Como uma forma de resolver estas limitações, diferentes estratégias para utilização de lógicas não clássicas foram propostas, visando o aumento da expressividade das linguagens de programação em lógica sem perder o formalismo de sua estrutura. Dentre estes sistemas de programação em lógica não clássica, encontramos no trabalho de Orgun e Ma (1994) uma visão geral de algumas das mais importantes implementações.

Tomando como referência diferentes sistemas baseados em lógica temporal (BARRINGER et al., 1995; FISCHER; GABBAY; VILA, 2005), propomos uma extensão da programação em lógica temporal baseada na introdução de operadores temporais passados ($\bullet$, $\blacksquare$, $\blacklozenge$, $\mathbb{S}$ e $\mathbb{Z}$) e futuros ($\circ$, $\Box$, $\Diamond$, $\mathbb{U}$ e $\mathbb{W}$). A sintaxe para tais programas lógicos pode ser considerada em função de uma nova definição de átomo:

Definição 2.10 *Uma expressão α é definida como um átomo (temporal) se, e somente se:*

- *$\alpha = A$, onde A é uma variável proposicional;*
- *$\alpha = \bullet\beta$, $\alpha = \blacksquare\beta$, $\alpha = \blacklozenge\beta$, $\alpha = \beta\mathbb{S}\gamma$ ou $\alpha = \beta\mathbb{Z}\gamma$; onde β e γ são átomos;*
- *$\alpha = \circ\beta$, $\alpha = \Box\beta$, $\alpha = \Diamond\beta$, $\alpha = \beta\mathbb{U}\gamma$ ou $\alpha = \beta\mathbb{W}\gamma$; onde β e γ são átomos;*

Os conceitos de literal, cláusula e programa são definidos como usual, em função do conceito temporal de átomo. Para a semântica desta linguagem, definimos uma abordagem de ponto fixo, que se baseia num tratamento seqüencial do fluxo de tempo. Este tratamento considera que a inferência sobre o presente só é realizada depois que o conhecimento sobre o passado estiver estabelecido, ou seja, a interpretação associada a um átomo α num ponto t do fluxo de tempo é calculada após a definição da interpretação de todos os átomos do programa em momentos $t' < t$. Esta utilização toma como referência o trabalho de Barringer et al. (1995), onde lógicas temporais são utilizadas para definição imperativa de um sistema executável, baseando-se no uso de relações de conseqüência que consideram um antecedente sobre o passado para realizar inferência sobre o futuro. Os autores argumentam que tal abordagem, pela direta relação com o fluxo de execução do programa, apresenta grande representatividade, como por exemplo para modelagem de sistemas reativos.

Em nosso trabalho, consideramos uma visão não estrita para os conceitos de passado e futuro, ou seja, o presente é considerado como ambos. Desta forma, os operadores utilizados são definidos da seguinte maneira:

- $\bullet\alpha$ é verdadeiro em t sss α é verdadeiro em $t-1$
- $\blacksquare\alpha$ é verdadeiro em t sss α é verdadeiro em todos os momentos $t' \leq t$
- $\blacklozenge\alpha$ é verdadeiro em t sss α é verdadeiro em algum momento $t' \leq t$
- $\alpha\mathbb{S}\beta$ é verdadeiro em t sss β é verdadeiro em algum momento $t' \leq t$, e α é verdadeiro em todo momento u tal que $t' < u \leq t$
- $\alpha\mathbb{Z}\beta$ é verdadeiro em t sss $\alpha\mathbb{S}\beta$ ou $\blacksquare\alpha$ forem verdadeiros
- $\bigcirc\alpha$ é verdadeiro em t sss α é verdadeiro em $t+1$
- $\Box\alpha$ é verdadeiro em t sss α é verdadeiro em todos os momentos $t' \geq t$
- $\Diamond\alpha$ é verdadeiro em t sss α é verdadeiro em algum momento $t' \geq t$
- $\alpha\mathbb{U}\beta$ é verdadeiro em t sss β é verdadeiro em algum momento $t' \geq t$, e α é verdadeiro em todo momento u tal que $t < u \leq t'$
- $\alpha\mathbb{W}\beta$ é verdadeiro em t sss $\alpha\mathbb{U}\beta$ ou $\blacklozenge\alpha$ forem verdadeiros

Para a definição semântica dos programas temporais, vamos considerar, primeiramente, uma semântica de ponto fixo simplificada, atribuindo significado apenas ao operador $\bullet$. Definimos, então, um operador de conseqüência imediata chamado $\bullet\mathcal{T}_\mathcal{P}$, cuja importância como estrutura auxiliar em nosso sistema será descrita mais adiante. A definição deste operador em um momento t é dada em função do ponto fixo calculado para o momento imediatamente anterior $t-1$ ($\mathcal{F}_\mathcal{P}^{t-1}$). Para um fluxo de tempo começando em $t = 1$, consideramos um ponto no tempo virtual ($t = 0$) tal que $\mathcal{F}_\mathcal{P}^0$ é definido arbitrariamente como verdadeiro para todo átomo $\blacksquare\alpha$ e $\alpha\mathbb{W}\beta$, e como falso para todos os outros átomos.

Definição 2.11 *O operador de consequência imediata $\bullet\mathcal{T}_\mathcal{P}$ de um programa temporal $\mathcal{P}$ é uma transformação sobre interpretações de $\mathcal{P}$. A aplicação de $\bullet\mathcal{T}_\mathcal{P}$ sobre uma interpretação $I_\mathcal{P}^t$ com relação a um momento t no tempo resulta em uma nova interpretação em t atribuindo verdadeiro a todos os átomos α tais que:*

a. α seja cabeça de uma cláusula na forma $\alpha \leftarrow \lambda_1, \lambda_2, ..., \lambda_n$ e $I_\mathcal{P}^t(\lambda_1 \wedge \lambda_2 \wedge ... \wedge \lambda_n)$ seja verdadeira.

b. α seja um átomo na forma $\bullet\beta$, e $\mathcal{F}_\mathcal{P}^{t-1}(\beta)$ seja verdadeiro.

Lema 2.12 *Para todo programa temporal acíclico $\mathcal{P}$, aplicações sucessivas do operador de conseqüência imediata $\bullet\mathcal{T}_\mathcal{P}$ em um ponto t no tempo convergem para um único ponto fixo $\mathcal{F}_\mathcal{P}^t$.*

Prova: A convergência de $\bullet\mathcal{T}_\mathcal{P}$ para programas acíclicos temporais pode ser verificada da mesma forma que a convergência de $\mathcal{T}_\mathcal{P}$ tradicional para programas acíclicos. Basta considerar que a regra 'a.' da definição de $\bullet\mathcal{T}_\mathcal{P}$ é semelhante à definição de $\mathcal{T}_\mathcal{P}$, e a atribuição de valores pela regra 'b.' é definida diretamente antes da primeira execução, e se mantém constante em todas as execuções de $\bullet\mathcal{T}_\mathcal{P}$, não afetando em sua convergência.□

Para a definição da semântica com relação aos operadores relacionados ao passado, considerando a abordagem seqüencial, utilizaremos uma definição recursiva destes operadores em função dos momentos passado e presente. Para o operador $\blacksquare$, por exemplo, podemos considerar que um átomo na forma $\blacksquare\alpha$ é verdadeiro no momento inicial do fluxo do tempo ($t = 1$) se α for verdadeiro no mesmo momento, e que $\blacksquare\alpha$ é verdadeiro para os outros momentos t se α for verdadeiro em t e $\blacksquare\alpha$ for verdadeiro em $t-1$. Desta forma, definindo um valor arbitrário para os átomos em um momento no tempo virtual $t = 0$, podemos definir um operador de conseqüência imediata baseado nos momentos presente e passado, como descrito nos itens 'b.' a 'f.' na definição 2.13.

Seguindo a representação baseada na seqüência do tempo, diversos autores propõem o uso de uma semântica imperativa para os operadores futuros, representando comprometimentos de um agente e as ações tomadas para realizar tal comprometimento (BARRINGER et al., 1995). Em nosso trabalho, buscamos uma utilização dos operadores futuros também na forma de comprometimentos, porém numa abordagem declarativa. Desta forma, as operadores futuros servem apenas para declaração de informações que devem ser verdade nos próximos momentos, sem definição dos passos necessários para atingir tal objetivo.

De forma similar aos operadores passados, podemos definir os operadores futuros recursivamente, com relação ao próximo momento no tempo. Por exemplo, se um átomo $\Box\alpha$ for verdadeiro em um momento t, considerando uma definição não estrita de futuro, temos que α deve ser verdadeiro no tempo t e $\Box\alpha$ deve ser verdadeiro no tempo $t+1$, e que estas afirmações representam completamente a definição de $\Box$. Os demais operadores podem ser definidos, em função de $\bigcirc$ como:

- $\Diamond\alpha \equiv \alpha \vee \bigcirc\Diamond\alpha$
- $\alpha\mathbb{U}\beta \equiv \beta \vee (\alpha \wedge \bigcirc(\alpha\mathbb{U}\beta)) \equiv (\beta \vee \alpha) \wedge (\beta \vee \bigcirc(\alpha\mathbb{U}\beta))$
- $\alpha\mathbb{W}\beta \equiv \beta \vee (\alpha \wedge \bigcirc(\alpha\mathbb{W}\beta)) \equiv (\beta \vee \alpha) \wedge (\beta \vee \bigcirc(\alpha\mathbb{W}\beta))$

No caso do operador $\Box$, utilizamos uma definição baseada numa conjunção, ou seja, quando $\Box\alpha$ for verdadeiro, ambos α e $\bigcirc\alpha$ podem ser afirmados. Por outro lado, os demais operadores são representados através do uso de disjunções, portanto uma análise mais específica se faz necessária para definir a relação de conseqüência em função da seqüência temporal. Nossa opção para representar tais disjunções consiste em evitar o caso em que ambos os lados sejam falso, lembrando que, na lógica clássica, $(\alpha \vee \beta) \leftrightarrow (\neg\alpha \rightarrow \beta)$ e $(\alpha \vee \beta) \leftrightarrow (\neg\beta \rightarrow \alpha)$.

Considerando sempre a abordagem seqüencial (passado $\rightarrow$ futuro), a representação de $\Diamond\alpha$ fica na forma $\neg\alpha \rightarrow \bigcirc\Diamond\alpha$. Para os demais operadores, a única expressão que não é imediatamente derivado da nossa abordagem é a disjunção $\alpha \vee \beta$, que permite duas representações. Utilizaremos cada representação para caracterizar um operador, de forma que a representação de $\alpha\mathbb{U}\beta$ considera o uso de $\neg\alpha \rightarrow \beta$ e $\alpha\mathbb{W}\beta$ utiliza $\neg\beta \rightarrow \alpha$. Esta escolha é feita porque o operador unless ($\mathbb{W}$) aceita, por definição, seqüências infinitas de α, ou seja, a menos que β seja verdadeiro, α será repetido.

Com base nestas representações, e considerando que o operador $\bigcirc$ pode ser diretamente associado com o operador $\bullet$ ($\bullet\bigcirc\alpha \equiv \alpha$) podemos definir um operador de conseqüência imediata para os operadores futuros, considerando também o momento presente e o imediatamente anterior, como nos itens 'g.' a 'n.' da definição 2.13.

Definição 2.13 *A aplicação do operador de consequência imediata $\mathcal{ST}_\mathcal{P}$ de um programa temporal $\mathcal{P}$ sobre uma interpretação $I^t_\mathcal{P}$ com relação a um momento t no tempo resulta em uma nova interpretação em t atribuindo verdadeiro a todos os átomos α, tal que ao menos uma das condições seguintes sejam verdadeiras:*

a. α *seja cabeça de uma cláusula na forma* $\alpha \leftarrow \lambda_1, \lambda_2, ..., \lambda_n$ *e* $I^t_\mathcal{P}(\lambda_1 \wedge \lambda_2 \wedge ... \wedge \lambda_n)$ *é verdadeiro;*

b. $\alpha = \bullet\beta$*, e* $\mathcal{F}^{t-1}_\mathcal{P}(\beta)$ *é verdadeiro;*

c. $\alpha = \blacksquare\beta$ *e ambos* $\mathcal{F}^{t-1}_\mathcal{P}(\blacksquare\beta)$ *e* $I^t_\mathcal{P}(\beta)$ *são verdadeiros;*

d. $\alpha = \blacklozenge\beta$ *e* $\mathcal{F}^{t-1}_\mathcal{P}(\blacklozenge\beta)$ *ou* $I^t_\mathcal{P}(\beta)$ *são verdadeiros;*

e. $\alpha = \beta\mathbb{S}\gamma$ *e* $I^t_\mathcal{P}(\gamma)$ *é verdadeiro, ou ambos* $\mathcal{F}^{t-1}_\mathcal{P}(\beta\mathbb{S}\gamma)$ *e* $I^t_\mathcal{P}(\beta)$ *são verdadeiros;*

f. $\alpha = \beta\mathbb{Z}\gamma$ *e* $I^t_\mathcal{P}(\gamma)$ *é verdadeiro, ou ambos* $\mathcal{F}^{t-1}_\mathcal{P}(\beta\mathbb{Z}\gamma)$ *e* $I^t_\mathcal{P}(\beta)$ *são verdadeiros;*

g. $\mathcal{F}^{t-1}_\mathcal{P}(\bigcirc\alpha)$ *é verdadeiro;*

h. $I^t_\mathcal{P}(\Box\alpha)$ *é verdadeiro;*

i. $\alpha = \Box\beta$ *e* $\mathcal{F}^{t-1}_\mathcal{P}(\Box\beta)$ *é verdadeiro;*

j. $\alpha = \Diamond\beta$*,* $\mathcal{F}^{t-1}_\mathcal{P}(\Diamond\beta)$ *é verdadeiro e* $\mathcal{F}^{t-1}_\mathcal{P}\beta$ *é falso;*

k. Existe um β *tal que* $I^t_\mathcal{P}(\beta\mathbb{U}\alpha)$ *seja verdadeiro e* $I^t_\mathcal{P}(\beta)$ *seja falso;*

l. $\alpha = \beta\mathbb{U}\gamma$*,* $\mathcal{F}^{t-1}_\mathcal{P}(\beta\mathbb{U}\gamma)$ *é verdadeiro e* $I^t_\mathcal{P}(\gamma)$ *é falso;*

m. Existe um β *tal que* $I^t_\mathcal{P}(\alpha\mathbb{W}\beta)$ *seja verdadeiro e* $I^t_\mathcal{P}(\beta)$ *seja falso;*

n. $\alpha = \beta\mathbb{W}\gamma$*,* $\mathcal{F}^{t-1}_\mathcal{P}(\beta\mathbb{U}\gamma)$ *é verdadeiro e* $I^t_\mathcal{P}(\gamma)$ *é falso;*

Se for verificado que as relações de conseqüência definidas pelos operador $\mathcal{ST}_\mathcal{P}$ não inserem ciclos nas cadeias de dependência entre átomos, o mesmo raciocínio para demonstração da convergência do operador $\mathcal{T}_\mathcal{P}$ tradicional para programas acíclicos pode ser aplicado.

3 SISTEMAS CONEXIONISTAS

3.1 Definições Básicas

Tendo por base o reconhecimento que o cérebro humano processa informações de forma inteiramente diferente de um computador digital convencional, a chamada Inteligência Artificial Conexionista busca inspiração na estrutura paralela e maciçamente conectada de funcionamento do cérebro como abordagem para modelar o comportamento inteligente (KASABOV, 1996). Haykin(1999, p.28) descreve:

> Redes neurais artificiais são processadores maciçamente paralelamente distribuídos, constituídos de unidades de processamento simples (neurônios artificiais), que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Estas redes se assemelham ao cérebro humano em dois aspectos: O primeiro é relacionado ao processo de aquisição do conhecimento, que é realizado a partir do ambiente através de um processo de aprendizagem. O segundo aspecto diz respeito ao armazenamento do conhecimento adquirido, que acontece através de forças de conexão entre neurônios, chamadas também de pesos sinápticos.

Também em trabalho do mesmo autor (1999) podemos encontrar a descrição de uma série de propriedades úteis e capacidades das redes, dentre as quais podemos citar:

- Capacidade de processamento não linear;
- Processamento paralelo para executar um mapeamento de entrada-saída;
- Adaptabilidade, capacidade de generalizar a partir de exemplos apresentados;
- Armazenamento contextual da informação;
- Tolerância a dados ruidosos ou incompletos;
- Uniformidade de análise e projeto para redes semelhantes;

Graças a estas suas propriedades, redes neurais artificiais são hoje utilizadas em diferentes aplicações envolvendo aproximação de funções, predição de séries temporais, classificação e reconhecimento de padrões, identificação e controle de sistemas e outras formas de processamento de dados.

A unidade fundamental para a operação das redes neurais é o neurônio artificial, cujo modelo mais tradicional apresenta um funcionamento que pode ser dividido em três etapas

principais. A primeira etapa é definida pelas sinapses (elos de conexão). Cada sinapse representa a entrada de um neurônio, é caracterizada por um peso ou força própria, de forma que um sinal aplicado na entrada de uma sinapse é multiplicada por este peso. Além dos valores recebidos pelas conexões externas, um outro valor de entrada, chamado bias também é aplicado no neurônio, com o objetivo de deslocar o valor da função de ativação. A segunda etapa consiste numa junção, realizada através do somatório destes sinais de entrada ponderados pelas sinapses e do bias. Por fim, uma função de ativação é aplicada ao resultado desta soma, formando assim o valor de saída do neurônio. A figura 3.1 ilustra esta estrutura descrita.

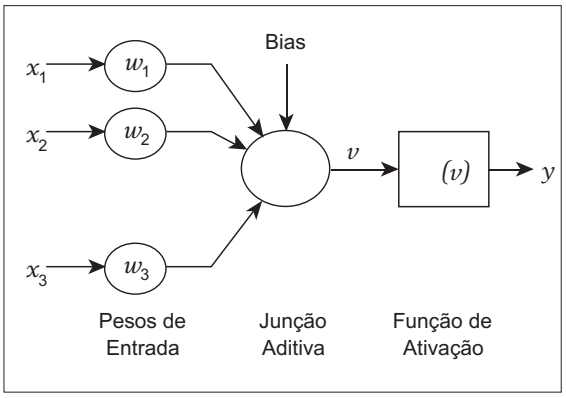

Figura 3.1: Modelo elementar de neurônio (HAYKIN, 1999)

Apesar de ser baseado em uma unidade fundamental bastante simples como o neurônio, as redes neurais apresentam a capacidade de um mapeamento complexo e não linear de entrada para a saída, graças ao uso de funções de ativação não linear, à quantidade de neurônios que as compõem e as estruturas de conectividade entre eles. A estas estruturas damos o nome de arquiteturas de redes, que de acordo com Haykin (1999) podem ser de 3 tipos diferentes: *Feed Forward* com camada única, *Feed Forward* de múltiplas camadas e redes recorrentes.

O modelo mais simples de rede neural utilizável para classificação de padrões é chamado de perceptron, proposto por Rosemblatt (1958), e consiste de um único neurônio com pesos e bias adaptáveis e cuja função de ativação uma função de limiar abrupto, geralmente a função sinal. Desta forma, o neurônio é ativado se (e somente se) o somatório dos valores ponderados de entrada ultrapassam o valor negativo do bias, também chamado *threshold*. Outro modelo tradicional de rede de única camada consiste no Adaline (ADAptive LINEar neuron), cuja principal diferença com relação ao perceptron consiste na função de ativação, em que uma função linear é utilizada em vez do limiar abrupto.

As redes *Feed Forward* são aquelas de arquitetura acíclica, geralmente organizadas na forma de camadas, onde o sinal de saída nos neurônios de uma camada é projetado para a entrada da camada seguinte, mas nunca havendo ligações entre neurônios da mesma camada ou para camadas anteriores. Nas redes *Feed Forward* de camada única, existe apenas um neurônio que faz o mapeamento de um conjunto de entradas para cada saída do modelo. Já nas redes de múltiplas camadas, algumas camadas de neurônios chamadas de ocultas são introduzidas antes da camada de saída, com a intenção de permitir a computação de funções mais complexas. De maneira geral, a camada de entrada de redes de múltiplas camadas é considerada como um conjunto de nós que simplesmente repassam o valor recebido na entrada para os neurônios conectados em sua saída, ou seja, os

neurônios que realmente realizam a junção de valores e aplicação da função de ativação encontram-se nas camadas ocultas e de saída. Na figura 3.2 ilustramos um exemplo rede neural *Feed Forward* com uma única camada oculta.

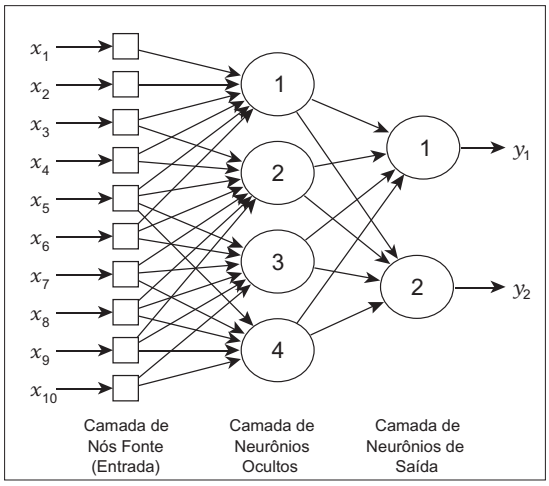

Figura 3.2: Exemplo de rede Feed Forward (HAYKIN, 1999)

Minsky e Papert (1969), em seu artigo histórico, provaram que o perceptron é incapaz de classificar padrões não linearmente separáveis, causando um impacto negativo na pesquisa em IA conexionista da década de 1970. Nos anos 80, diversos autores retomaram as pesquisas em redes neurais, se valendo das capacidades de utilização de arquiteturas multi camadas. Hornik et al. (1989) provaram que redes neurais *Feed Forward* com uma única camada oculta são aproximadores universais, capazes de aproximar qualquer função Borel-mensurável (classe de funções que compreende todas as funções contínuas) em qualquer grau de aproximação desejado, desde que contenha um número suficiente de neurônios na camada oculta.

3.2 Redes Recorrentes e Representação Temporal

Haykin (1999) considera o tempo como um componente essencial para o processo de aprendizagem, uma entidade ordenada que é básica para diversos aspectos da modelagem do comportamento inteligente, independentemente de ser explícita ou implicitamente representado na estrutura. Duas principais formas podem ser consideradas para a representação de aspectos temporais em redes neurais: O uso de unidades de atraso ou de ligações recorrentes entre os neurônios.

Unidades de atraso consistem em mecanismos de memória de curto prazo, que retornam como saída o resultado de uma função aplicada sobre os últimos valores recebidos em sua entrada. O modelo mais simples de unidades de atraso apresenta na saída uma cópia do valor recebido na entrada no último instante de tempo. Nos trabalhos de Waibel et al. (1990) e Clouse et al. (1997) encontramos exemplos de sistemas baseados no uso deste recurso para aprendizagem de séries temporais.

O uso de redes recorrentes se baseia numa representação mais implícita dos aspectos temporais. A idéia consiste, principalmente, na aplicação do valor de ativação de um neurônio M à entrada de uma unidade pertencente à mesma camada ou a uma camada anterior. Desta forma, o resultado do processamento deste neurônio M será usada como

informação de entrada para um novo processamento na rede, mantendo o valor com o passar do tempo.

Dentre as arquiteturas recorrentes mais utilizadas, podemos citar a rede de Elman (1990), um modelo MLP recorrente que apresenta dois tipos de neurônios na primeira camada: Os neurônios de entrada propriamente ditos, que recebem os valores externos para o processamento, e os neurônios de contexto, que recebem, através de uma ligação recorrente, o valor de saída de um neurônio da camada oculta. Para cada neurônio H da camada oculta, existe um neurônio de contexto C que recebe recorrentemente o valor de saída de H. O restante da arquitetura da rede é definida da mesma forma que uma rede *Feed Forward* totalmente conectada, i.e., cada neurônio da camada de entrada é conectado a todos da camada oculta e cada neurônio da camada oculta é conectado a todos da camada de saída.

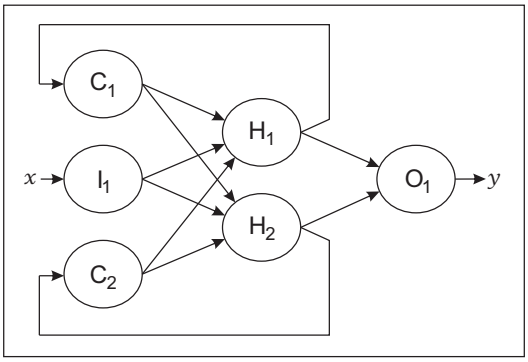

Figura 3.3: Exemplo de rede de Elman

Para representação temporal em nosso trabalho, consideramos o modelo denominado NARX (Nonlinear AutoRegressive with eXogenous inputs) (SIEGELMANN; HORNE; GILES, 1995). Este modelo faz uso de unidades de atraso unitário e ligações recorrentes para realizar o processamento temporal. Nesta arquitetura, cada neurônio de entrada pode receber um valor de entrada no tempo presente ou atrasado através de um banco de unidades de atraso. Além disto, pode receber também o valor de ativação de neurônios de saída, também atrasado por um banco de unidades. O restante da arquitetura permanece semelhante ao tradicional modelo MLP *Feed Forward*. A figura 3.4 ilustra a estrutura de uma rede NARX, onde as caixas marcadas com z^{-1} denotam unidades de atraso.

Siegelmann et al. (1995) provaram que o modelo NARX, mesmo com a limitação de apresentar ligações recorrentes apenas da saída para a entrada, é capaz de simular qualquer outro modelo de rede neural recorrente, independentemente da liberdade de conectividade entre neurônios. Por outro lado, considerando o poder computacional de redes neurais recorrentes, podemos considerá-las equivalentes a Máquinas de Turing universais. Siegelmann e Sontag (1992), por exemplo, provaram que qualquer Máquina de Turing pode ser simulada através do uso de redes neurais recorrentes totalmente conectadas, com neurônios tradicionais baseados na junção aditiva dos valores ponderados de entrada e função de ativação sigmoidal.

3.3 Aprendizagem em redes neurais

Dentre as principais características das redes neurais artificiais, podemos considerar sua propensão natural para a adaptação de sua estrutura em função da apresentação de

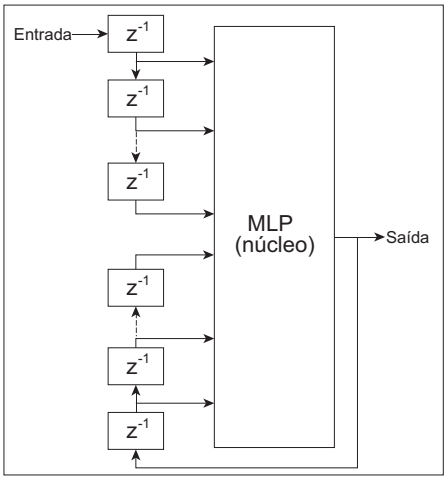

Figura 3.4: Estrutura de rede NARX

exemplos (aprendizagem empírica), através do uso de algoritmos de aprendizagem. Esta aprendizagem ocorre principalmente através da adequação dos pesos sinápticos e bias dos neurônios em função do erro de saída. Diferentes arquiteturas de redes neurais artificiais permitem a aplicação de processos de aprendizagem envolvendo as diferentes abordagens de aprendizagem empírica (KASABOV, 1996):

- Aprendizagem supervisionada: Processo que considera a existência de um sistema supervisor, que informa ao aprendiz a resposta de saída esperada com relação a cada entrada aplicada;
- Aprendizagem não-supervisionada: Processo em que o agente desenvolve um mapeamento de entrada para saída em função das características dos padrões de entrada, sem informações sobre a saída desejada;
- Aprendizagem por reforço: Processo em que, em vez de receber informações sobre a saída esperada, o sistema aprendiz precisa tentar diferentes opções para descobrir quais delas oferecem melhor recompensa.

O algoritmo mais utilizado para realização de aprendizagem supervisionada em redes MLP é chamado Backpropagation, proposto por Rumelhart et al. (1986). Este algoritmo se baseia na idéia do gradiente descendente para a correção dos pesos das conexões de entrada de um neurônio em função do erro de saída. No caso das camadas ocultas, onde não existe informação sobre o erro, o algoritmo se baseia numa estimativa de erro retro-propagada desde a camada de saída (devido a isso o algoritmo leva o nome de Backpropagation).

O algoritmo Backpropagation pode ser descrito da seguinte maneira: Cada neurônio, seja ele oculto ou de saída, ao receber informações sobre o erro (e) calcula um valor de delta (δ). Este valor de delta é retro propagado para a camada anterior, e os pesos da entrada são recalculados, pelo acréscimo de um valor Δw ao peso anterior. Para os neurônios de saída, o valor do erro é definido pela diferença entre o valor desejado de saída (z) e o valor resultante do processamento(y). No caso das camadas ocultas, o valor do erro é definido através da soma dos valores de delta recebidos da camada seguinte, ponderados pelo peso da conexão. A figura 3.5 ilustra um exemplo de aplicação do Backpropagation, indicando os valores usados para correção dos pesos.

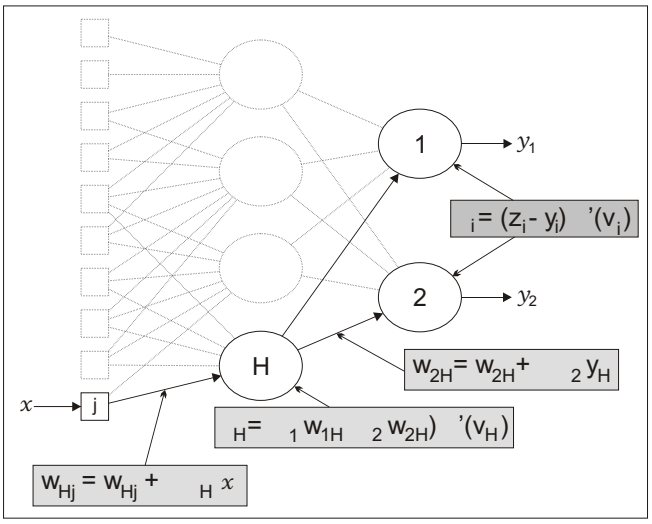

Figura 3.5: Cálculo dos valores retro-propagados

A variável η utilizada para correção do peso consiste na taxa de aprendizagem, que consiste em um valor entre 0 e 1 que indica o impacto da aprendizagem sobre a variação dos pesos. Diferentes estratégias para cálculo e adaptação deste valor são encontrados na literatura (SCHIFFMANN; JOOST; WERNER, 1994), buscando melhorar a convergência e a desempenho do aprendizado do algoritmo Backpropagation. Outra estratégia existente para esta melhoria consiste na utilização uma taxa de *momentum* para definição do acréscimo no peso (ΔW). A taxa de momentum pode ser definida como um percentual sobre o valor da última atualização realizada ao peso.

No caso de redes temporais, diferentes variações do backpropagation podem ser consideradas. Para redes utilizando unidades de atraso, podemos considerar o uso do algoritmo Temporal Backpropagation (WAIBEL et al., 1990), que consiste basicamente no tradicional algoritmo Backpropagation, dando um tratamento especial para o uso de unidades de atraso. Durante a aprendizagem, o valor de δ dos neurônios conectados à saída destas unidades é retropropagado com atraso para cálculo da estimativa de erro na unidade conectada em sua entrada, de forma similar à propagação com atraso realizada na etapa feed forward.

Para redes recorrentes, podemos considerar o uso do algoritmo de aprendizagem BPTT (Backpropagation Through Time), que consiste basicamente em 'desdobrar' uma rede recorrente de forma a gerar uma rede *Feedforward*. A rede resultante é treinada utilizando o algoritmo Backpropagation tradicional, com a restrição de que, para neurônios da rede desdobrada associados o mesmo neurônio da rede original, os pesos de entrada devem permanecer os mesmos. Desta forma, o gradiente para correção destes pesos deve ser obtido em função de todos os gradientes associados na rede desdobrada (WERBOS, 1990).

Para redes de Elman, a aprendizagem é realizada de forma mais simplificada, conforme descrito por Elman (1990). Neste processo, os pesos das redes recorrentes não são atualizados, e a retropropagação do erro é realizada de maneira tradicional, sendo as ligações recorrentes apresentam pesos fixos (não alterados no processo de aprendizagem) e o erro dos neurônios de contexto são retropropagados para os neurônios ocultos com atraso, de forma similar ao descrito no Backpropagation Temporal.

Além da atualização dos pesos, outra forma existente para realizar aprendizagem empírica em redes neurais consiste em adaptar a arquitetura em função da aplicação de exemplos. As principais abordagens consistem nos métodos construtivos, que iniciam com uma arquitetura simples e vão adicionando neurônios e conexões até a a rede atingir um desempenho adequado, e nos métodos de poda, que partem de uma rede com grande número de neurônios e vão simplificando a rede enquanto ela mantém um desempenho de acordo com o esperado.

4 INTEGRAÇÃO NEURO-SIMBÓLICA

4.1 Diferentes abordagens para integração neuro-simbólica

Dentro da Inteligência Artificial, conforme mencionado anteriormente, podemos considerar duas diferentes linhas com objetivo de modelar o comportamento cognitivo. O paradigma simbólico se baseia no uso de símbolos e regras, para representar o conhecimento, e sistemas capazes de manipular estas estruturas, para realizar inferências sobre o conhecimento representado. Diversos autores consideram que o comportamento cognitivo acontece em um nível simbólico, e que um sistema baseado na representação e manipulação de símbolos físicos apresenta os meios necessários e suficientes para modelar tal comportamento (NEWELL; SIMON, 1976).

Podemos considerar, dentre as principais vantagens dos sistemas simbólicos, a clareza e o formalismo da representação do conhecimento. Ainda assim, diversas desvantagens são apontadas no desenvolvimento de sistemas simbólicos, como a dificuldade para obtenção e codificação do conhecimento necessário, e a alegada rigidez para a aplicação de rotinas de formação da base de conhecimentos a partir da experiência (aprendizagem empírica).

Por outro lado, a Inteligência Artificial Conexionista se baseia em uma abordagem diferente para representação do conhecimento. Autores como Smolensky (1988) consideram que os processos mentais ocorrem em um nível sub-conceitual, de forma que o comportamento cognitivo é tratado como um efeito emergente de sistemas complexos e massivamente distribuídos que realizam o processamento em um nível subsimbólico. As redes neurais artificiais, como definido no capítulo anterior, apresentam uma série de vantagens, principalmente com relação à aplicação de técnicas para aprendizagem empírica.

A pesquisa relacionada com redes neurais apresenta, no decorrer de sua história, uma série de argumentações contrarias à sua utilização. O trabalho de Minsky e Papert (MINSKY; PAPERT, 1969), expôs um problema de expressividade de sistemas neurais baseados no perceptron, alegando a incapacidade destes sistemas de reconhecimento de classes não linearmente separáveis, problema este resolvido com a utilização de redes com múltiplas camadas de neurônios. Outros autores, como Fodor e Pylyshyn (1988), argumentam que sistemas conexionistas são inadequados para representação cognitiva, tendo em vista que não consideram a estrutura sintática e semântica de representações mentais. As redes neurais apresentam uma desvantagem considerada crítica para sua aplicabilidade em diversos casos, que é a incompreensibilidade do conhecimento aprendido (ANDREWS; DIEDERICH; TICKLE, 1995). Em um sistema conexionista, o conhecimento adquirido em um processo de aprendizagem fica codificado na forma de pesos em sua arquitetura, tornando-se inacessível para a compreensão de especialistas.

Diferentes autores vem propondo a integração destes paradigmas de forma a unir as

principais características da IA conexionista e simbólica, e suprir as limitações existentes em cada. Por um lado, a integração de redes neurais artificiais em sistemas simbólicos permite a utilização de uma abordagem para aprendizagem empírica que alegadamente apresenta grande desempenho e maior robustez para lidar com sistemas ruidosos. Por outro lado, a pesquisa em integração neuro simbólica mostra que sistemas conexionistas apresentam a possibilidade de criação de métodos para introduzir e expressar de forma compreensível o conhecimento codificado numa arquitetura neural.

Encontramos, na literatura, diferentes formas para classificação de técnicas de integração neuro-simbólica. Hilario (1995) apresenta uma taxonomia baseada na forma que os diferentes paradigmas estão acoplados, e da abordagem computacional para representação do comportamento inteligente. A figura 4.1 ilustra esta classificação. Outra taxonomia similar, porém simplificada, é descrita por Wermter e Sun (2000), onde três classes diferentes são consideradas: arquiteturas neurais unificadas, que consistem na utilização plena de redes neurais para representação do conhecimento simbólico, arquiteturas de transformação, que consistem em sistemas que realizam tradução do conhecimento em ambos os sentidos (simbólico ↔ conexionista), e arquiteturas híbridas modulares, onde os módulos de processamento simbólicos e conexionistas são usados para realização de diferentes tarefas.

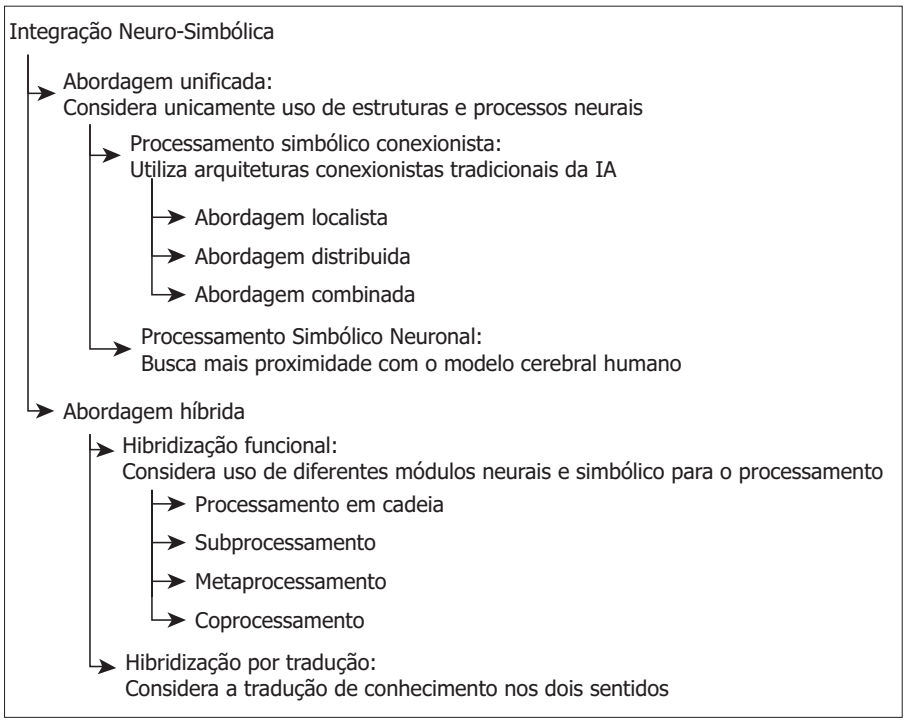

Figura 4.1: Taxonomia de sistemas neuro-simbólicos (HILARIO, 1995)

4.2 Redes neurais realizando inferência

Neste trabalho, focamos mais especificamente em sistemas que utilizam redes neurais para realização de inferência simbólica. Considerando a taxonomia na figura 4.1 tais sistemas podem ser considerados como processamento simbólico conexionista (abordagem unificada) ou como hibridização por tradução (abordagem híbrida). Browne e Sun (2001) enumeram diferentes justificativas para a pesquisa relacionada a esta forma de integra-

ção, como a inspiração biológica no modelo cerebral humano, que é capaz de raciocínio simbólico e algorítmico, e a unificação de teorias e explicações encontradas em ambos paradigmas, além das vantagens já descritas para os sistemas neuro-simbólicos como um todo. Os mesmo autores consideram duas diferentes formas de realizar a representação de conhecimento em arquiteturas conexionistas: representação localista e representação distribuída.

Uma representação localista de conhecimento em redes neurais é aquela que associa a cada símbolo um neurônio (ou conjunto de neurônios) específico da rede. Page (2000) apresenta diferentes definições encontradas na bibliografia para representação de conhecimento localista em redes neurais, ilustrando que em um sistema localista a atividade de unidades individuais pode ser avaliada diretamente para um nível simbólico, sem necessidade de conhecimento da ativação das demais unidades, de forma que a taxa de ativação de uma unidade geralmente define a quantidade de evidência (grau de confiança) existente para o conceito associado a esta unidade no presente contexto.

Em contraponto a estas idéias, alguns autores propõem a utilização de representação distribuída de conhecimento em redes neurais, que pode ser definida como aquela em que muitos neurônios participam da representação de cada símbolo e que diferentes representações compartilham neurônios. Um modelo distribuído não realiza nenhum comprometimento com nenhuma forma de representação, além da suposição de que estas representações acontecem de forma distribuída.

Os sistemas de representação distribuídos podem ser descritos através de duas propriedades: extensão e sobreposição. Uma representação é dita estendida se ela apresenta mais de uma unidade para representar um determinado conceito. Por superposição se entende a existência de unidades que sejam ativadas na representação de mais de um conceito.

As principais vantagens de uma abordagem localista são relacionadas com a sua proximidade do nível simbólico para a representação e manipulação do conhecimento. Browne e Sun (2001) citam que estes modelos são capazes de realizar raciocínio simbólico baseado em regras e substituir sistemas híbridos que contenham componentes simbólicos de processamento. Esta proximidade também permite uma maior facilidade para inserção e extração de conhecimento simbólico nas redes neurais, além de possibilitar a manipulação de ligação de variáveis (como descrito adiante).

As maiores desvantagens apontadas para a representação localista (e contornadas no caso das representações distribuídas) consistem na dificuldade de generalização, por apresentarem rigidez semelhante ao sistemas simbólicos, sua dificuldade para aplicação de algoritmos de aprendizagem conexionistas, a alta complexidade causada pela necessidade de representação de todos os conceitos utilizados e a falta de robustez para tratar ruídos (tolerância a falhas nos dados).

Page, em sua defesa dos sistemas localistas (PAGE, 2000), mostra que grande parte destas desvantagens consideradas não são inteiramente verdadeiras, devido principalmente ao fato que sistemas de representação localistas podem apresentar um componente distribuído, sendo acrescentadas unidades para representação local de conceitos. Mais explicitamente, se um conceito é representado de forma local em uma camada da rede neural, ele provavelmente se encontre representado de forma distribuída em uma camada anterior.

Como exemplo de sistema localista, baseado em lógica proposicional para representação do conhecimento simbólico, podemos citar o modelo proposto por Towell e Shavlik (1994). Este modelo chamado KBANN (Knowledge-Based Artificial Neural Networks) se baseia em duas etapas. Na primeira etapa, o conhecimento é inserido na rede através

de regras na forma de sentenças da lógica proposicional, mais especificamente cláusulas de Horn. Estas regras definem as conexões entre as camadas e o peso destas conexões. O conhecimento existente na rede ao final desta etapa é exatamente aquele especificado pelo conjunto de regras. Na segunda etapa, a rede é treinada com exemplos através do algoritmo Backpropagation (com pequenas alterações). Após esta etapa, o conhecimento inicial da rede estará alterado, de forma a se adaptar às características dos exemplos aplicados no treinamento.

A inserção de conhecimento simbólico em uma rede KBANN se dá através da utilização de um neurônio, com função de ativação sigmóide unipolar, para a representação de cada variável proposicional, bem como para representar sub-expressões do programa lógico na forma de conjunções e disjunções, considerando um valor de ativação maiores que 0.5 para o neurônio no caso da expressão ser interpretado positivamente, e um valor menor que 0.5 representando a negação. Desta forma, o bias de cada neurônio é adaptado de forma a permitir que a operação desejada seja computada. A figura 4.2 ilustra um exemplo deste processo de inserção de conhecimento.

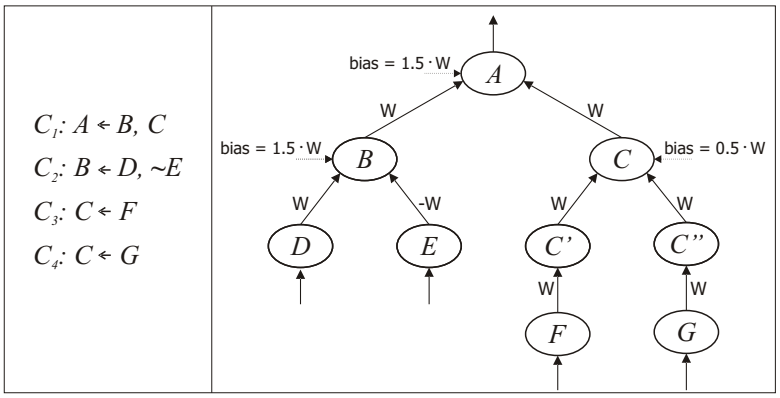

Figura 4.2: Exemplo de rede KBANN

Dentre as principais desvantagens deste método, estão uma limitação teórica com relação ao número máximo de entradas para cada neurônio. Isto ocorre pois a representação utilizada é unipolar, utilizando valores próximos a 0 para representar o valor lógico *"falso"* e valores próximos a 1 para representar *"verdadeiro"*. Desta forma, o somatório de valores próximos a 0 poderia levar a um valor alto, interpretado como 1 pela rede. Além disso, outro problema existente é o crescimento do número de camadas ocultas em função do tamanho da maior cadeia de dependências dentre as sentenças lógicas de inicialização da rede. Tal crescimento pode acarretar em diminuição no desempenho durante a tarefa de aprendizagem.

4.3 Aprendizagem e Extração de Conhecimento

Para sistemas baseados no uso de redes neurais para realização de inferência simbólica, dois aspectos, além da representação de conhecimento na arquitetura de rede, são importantes: a aprendizagem empírica e a extração de regras simbólicas a partir do conhecimento armazenado na rede neural. A união destes três aspectos principais constitui o ciclo de comportamento dos sistemas de aprendizagem neuro-simbólicos (D'AVILA GARCEZ; BRODA; GABBAY, 2002).

Dentro da engenharia de conhecimento, um dos grandes desafios encontrados é o cha-

mado *Knowledge Acquisition Bottleneck* (MICHALSKI, 1987), ou seja, a dificuldade de obter e codificar a base de conhecimento necessária para o desenvolvimento de sistemas inteligentes. Este problema, reconhecido há duas décadas pela comunidade científica (FEIGENBAUM, 2003), e a busca de técnicas para contorná-lo, alavancou as pesquisas na chamada aprendizagem de máquina.

A aprendizagem é considerada um componente essencial do comportamento inteligente, e pode ser definida como a capacidade de uma entidade em adaptar-se a estímulos e mudanças recebidos do ambiente, objetivando melhorias em seu desempenho (MICHALSKI, 1987). A pesquisa em aprendizagem de máquina consiste na tentativa de modelar computacionalmente este processo de adaptação. Existem diferentes formas de realizar aprendizagem, com destaque ao processo de generalização de conceitos, por parte do aprendiz, a partir de exemplos apresentados pelo ambiente.

Mesmo com a existência de diversas estratégias para a realização da aprendizagem empírica em sistemas simbólicos (MICHALSKI, 1987), estes são alegadamente menos adequados para esta tarefa devido a rigidez de sua estrutura de representação (HILARIO, 1995). Por este motivo, a integração de redes neurais artificiais a modelos simbólicos é uma estratégia altamente interessante para a superação dos problemas descritos.

Por outro lado, sistemas conexionistas são deficientes no que diz respeito à compreensibilidade do conhecimento codificado em sua arquitetura. Por isso, outro grande foco na área de integração neuro-simbólica consiste no desenvolvimento de técnicas para representar, de forma simbólica, o conhecimento aprendido empiricamente por redes neurais. Este desenvolvimento objetiva, entre outras coisas, aumentar a aplicabilidade do uso de redes neurais, especialmente em domínios de segurança crítica, onde as respostas obtidas pelo sistema precisam ser justificadas de forma compreensível.

Andrews, Diederich e Tickle (1995) descrevem uma taxonomia para os métodos de extração de regras de redes neurais, que se baseia em diferentes aspectos: A expressividade das regras, a translucência da rede no processo de extração, a generalidade do método com relação à aplicabilidade em diferentes arquiteturas de rede, a qualidade das regras extraídas e a complexidade do algoritmo.

A translucência da rede diz respeito ao tratamento da rede pelo método de extração, isto é, se a rede é considerada como um sistema único e indivisível (caixa preta), ou se as configurações internas da rede são utilizadas para a extração. Com relação a este aspecto, duas classes principais podem ser definidas. Os métodos decomposicionais (locais) tratam a extração das regras no nível dos neurônios e conexões da rede, ou seja, utiliza a informação estrutural da rede de forma transparente. Os métodos pedagógicos (globais) tratam as redes como caixas pretas, ou seja, se valem do resultado da aplicação da rede a alguns casos de exemplo para generalizar seu comportamento.

Dentre os métodos decomposicionais mais tradicionais, podemos citar o método Subset, descrito por Towell e Shavlik (1994) como um algoritmo local de extração de conhecimentos cujo funcionamento é baseado na tentativa de encontrar subconjuntos de ligações para cada neurônio oculto ou de saída de forma que a soma dos pesos destas ligações exceda o valor negativo do bias.

O grande problema deste algoritmo consiste na explosão combinatória, tendo em vista que a busca por subconjuntos cresce exponencialmente em função do número de ligações. Desta forma, em algumas aplicações o tamanho dos subconjuntos é limitado para contornar este problema. Porém, desta forma, o algoritmo perde parte da sua capacidade de expressão.

O algoritmo NofM, também baseado em uma abordagem local, difere do Subset por

buscar expressões da forma “N dos seguintes M antecedentes são verdadeiros” (N of M). Este método surgiu porque foi notado que muitas das regras descobertas pelo método Sub-Set continham conceitos neste estilo. Tendo por base este conceito, o algoritmo torna-se mais geral do que o Subset, pois as regras meramente conjuntivas ou meramente disjuntivas podem também ser descritas na forma “N of M” (TOWELL; SHAVLIK, 1994).

O funcionamento deste algoritmo se baseia na idéia de encontrar, para cada unidade oculta ou de saída, grupos de ligações com pesos similares. Dentre estes grupos, são desconsiderados aqueles de menor influência (com poucos membros ou pesos baixos). Após este processo, a rede passa novamente por um algoritmo Backpropagation suavemente alterado, para atualização apenas dos valores de bias. Depois desta atualização dos bias é que as regras na forma “N of M” são extraídas de cada unidade, passando depois por um processo de simplificação e eliminação de redundâncias.

Como exemplo de sistema pedagógico, podemos citar o algoritmo TREPAN (TREes PArroting Networks), que tem a finalidade de representar o conhecimento armazenado em uma rede neural artificial na forma de uma árvore de decisão (MARTINELLI, 1999). Sua tarefa é gerar uma árvore de decisão tomando como entrada uma rede neural treinada e os dados utilizados para o seu treinamento. Este algoritmo segue a abordagem global e não leva em consideração a arquitetura interna da rede, o que o torna bastante genérico para ser aplicado à maioria dos modelos de redes neurais (MARTINELLI, 1999; NOBRE et al., 1998).

O TREPAN emprega o mesmo princípio utilizado pelos algoritmos convencionais de indução de árvores de decisão, que constroem uma árvore de decisão através de particionamentos recursivos do conjunto de exemplos. Porém, em vez de utilizar a informação dada pelo conjunto de treinamento para a indução das árvores de decisão, a classe associada a cada exemplo é definida através da aplicação do processamento da rede sobre um novo conjunto de exemplos. Desta forma, o este conjunto de exemplos pode ser definido de forma diferenciada, proporcionando melhorias no processo de obtenção da árvore de decisão.

4.4 Transpondo a limitação proposicional

Devido às razões definidas anteriormente, existe um crescente interesse na pesquisa na integração dos paradigmas simbólico e conexionista, e mais especificamente no uso de redes neurais para a representação de raciocínio simbólico. Ainda assim, podemos considerar que ainda existem obstáculos consideráveis para ser transpostos. Dentre eles, sob um ponto de vista computacional, podemos citar a natureza proposicional dos sistemas desenvolvidos até então (HITZLER; HÖLLDOBLER; SEDA, 2004). Poucos sistemas contemplam uma expressividade que transcende esta natureza proposicional em busca de um sistema capaz de representação de funções e predicados de primeira ordem, e ainda assim os sistemas existentes apresentam diversas limitações.

Dentro de uma representação de conhecimento, símbolos podem ser representados por variáveis, que podem ou não ser instanciadas, ou seja, ser designadas para representação de uma entidade específica. Uma das idéias importantes para representação de lógicas mais expressivas consiste na incorporação de ligação de variáveis (variable binding), que define a associação entre variáveis da representação e conceito reais, ou mesmo entre diferentes variáveis dentro da representação. Esta ligação pode ser descrita como uma forma complexa de casamento de padrões. Segundo Browne e Sun (2000) existem 3 formas de realizar esta operação:

- Casamento entre símbolos atômicos, ou seja, comparação de dois símbolos sintaticamente semelhantes, sem realizar associação com nenhuma variável propriamente dita;
- Casamento *"standard"* de padrões, onde as variáveis consideradas em um padrão são ligadas a entidades reais do domínio;
- Unificação, onde ocorre o casamento entre variáveis do padrão a ser avaliado e outras variáveis existentes na descrição do modelo.

Podemos enumerar diversos argumentos que para justificar a importância do processo de ligação de variáveis para a realização de tarefas de raciocínio complexo (BROWNE; SUN, 2000):

- A ligação de variáveis é essencial para a criação de estruturas dinâmicas em sistemas conexionistas
- A ligação de variáveis é necessária para aumentar o poder computacional em sistemas conexionistas, tendo em vista que sem o uso de variáveis, algumas regras gerais teriam que ser representadas através de um conjunto muito grande de regras específicas;
- A ligação de variáveis também é considerada como fundamental para representação da cognição humana, como por exemplo na modelagem da faculdade de manipulação de linguagem.

Existem diferentes formas de implementar a ligação entre variáveis em sistemas conexionistas. Em sistemas localistas, podemos citar a propagação de sinais, que consiste em alocar um nó para representar cada variável associada com cada conceito, e alocar um valor para representar cada objeto particular no domínio, sendo tratado como um sinal deste objeto. Desta forma, o valor de ativação de cada neurônio pode ser tratado como o sinal (representação) de cada objeto, e este valor pode ser propagado através da rede.

Outro método localista para implementar ligação de variáveis é chamado sincronização de fase, que consiste na utilização dos aspectos temporais da ativação em adição (ou substituição) dos valores de ativação instantâneos. O processo de sincronização de fase envolve a utilização de diferentes fases do ciclo de ativação do neurônio para a representação dos diferentes objetos envolvidos no raciocínio, e a utilização do sincronismo entre estas fases para representar a ligação de variáveis propriamente dita.

Dentro de sistemas que utilizam uma abordagem distribuída para representação do conhecimento, podemos citar o método de produto de tensores para efetuar a ligação das variáveis. Este método utiliza uma descrição onde estruturas são representadas por conjuntos de *slots*, que definem os papéis ou atributos, e *fillers*, que definem os valores. Desta forma o produto de tensores é utilizado para permitir a representação de um conjunto de pares variável-valor através da ativação acumulada em coleções de tamanho fixo de unidades.

Ainda assim, existem diversos obstáculos para a realização da tarefa de ligação de variáveis. Uma outra solução para representar lógicas de primeira ordem, com uma implementação limitada da ligação de variáveis, consiste em instanciar as ocorrências das variáveis nos diferentes predicados, caindo novamente no caso proposicional. O grande problema desta abordagem é o aumento no número de átomos a ser representados, e conseqüentemente do número de neurônios na rede. Como solução deste problema, Hitzler,

Hölldobler e Seda (2004) propõem a utilização de um mapeamento das possíveis interpretações do conjunto de átomos do programa para números reais. Tal mapeamento deve ser inversível, de forma a permitir a conversão entre a representação lógica (interpretações) e a representação numérica em ambos os sentidos. Desta forma, tendo uma representação numérica dos elementos do domínio, é possível utilizar a capacidade de aproximação de funções reais, intrínseca às redes neurais, com o intuito de emular o processamento lógico.

5 CILP E EXTENSÕES

5.1 CILP

Hölldobler e Kalinke (1994) propuseram um modelo neuro simbólico baseado na representação localista de programas lógicos em uma rede neural MLP, utilizando neurônios nas camadas de entrada e saída para representar cada átomo do programa e neurônios na camada oculta para representar as cláusulas. Nesta abordagem, conexões são inseridas entre neurônios da camada de entrada (representando um átomo A) e neurônios da camada oculta (representando cláusulas em que A aparece no corpo). Os pesos destas conexões são definidos de acordo com a forma que o átomo aparece na cláusula, ou seja, são positivos se o átomo aparece positivamente (A) ou negativos se o átomo está na forma negada ($\sim A$) . Estes pesos das conexões e o peso do bias dos neurônios ocultos são configurados de forma que o valor de ativação seja positivo se, e somente se, a conjunção dos literais representados na camada de entrada for verdadeira. Da mesma maneira, são inseridas conexões entre neurônios ocultos, representando cláusulas cuja cabeça seja um átomo A, e neurônios de saída representando este átomo. Estas conexões apresentam peso positivo, e o bias dos neurônios de saída é configurado de forma que a saída seja positiva se, e somente se, ao menos um dos valores aplicados à sua entrada for positivo.

A figura 5.1 mostra a rede gerada através da tradução do programa descrito seguindo o modelo de Holldobler e Kalinke. Tomando como exemplo a cláusula $B \leftarrow\sim D, E$, notamos que o neurônio oculto C_2 recebe as duas entradas, representando os átomos D e E, ponderadas por pesos de acordo com a forma que os átomos aparecem. Neste caso, o bias será igual a $-1.5W$ (definido por $-(nW - 0.5)$, onde n é o número de entradas no neurônio). Este valor é selecionado para que o neurônio seja ativado se, e somente se, todos os valores aplicados na entrada forem positivos (após multiplicado pelo peso), ou seja, se D for falso e E for verdadeiro, o somatório ponderado das entradas do neurônio serão iguais a $2W$, que adicionando ao bias resulta em um valor positivo. Nos demais casos, o máximo valor possível para a soma ponderada das entradas será 1, levando a um valor negativo se somado ao bias. Lembrando que a função de ativação é a função de limiar abrupto, a saída do neurônio será igual a 1 somente no primeiro caso. Considerando, de forma similar, o exemplo do neurônio de saída representando o átomo C, temos que o bias é configurado com o valor $0.5W$, de forma que a saída apresente valor positivo quando, ao menos, uma entrada for positiva, e valor negativo se todas as entradas forem negativas.

Mesmo representando corretamente a semântica de um programa lógico, este modelo apresenta uma importante desvantagem com relação à aplicabilidade da rede MLP gerada. Conforme descrito anteriormente, os principais algoritmos de treinamento para rede MLP, como o caso do Backpropagation, se baseiam na derivada da função de ativação para cálculo da estimativa de erro a ser retropropagada, requerendo o uso funções de ati-

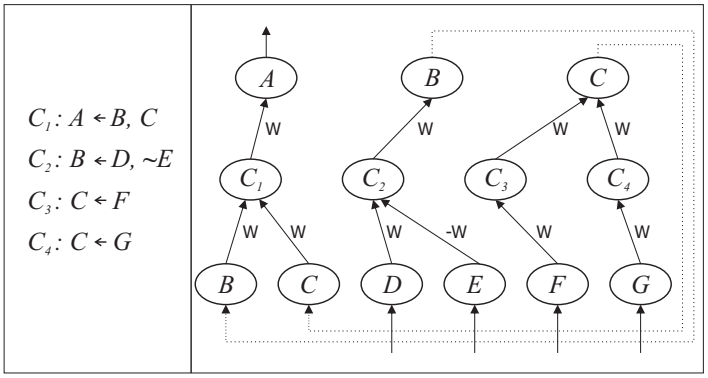

Figura 5.1: Exemplo de rede de Holldobler e Kalinke

vação contínuas e diferenciáveis. Desta forma, estes algoritmos não são aplicáveis para o modelo de Holldobler e Kalinke, tendo em vista que estes utilizam funções de limiar abrupto. Ainda com relação à escolha de função de ativação, o uso de uma função unipolar como no caso do modelo KBANN pode acarretar em um comportamento indesejado. A negação de um átomo nestes modelos pode ser representada por um valor positivo de baixo módulo (próximo a zero), porém a soma de diversos destes valores pode conduzir a um resultado de módulo alto o suficiente para levar a uma falsa interpretação positiva de um átomo. Desta forma, o modelo apresenta uma limitação teórica com relação ao número máximo de entradas por neurônio, ou seja, de átomos que podem ser representado em uma mesma conjunção ou disjunção.

Seguindo a mesma estratégia de Holldobler e Kalinke (1994) e buscando corrigir os problemas mencionados, D'Avila Garcez e Zaverucha (1999) propuseram o modelo CILP (Connectionist Inductive Learning and logic Programming), que faz uso de uma função de ativação sigmoidal bipolar para os neurônios ocultos e de saída da rede neural gerada. Consideraremos, para o modelo CILP, as seguintes definições:

- $k(l)$ e $\mu(l)$ consistem no número de literais no corpo de uma cláusula C_l e no número de cláusulas com a mesma cabeça de C_l, respectivamente;

- $Max_{k\mu}$ define o maior valor entre o número de literais no corpo de uma mesma cláusula e o número de cláusulas com uma mesma cabeça. Na rede gerada, este valor representa o número máximo de entradas em um mesmo neurônio;

- A_{min} consiste em uma constante positiva, menor que 1, tal que uma interpretação positiva (para uma variável proposicional) é representada por um valor numérico positivo entre A_{min} e 1, e uma interpretação negativa é representada por um valor entre -1 e $-A_{min}$. Este valor é definido, arbitrariamente, de forma que $\frac{1-Max_{k\mu}}{1+Max_{k\mu}} < A_{min} < 1$;

- $\phi(x)$ é a função logarítmica bipolar $\frac{2}{1+e^{-\beta x}} - 1$, onde β é um parâmetro que define o 'slope' (decaimento) da função. $\psi(x)$ é a função linear (identidade);

- W é o valor do peso das conexões sinápticas positivas, e $-W$ é o peso das conexões negativas. W é definido por um valor maior que $\frac{ln(1+A_{min})-ln(1-A_{min})}{Max_{k\mu}(A_{min}-1)+A_{min}+1} \cdot \frac{2}{\beta}$

Os valores de W e A_{min} são definidos para permitir a computação correta da semântica desejada, como descrito mais detalhadamente em outros trabalhos (D'AVILA GARCEZ; ZAVERUCHA, 1999; D'AVILA GARCEZ; BRODA; GABBAY, 2002). Os valores de bias dos neurônios são definidos de acordo com o número de entradas, de forma a garantir a representação de conjunção pelos neurônios ocultos (e disjunção pelos neurônios de saída). Considerando n como o número de entradas, os valores de bias para os neurônios ocultos e de saída são definidos como $\frac{(1+A_{min})(k(l)-1)}{2}W$ e $\frac{(1+A_{min})(1-\mu(l))}{2}W$, respectivamente. A figura 5.2 mostra uma descrição geral do algoritmo de tradução do algoritmo CILP, considerando a seguinte notação:

- $Clausulas(\mathcal{P})$: Conjunto de cláusulas em um programa lógico $\mathcal{P}$;
- $Atomos(\mathcal{P})$: Conjunto de átomos em um programa lógico $\mathcal{P}$;
- $Corpo(C_l)$: Conjunto de literais que aparecem no corpo de uma cláusula C_l;
- $Cabeca(C_l)$: Átomo que aparecem como cabeça de uma cláusula C_l;
- $Neuronios(\mathcal{N})$: Conjunto de neurônios em uma rede neural $\mathcal{N}$;
- $InsereNeuronioEntrada(\mathcal{N}, M)$ (respectivamente, $InsereNeuronioOculto$ e $InsereNeuronioSaida$): Procedimento para inserir um neurônio de entrada (respectivamente, oculto e de saída) M em uma rede neural $\mathcal{N}$;
- $Ativacao(N)$: Função de ativação de um neurônio N;
- $Bias(N)$: Bias de um neurônio N;
- $Conecta(\mathcal{N}, M, M', W)$: Insere uma conexão com peso W entre os neurônios M e M' da rede $\mathcal{N}$;

A aplicação do algoritmo de tradução CILP gera uma arquitetura de rede similar ao método de Hölldobler e Kalinke (1994), como mostrado na figura 5.1. No caso deste programa, considerando um valor de β igual a 1, e o valor de $Max_{k\mu}$ é definido por 2. Definimos, então, A_{min} como sendo um valor arbitrário maior que 0.333(1/3), (considere $A_{min} = 0.75$) e W como qualquer valor maior que 3.1135 (considere $W = 4$). Podemos notar, por exemplo, para o neurônio que representa a cláusula C_2, que o valor de bias será igual a -3.5 Neste caso, se ao menos uma das entradas estiver com valor negativo (menor que $-WA_{min}$) a soma ponderada das entradas e do bias será no máximo igual ao valor do bias, e a aplicação da função de ativação sobre esta soma será menor que $-A_{min}$ (0.9413). Por outro lado, se todos os valores forem positivos, a soma será maior que (2.5) e o valor de saída do neurônio será maior que 0.8482, ou seja, maior que A_{min}.

Teorema 5.1 *Uma rede neural gerada através da aplicação do algoritmo de tradução CILP_Translation sobre um programa lógico proposicional $\mathcal{P}$ computa o operador de conseqüência imediata $\mathcal{T}_{\mathcal{P}}$ de $\mathcal{P}$ (D'AVILA GARCEZ; ZAVERUCHA, 1999).*

Prova: O inverso da função de ativação $y = \frac{2}{1+e^{-\beta x}} - 1$ é dado por $x = ln(\frac{1+y}{1-y}) \times \frac{1}{\beta}$, portanto, sempre que a soma dos valores de entrada ponderados e do bias for maior ou igual a $ln(\frac{1+A_{min}}{1-A_{min}}) \cdot \frac{1}{\beta}$, o valor de ativação do neurônio será maior ou igual a A_{min}. Da mesma forma, se este somatório for menor ou igual a $-ln(\frac{1+A_{min}}{1-A_{min}}) \cdot \frac{1}{\beta}$, a ativação do neurônio será menor ou igual a $-A_{min}$

CILP_Translation($\mathcal{P}$)
  para cada $C_l \in Clausulas(\mathcal{P})$ **faça**
    $InsereNeuronioOculto(\mathcal{N}, h_l)$;
    para cada $A \in Corpo(C_l)$ **faça**
      se $in_A \notin Neuronios(\mathcal{N})$ **então**
        $InsereNeuronioEntrada(\mathcal{N}, in_A)$;
        $Ativacao(in_A) \leftarrow \psi(x)$;
      $Conecta(\mathcal{N}, in_A, h_l, W)$;
    fim
    para cada $\sim A \in Corpo(C_l)$ **faça**
      se $in_A \notin Neuronios(\mathcal{N})$ **então**
        $InsereNeuronioEntrada(\mathcal{N}, in_A)$;
        $Ativacao(in_A) \leftarrow \psi(x)$;
      $Conecta(\mathcal{N}, in_A, h_l, -W)$;
    fim
    se $out_{Cabeca(C_l)} \notin Neuronios(\mathcal{N})$ **então**
      $InsereNeuronioSaida(\mathcal{N}, out_{Cabeca(C_l)})$;
    $Conecta(\mathcal{N}, h_l, out_{Cabeca(C_l)}, W)$;
    $Bias(h_l) \leftarrow -\frac{(1+A_{min})(k(l)-1)}{2}W$;
    $Bias(out_{Cabeca(C_l)}) \leftarrow -\frac{(1+A_{min})(1-\mu_l)}{2}W$;
    $Ativacao(h_l) \leftarrow \phi(x)$;
    $Ativacao(out_{Cabeca(C_l)}) \leftarrow \phi(x)$;
  fim
  para cada $A \in Atomos(\mathcal{P})$ **faça**
    se $(in_A \in Neuronios(\mathcal{N})) \wedge (out_A \in Atomos(\mathcal{N}))$ **então**
      $Conecta(\mathcal{N}, out_A, in_A, 1)$
  fim
  return $\mathcal{N}$;
end

Figura 5.2: Algoritmo de tradução CILP

Primeiramente, vamos verificar que valor de ativação dos neurônios ocultos da rede CILP será maior que A_{min} se todos os valores de entrada ponderados pelo peso W (ou -W no caso negativo) forem maiores ou iguais a WA_{min}. Como o bias dos neurônios ocultos é igual a $-\frac{(1+A_{min})(k(l)-1)}{2}W$, o somatório do bias com as entradas ponderadas será igual a $k(l) \cdot WA_{min} - \frac{(1+A_{min})(k(l)-1)}{2}W = \frac{k(l)(A_{min}-1)+A_{min}+1}{2} \cdot W$. Como $W > \frac{ln(1+A_{min})-ln(1-A_{min})}{Max_{k\mu}(A_{min}-1)+A_{min}+1} \cdot \frac{2}{\beta}$ e $Max_{k\mu} \geq k(l)$, temos que o somatório sera maior que $ln(\frac{1+A_{min}}{1-A_{min}}) \cdot \frac{1}{\beta}$, portanto a ativação do neurônio será maior ou igual a A_{min}.

Por outro lado, se ao menos uma das entradas ponderadas for menor ou igual a $-WA_{min}$, temos que o valor máximo do somatório será quando $k(l)-1$ entradas ponderadas forem iguais a W (entrada unitária ponderada pelo peso) e a restante for igual a $-WA_{min}$. Desta forma o somatório dos valores será igual a $-\frac{k(l)(A_{min}-1)+A_{min}+1}{2} \cdot W$ e, consequentemente, menor que $ln(\frac{1+A_{min}}{1-A_{min}}) \cdot \frac{1}{\beta}$. Portanto a ativação do neurônio será menor ou igual a $-A_{min}$.

Desta forma, podemos verificar que um neurônio da camada oculta computa a representação de uma conjunção dos valores de entrada. De forma similar, podemos provar que, devido à definição do valor de bias, os neurônios de saída computam a disjunção dos valores de entrada. Pela definição de $\mathcal{T}_{\mathcal{P}}$, temos que $\mathcal{T}_{\mathcal{P}}(I_{\mathcal{P}})(A)$ para um átomo A sera verdadeiro se, e somente se, houver uma cláusula cuja cabeça seja A e a conjunção dos

literais em seu corpo seja verdadeira, ou seja, se a disjunção formada pelas conjunções de cada cláusula for verdadeira. Desta forma, considerando a estrutura de conexões da rede, temos que um átomo de saída A é representado na rede por um neurônio de saída que computa esta disjunção e, portanto, computa $\mathcal{T}_{\mathcal{P}}$. □

A figura 5.1 vale como exemplo, também, de uma aplicação do algoritmo de tradução CILP. Nesta figura, podemos notar a inserção de ligações recorrentes entre neurônios de saída e entrada representando o mesmo átomo. Estas ligações são geradas com o objetivo de aplicar na entrada da rede o valor resultante da computação anterior do operador $\mathcal{T}_{\mathcal{P}}$, e desta forma permitir o cálculo recursivo deste operador até a convergência para um ponto fixo, que representa a semântica desejada do programa. Desta forma, temos que a rede gerada pelo algoritmo de tradução é capaz de computar a semântica de ponto fixo para qualquer programa lógico acíclico.

5.2 Representando Lógicas Modais

Buscando aumentar a representatividade de sistemas neuro simbólico, superando as limitações da lógica proposicional clássica, diferentes extensões para o modelo CILP foram propostas para tratamento de lógicas não clássicas. D'Avila Garcez, Lamb e Gabbay (2002) propuseram um modelo neuro simbólico para representar a semântica de programas em lógica modal utilizando conjuntos de redes CILP. Este sistema, chamado CML (Connectionist Modal Logics) é baseado na estrutura relacional entre mundos possíveis de Kripke, considerando uma sintaxe que utiliza rótulos para cada cláusula, de forma que todas as cláusulas que apresentam o mesmo rótulo formam a descrição de um mundo possível. Cada conjunto de cláusulas com o mesmo rótulo é tratado como um programa individual, e representado na forma conexionista por uma rede gerada através aplicação do algoritmo CILP. Além disto, a semântica dos operadores de necessidade □ e possibilidade ◊ são representados através de neurônios inseridos na camada oculta de uma rede, recebendo ligações vindas dos neurônios de saída de redes representando os outros mundos possíveis. A semântica destes operadores é definida na tabela 5.1, de acordo com o proposto por Broda et al. (2004).

Tabela 5.1: Regras para Operadores Modais

$$\frac{[R(\omega,g_\alpha(\omega))]\ldots g_\alpha(\omega):\alpha}{\omega:\Box\alpha}\,\Box I \qquad \frac{\omega_1:\Box\alpha, R(\omega_1,\omega_2)}{\omega_2:\alpha}\,\Box E$$

$$\frac{\omega:\Diamond\alpha}{f_\alpha(\omega):\alpha, R(\omega,f_\alpha(\omega))}\,\Diamond E \qquad \frac{\omega_2:\alpha, R(\omega_1,\omega_2)}{\omega_1:\Diamond\alpha}\,\Diamond I$$

A representação das regras $\Box E$ e $\Diamond I$ são bastante intuitivas. Para a primeira, é utilizado um neurônio oculto na rede que representa ω_2 recebendo informação de todos os neurônios de saída representando $\Box\alpha$ em mundos ω_1. Este neurônio oculto computa a disjunção destes valores, e aplica o valor ao neurônio de saída representando α em ω_2. Como exemplo desta aplicação, podemos ver na figura 5.3 que o neurônio de saída representando o átomo $\Box A$ em ω_1 é ligado a neurônios oculto nas redes que representam ω_2 e ω_3. Tais neurônios, por sua vez, são associados à representação do átomo A nas respectivas redes.

A regra $\Diamond I$ é representada de forma similar, com um neurônio computando a disjunção de todos os átomos α em mundos ω_2 tais que $R(\omega_1,\omega_2)$ e aplicando o valor no neurônio

representando $\Box\alpha$ em ω_1. O exemplo deste caso é mostrado na figura 5.3 para a representação do $\Diamond C$ na rede ω_1, recebendo ligação da rede representando ω_2.

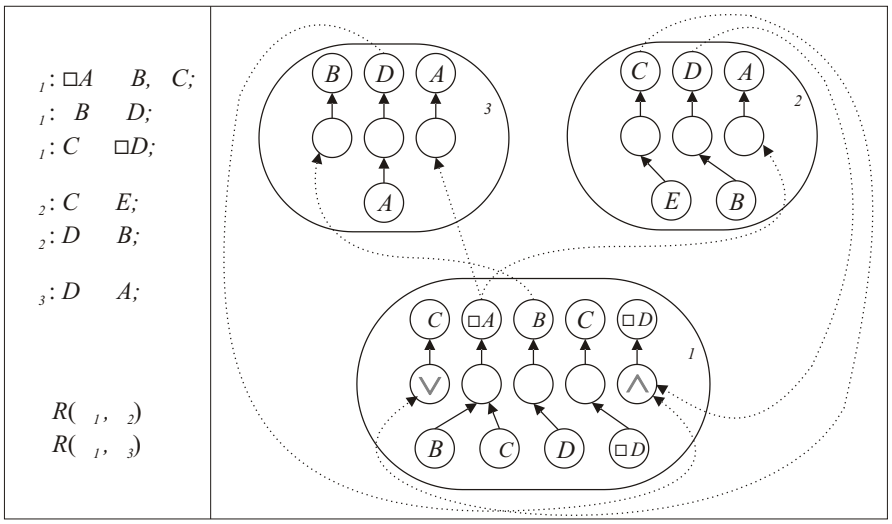

Figura 5.3: Exemplo da tradução de um programa modal

Para representar a regra $\Box_I$ com relação a um átomo $\Box\alpha$, o algoritmo insere um neurônio oculto na rede que representa um mundo possível w, de forma que este neurônio receba o valor de saída de todos os neurônios representando o átomo A em redes associadas a mundos $\omega'|R(\omega,\omega')$, e compute a conjunção destes valores. Ainda na mesma figura, esta representação é ilustrada no caso do átomo $\Box D$ em ω_1.

Para a regra $\Diamond_E$ podemos considerar duas diferentes maneiras de selecionar a rede que represente um mundo possível arbitrário $f_\alpha(\omega)$ para a representação de α: Gerar uma nova rede para representar $f_\alpha(\omega)$, ou selecionar $f_\alpha(\omega)$ dentre as redes existentes. No nosso exemplo, consideramos a segunda abordagem, mostrada no caso do átomo $\Diamond B$ de ω_1, selecionando-se ω_2 para aplicação do valor. Na figura 5.5, mostramos o algoritmo para tradução de programas lógicos modais. D'Avila Garcez, Lamb e Gabbay (2002) provaram que redes geradas por este algoritmo computam corretamente a semântica de ponto fixo de programas modais aceitáveis.

5.3 Representando Tempo e Conhecimento

Seguindo a mesma idéia de utilizar um conjunto de redes para representar diferentes mundos possíveis, encontramos no trabalho de D'Avila Garcez e Lamb (2006) uma proposta para representação de programas em lógica temporal e epistêmica. Nesta abordagem, chamada CTLK (Connectionist Temporal Logic of Knowledge), o conjunto de redes usado para representar os diferentes mundos possíveis é tratado de forma bidimensional, sendo uma dimensão para representar o fluxo de tempo e uma dimensão para representação dos diferentes agentes.e), o conjunto de redes usado para representar os diferentes mundos possíveis é tratado de forma bidimensional, sendo uma dimensão para representar o fluxo de tempo e uma dimensão para representação dos diferentes agentes.

A representação temporal no sistema é descrita em função do operador de próximo momento $\bigcirc$. A regra de introdução do operador, com relação a um átomo $\bigcirc A$, é representada através de um neurônio oculto na rede representando um ponto t no fluxo de tempo. Este neurônio recebe como entrada o valor do neurônio de saída representando o átomo

A no momento $t+1$, e por sua vez alimenta o neurônio de saída representando $\bigcirc A$ em t. De maneira similar, um neurônio oculto é usado para realizar a ligação do neurônio representando $\bigcirc A$ em $t-1$ para o neurônio representando A em t. A semântica considerada para o operador de conhecimento K é definida de maneira similar ao operador $\Box$, porém nenhuma rotina para a representação de regras semânticas deste operador no conjunto de redes neurais é definido. Mostramos, na figura 5.6 o algoritmo de tradução para o sistema CTLK. Na figura 5.4 ilustramos um exemplo da aplicação do algoritmo CTLK sobre um programa em lógica temporal, mostrando as ligações entre neurônios representando $\bigcirc A$ e $\bigcirc B$ em t_1, para A e B, respectivamente, em t_2, e de $\bigcirc A$ em t_2 para A em t_3.

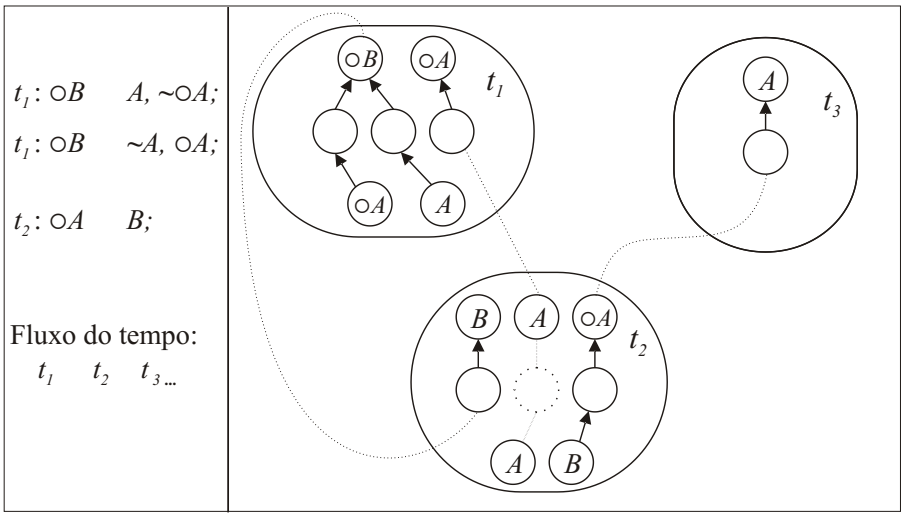

Figura 5.4: Exemplo da tradução de um programa temporal

5.4 Outros sistemas de raciocínio e extração de regras

O modelo CILP serviu como base, também para representar outros sistemas lógicos e mecanismos de raciocínio. No trabalho de D'Avila Garcez e Lamb (D'AVILA GARCEZ; LAMB, 2006) encontramos uma abordagem para representação de programas em lógica intuicionista, com semântica definida através do conceito de mundos possíveis, de forma que a negação de um átomo A ($\neg A$) seja definida como verdadeira em um mundo possível w sempre que A não for definido como verdadeiro em nenhum mundo possível $w'|R(w,w')$. Novamente, um conjunto de redes CILP são utilizados para a representação do conjunto de mundos possíveis, de forma que neurônios ocultos e ligações entre as redes sejam inseridos para representação da semântica desejada.

Outro sistema neuro-simbólico para raciocínio e aprendizagem baseado no modelo CILP consiste na representação conexionista de frameworks de argumentação baseados em valores. Estes frameworks, propostos por Bench-Capon (2003), utilizam grafos direcionados para a representação de modelos de argumentação, onde cada nó do grafo representa um argumento e cada aresta representa uma relação de ataque entre os argumentos. Diferentes valores, dentro de um conjunto, podem ser atribuídos para os argumentos, e uma relação de ordem total entre estes valores (associada ao conceito de audiência) pode ser definida, de forma que um argumento é preservado se não for atacado por nenhum argumento de valor maior que ele, ou se todos argumentos que o atacam forem, por sua vez, atacados por argumentos de valor ainda maior. A representação conexionista deste

modelo, proposta por d'Avila Garcez, Gabbay e Lamb (2005), consiste em representar um conjunto de cláusulas da forma $a_i \leftarrow a_i$, para todo argumento a_i no modelo, e inserir ligações entre a camada oculta e a camada de saída de forma a inibir a ativação dos neurônios que representam um argumento quando este é atacado por algum argumento de valor maior (de acordo com a ordem estabelecida).

Para extração de regras em redes geradas pelo algoritmo CILP, bem como qualquer rede MLP que apresente uma única camada oculta, D'Avila Garcez, Broda e Gabbay (2001) propõem um método baseado na união das abordagens pedagógica e decomposicional. A abordagem global (pedagógica) é utilizada para extração de regras de redes neurais regulares, ou seja, redes com uma única camada oculta tais que as todas conexões que partem de um mesmo nó de entrada (e todas as conexões que atingem o mesmo nó de saída) apresentem o mesmo sinal. Para a utilização desta abordagem, utiliza-se um ordenamento parcial dos vetores de entrada utilizados no processo de extração, objetivando a redução do espaço de busca de regras possíveis para descrever o conhecimento da rede, e atenuando assim uma das grandes desvantagens da abordagem pedagógica. Os autores provam que o conjunto de regras extraído desta forma é completo e consistente com relação ao conhecimento da rede.

Para estender o algoritmo a redes não regulares, o processo se baseia no fato de que toda rede não regular pode ser decomposta em componentes regulares. Desta forma, utiliza-se a abordagem local para a extração do conhecimento da rede, ou seja, aplica-se o processo descrito acima localmente a cada componente regular da rede. Nesta aplicação, a condição de completude do método é perdida, porém a sua consistência é mantida.

CML_Translation($\mathcal{P}$)
para cada $C_l \in Clausulas(\mathcal{P})$ **faça**
se $Rotulo(C_l) = \omega_i$ **então** $InsereClausula(C_l, \mathcal{P}_i)$
fim
para cada $\mathcal{P}_i$ **faça**
para cada $\Box L_j \in Literais(\mathcal{P})$ **faça** $Renomeia(\Box L, L_j^{\Box})$;
para cada $\Diamond L_j \in Literais(\mathcal{P})$ **faça** $Renomeia(\Diamond L, L_j^{\Diamond})$;
$\mathcal{N}_1 \leftarrow CILP\_Translation(\mathcal{P}_1)$;
fim
para cada $\mathcal{N}_i$ **faça**
para cada $out\_L_j^{\Diamond} \in Neuronios(\mathcal{N})$ **faça**
$SelecionaRedeArbitraria(\mathcal{N}_k, R(i,k))$;
$InsereNeuronioOculto(\mathcal{N}_k, N_j^M)$;
$Bias(\mathcal{N}_k, N_j^M) \leftarrow -\theta^M$;
$Ativacao(\mathcal{N}_k, N_j^M) \leftarrow s(x)$;
$Conecta(\mathcal{N}_i, out\_L_j^{\Diamond}, \mathcal{N}_k, N_j^M, 1)$;
$Conecta(\mathcal{N}_k, N_j^M, \mathcal{N}_k, out\_L_j, W^M)$;
$InsereNeuronioOculto(\mathcal{N}_i, N_j^{\vee})$;
$Ativacao(\mathcal{N}_i, N_j^{\vee}) \leftarrow s(x)$;
$Bias(\mathcal{N}_i, N_j^{\vee}) \leftarrow -\theta^{\vee}$;
$Conecta(\mathcal{N}_i, N_j^{\vee}, \mathcal{N}_i, out\_L_j^{\Diamond}, W^M)$;
para cada $out\_L_j \in Neuronios(\mathcal{N}_k) | R(i,k)$ **faça**
$Conecta(\mathcal{N}_k, out\_L_j, \mathcal{N}_i, N_j^{\vee}, 1)$;
fim
para cada $out\_L_j^{\Box} \in Neuronios(\mathcal{N})$ **faça**
para cada $\mathcal{N}_k | R(i,k)$ **faça**
$InsereNeuronioOculto(\mathcal{N}_k, N_j^M)$;
$Bias(\mathcal{N}_k, N_j^M) \leftarrow -\theta^M$;
$Ativacao(\mathcal{N}_k, N_j^M) \leftarrow s(x)$;
$Conecta(\mathcal{N}_i, out\_L_j^{\Box}, \mathcal{N}_k, N_j^M, 1)$;
$Conecta(\mathcal{N}_k, N_j^M, \mathcal{N}_k, out\_L_j, W^M)$;
fim
$InsereNeuronioOculto(\mathcal{N}_i, N_j^{\wedge})$;
$Ativacao(\mathcal{N}_i, N_j^{\wedge}) \leftarrow s(x)$;
$Bias(\mathcal{N}_i, N_j^{\wedge}) \leftarrow -\theta^{\wedge}$;
$Conecta(\mathcal{N}_i, N_j^{\wedge}, \mathcal{N}_i, out\_L_j^{\Box}, W^M)$;
para cada $out\_L_j \in Neuronios(\mathcal{N}_k) | R(i,k)$ **faça**
$Conecta(\mathcal{N}_k, out\_L_j, \mathcal{N}_i, N_j^{\wedge}, 1)$;
fim
fim
end

Figura 5.5: Algoritmo de tradução CML

CTLK_Translation($\mathcal{P}$)
  para cada $C_l \in Clausulas(\mathcal{P})$ **faça**
    $t \leftarrow Rotulo(C_l)$;
    if $\bigcirc\alpha \in CorpoC_l$ **then**
      se $out_{\bigcirc\alpha} \notin Neuronios(\mathcal{N}_t)$ **então** $InsereNeuronioSaida(\mathcal{N}_t, out_{\bigcirc\alpha})$;
      se $out_{\alpha} \notin Neuronios(\mathcal{N}_{t+1})$ **então** $InsereNeuronioSaida(\mathcal{N}_{t+1}, out_{\alpha})$;
      $Bias(\mathcal{N}_t, out_{\bigcirc\alpha}) \leftarrow Bias(\mathcal{N}_{t+1}, out_{\alpha}) \leftarrow \frac{(1+A_{min})(k_l-1)}{2}$;
      $Ativacao(\mathcal{N}_t, out_{\bigcirc\alpha}) \leftarrow Ativacao(\mathcal{N}_{t+1}, out_{\alpha}) \leftarrow \phi(x)$;
      $InsereNeuronioOculto(\mathcal{N}_t, H^{\vee})$;
      $Bias(\mathcal{N}_t, H^{\vee}) \leftarrow \theta^{\vee}$;
      $Ativacao(\mathcal{N}_t, H^{\vee}) \leftarrow s(x)$;
      $Conecta(\mathcal{N}_{t+1}, out_{\alpha}, \mathcal{N}_t, H^{\vee}, 1)$;
      $Conecta(\mathcal{N}_t, H^{\vee}, \mathcal{N}_t, out_{\bigcirc\alpha}, W^M)$;
    end
    if $\bigcirc\alpha = CabecaC_l$ **then**
      se $out_{\bigcirc\alpha} \notin Neuronios(\mathcal{N}_t)$ **então** $InsereNeuronioSaida(\mathcal{N}_t, out_{\bigcirc\alpha})$;
      se $out_{\alpha} \notin Neuronios(\mathcal{N}_{t+1})$ **então** $InsereNeuronioSaida(\mathcal{N}_{t+1}, out_{\alpha})$;
      $Bias(\mathcal{N}_t, out_{\bigcirc\alpha}) \leftarrow Bias(\mathcal{N}_{t+1}, out_{\alpha}) \leftarrow \frac{(1+A_{min})(k_l-1)}{2}$;
      $Ativacao(\mathcal{N}_t, out_{\bigcirc\alpha}) \leftarrow Ativacao(\mathcal{N}_{t+1}, out_{\alpha}) \leftarrow$;
      $InsereNeuronioOculto(\mathcal{N}_{t+1}, H^{\bigcirc})$;
      $Bias(\mathcal{N}_{t+1}, H^{\bigcirc}) \leftarrow \theta^{\bigcirc}$;
      $Ativacao(\mathcal{N}_{t+1}, H^{\bigcirc}) \leftarrow s(x)$;
      $Conecta(\mathcal{N}_t, out_{\bigcirc\alpha}, \mathcal{N}_{t+1}, H^{\bigcirc}, 1)$;
      $Conecta(\mathcal{N}_{t+1}, H^{\bigcirc}, \mathcal{N}_{t+1}, out_{\alpha}, W^M)$;
    end
  fim
end

Figura 5.6: Algoritmo de tradução temporal CTLK

6 REPRESENTAÇÃO SEQÜENCIAL DE TEMPO EM REDES CILP

6.1 Correções ao modelo CILP

Nossa proposta para hibridização entre sistemas simbólicos e conexionistas é baseado na tradução da semântica de programas lógicos temporais, como definidos na seção 2.4, para redes neurais. Porém, para permitir o desenvolvimento de técnicas de representação de aspectos temporais em sistemas baseados no modelo CILP, consideramos importante realizar, primeiramente, uma análise de diferentes aspectos concernentes ao comportamento das redes neurais geradas pelo algoritmo de tradução CILP, com relação à representação de um programa lógico. O primeiro aspecto a ser considerado diz respeito ao tratamento a ser dado aos valores aplicados à entrada da rede. Se considerarmos estes valores como representando uma interpretação inicial aos átomos do programa, ela não acarretará em nenhum efeito sobre resultado do cálculo da semântica de ponto fixo. Conforme mencionamos anteriormente, os resultados de Gelfond e Lifschitz (1988) indicam que aplicações recursivas do operador de conseqüência imediata convergem para o ponto fixo, independentemente da atribuição inicial de valores aos átomos. Considerando o programa usado como exemplo no capítulo anterior ($\{A \leftarrow B,C,\ B \leftarrow D, \sim E,\ C \leftarrow F,\ C \leftarrow G\}$), podemos notar que uma seqüência de aplicações do operador $\mathcal{T}_\mathcal{P}$ converge sempre para o ponto onde todos os átomos são falsos. Usando a notação de conjunto dos átomos associados ao valor verdadeiro para representar uma interpretação, a tabela 6.1 mostra estas aplicações sobre três interpretações iniciais I_0 diferentes.

Tabela 6.1: Exemplos de aplicações consecutivas de $\mathcal{T}_\mathcal{P}$

$I_0 = \{D,F\}$	$I_0 = \{D,E,G\}$	$I_0 = \{A,D,E,G\}$
$\mathcal{T}_\mathcal{P}(\{D,F\}) = \{B,C\}$	$\mathcal{T}_\mathcal{P}(\{D,E,G\}) = \{C\}$	$\mathcal{T}_\mathcal{P}(\{A,D,E,G\}) = \{C\}$
$\mathcal{T}_\mathcal{P}(\{B,C\}) = \{A\}$	$\mathcal{T}_\mathcal{P}(\{C\}) = \{\}$	$\mathcal{T}_\mathcal{P}(\{C\}) = \{\}$
$\mathcal{T}_\mathcal{P}(\{A\}) = \{\}$	$\mathcal{T}_\mathcal{P}(\{\}) = \{\}$	$\mathcal{T}_\mathcal{P}(\{\}) = \{\}$

Considerando que as redes neurais apresentam um enfoque voltado ao mapeamento de entrada para saída, um sistema baseado na integração das abordagens lógicas e neurais precisam considerar um diferente tratamento para os valores aplicados à entrada. Analisando sob o ponto de vista lógico, valores positivos nestas entradas deveriam ser considerados como fatos pelo sistema, isto é, informações que devem ser verdadeiras durante toda a computação da semântica do programa. A tabela 6.2 repete a análise anterior considerando esta abordagem, onde os átomos marcados em negrito são aqueles que denotam a informação de entrada mantidas no decorrer do processamento.

Tabela 6.2: Exemplos de aplicações consecutivas de $\mathcal{T}_\mathcal{P}$ com interpretação fixa

$I_0 = \{\mathbf{D}, \mathbf{F}\}$ $\mathcal{T}_\mathcal{P}(\{\mathbf{D}, \mathbf{F}\}) = \{B, C, \mathbf{D}, \mathbf{F}\}$ $\mathcal{T}_\mathcal{P}(\{B, C, \mathbf{D}, \mathbf{F}\}) = \{A, B, C, \mathbf{D}, \mathbf{F}\}$ $\mathcal{T}_\mathcal{P}(\{A, B, C, \mathbf{D}, \mathbf{F}\}) = \{A, B, C, \mathbf{D}, \mathbf{F}\}$	
$I_0 = \{\mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_\mathcal{P}(\{D, E, G\}) = \{C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_\mathcal{P}(\{C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$	$I_0 = \{\mathbf{A}, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_\mathcal{P}(\{\mathbf{A}, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{\mathbf{A}, C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_\mathcal{P}(\{\mathbf{A}, C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{\mathbf{A}, C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$

Uma das conseqüências negativas desta caracterização dos programas lógicos pode ser percebida no cálculo da semântica de ponto fixo para alguns casos de programas aceitáveis não acíclicos. Considere o programa proposicional $\mathcal{P}$ definido pelas cláusulas $C \leftarrow B$, A e $A \leftarrow C$, A. Considerando o modelo $\varnothing$ (A, B e C atribuídos a falso), e o mapeamento de nível definido por $\{B = 1, C = 2, A = 3\}$, temos que $\mathcal{P}$ é aceitável, e que aplicações sucessivas de $\mathcal{T}_\mathcal{P}$ sobre qualquer interpretação inicial levam a um ponto fixo, que é o próprio modelo $\varnothing$. Por outro lado, se considerarmos a atribuição de um valor positivo à variável B, que pode ser tratado logicamente como a inserção de um fato $B \leftarrow$, podemos notar que o programa deixa de ser aceitável. Inclusive, neste caso, o operador $\mathcal{T}_\mathcal{P}$ não apresenta nenhum ponto fixo, como pode ser visto na figura 6.1. Vale lembrar que este programa hipotético, mesmo sem muito sentido real, ainda pode ser considerável *aceitável* pela definição de D'Avila Garcez e Zaverucha (1999).

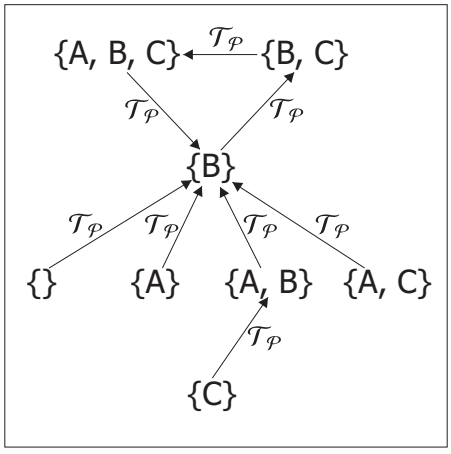

Figura 6.1: $\mathcal{T}_\mathcal{P}$ para um programa aceitável

Por este motivo, restringimos nossas análises e desenvolvimentos para a classe dos programas acíclicos. Se um programa é acíclico com relação a um mapeamento de nível específico, este mesmo mapeamento de nível pode ser mantido para verificar a propriedade acíclica do programa, independentemente da inserção de novos fatos, tendo em vista que fatos não apresentam nenhum átomo no corpo.

Voltando ao exemplo anterior, se traduzirmos o programa para uma rede neural através do algoritmo CILP, considerando que todo átomo pode receber valor atribuído externamente (ou seja, todo átomo apresenta um neurônio de entrada o representando) geraremos uma rede como na figura 6.2. Considerando novas aplicações das mesmas interpretações na entrada, obteremos na saída da rede o resultado como descrito na tabela 6.3, onde $\mathcal{T}_\mathcal{P}*$ consiste na transformação de entrada para saída realizada pela rede, e os valores em

negrito são aqueles aplicados na entrada da mesma.

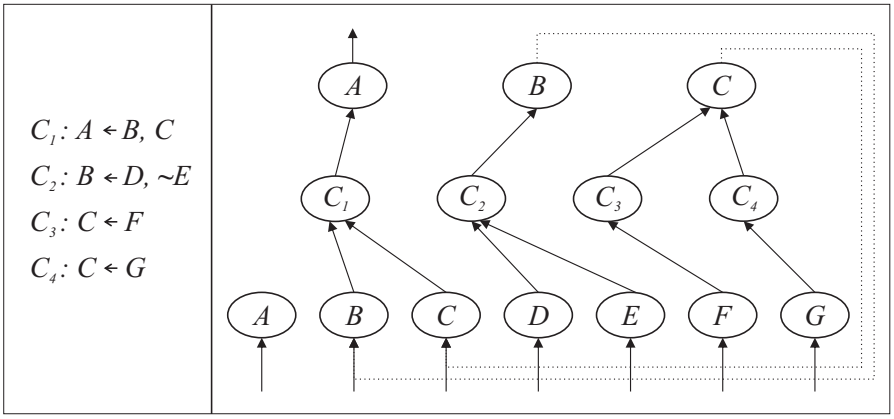

Figura 6.2: Arquitetura gerada pelo CILP

Tabela 6.3: Exemplos de aplicações consecutivas de $\mathcal{T}_{\mathcal{P}}$ representado em rede neural CILP

$I_0 = \{\mathbf{D}, \mathbf{F}\}$ $\mathcal{T}_{\mathcal{P}} * (\{\mathbf{D}, \mathbf{F}\}) = \{B, C, \}$ $\mathcal{T}_{\mathcal{P}} * (\{B, C, \mathbf{D}, \mathbf{F}\}) = \{A, B, C\}$ $\mathcal{T}_{\mathcal{P}} * (\{A, B, C, \mathbf{D}, \mathbf{F}\}) = \{A, B, C\}$	
$I_0 = \{\mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_{\mathcal{P}} * (\{\mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{C\}$ $\mathcal{T}_{\mathcal{P}} * (\{C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{C\}$	$I_0 = \{\mathbf{A}, \mathbf{D}, \mathbf{E}, \mathbf{G}\}$ $\mathcal{T}_{\mathcal{P}} * (\{\mathbf{A}, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{C\}$ $\mathcal{T}_{\mathcal{P}} * (\{\mathbf{A}, C, \mathbf{D}, \mathbf{E}, \mathbf{G}\}) = \{C\}$

Podemos notar que, no último exemplo, os valores finais de saída não refletem a conseqüência imediata dos valores aplicados na entrada. Isto acontece porque o valor aplicado a um neurônio de entrada representando um átomo qualquer não causa nenhum efeito sobre o valor do neurônio que representa o mesmo átomo na saída rede. Pela ausência de ligações entre estes átomos, chamamos este problema de *missing links issue*.

Antes de definir uma solução, um outro detalhe precisa ser considerado. Não existe nenhuma descrição explícita do tratamento a ser dado quando um neurônio de entrada recebe dois valores distintos, um valor externo e um valor recorrente. Considerando que programas em lógica utilizam negação 'default', isto é, a negação de um átomo representa que não existe nenhuma atribuição positiva a ele, temos que sempre que qualquer um dos valores for positivo, o neurônio deveria ser ativado.

Para implementar a solução, precisamos considerar que mais informações laterais do problema devem ser dadas ao algoritmo de tradução. Consideraremos que toda vez que o valor de um átomo A for atribuído externamente, este será marcado com um flag $Input(A)$. A estratégia proposta para solucionar o 'missing links issue' consiste em separar a representação de um átomo em dois neurônios diferentes, quando necessário, sendo um deles para representar a informação recorrente da saída e o outro para tratar a informação recebida externamente. Para atingir o comportamento esperado, consideramos a inserção de uma cláusula na forma $A \leftarrow A'$, sempre que um átomo A aparece em ambas situações. Desta forma o átomo A' seria inserido no programa unicamente para a geração, pelo algoritmo de tradução, de um neurônio de entrada responsável por receber o valor externo.

Desta forma, esta cláusula seria suficiente para garantir que o neurônio de saída representando um átomo A apresente o valor de sua atribuição de entrada, quando positivo.

Além do tratamento das entradas da rede, outra importante questão diz respeito ao controle das recorrentes etapas *feed forward* da rede. Considerando a definição da semântica de ponto fixo de um programa lógico, temos que diversas execuções do operador de conseqüência imediata podem ser necessários para atingir o modelo estável. Como cada execução de $\mathcal{T}_\mathcal{P}$ é representada por meio de uma propagação *feed forward* na rede gerada, temos que várias etapas de propagação podem ser necessárias, considerando a mesmo padrão de entrada aplicado à rede.

Desta forma, um mecanismo para controle de execuções da rede se faz necessário para garantir a correção do modelo. Uma proposta para tal mecanismo consiste em definir uma constante $v_\mathcal{P}$, para cada programa $\mathcal{P}$, de forma que $v_\mathcal{P}$ execuções do operador $\mathcal{T}_\mathcal{P}$ sejam suficientes para garantir a convergência para o ponto fixo. Desta forma, cada vez que um padrão de entrada for aplicado à rede, esta deve ser executada $v_\mathcal{P}$ vezes antes da aplicação de um novo conjunto de valores. Para definir o valor de $v_\mathcal{P}$ para programas acíclicos, consideramos a definição de uma constante v_A para cada átomo do programa, de forma que v_A execuções do operador $\mathcal{T}_\mathcal{P}$ sejam suficientes para atingir a interpretação estável para o átomo A. O valor de v_A é definido da seguinte forma:

- $v_A = 0$ se A não aparece como cabeça em nenhuma cláusula, ou se aparece como cabeça de cláusula com corpo vazio;
- $v_A = max(v_B) + 1$ se A aparece como cabeça em alguma cláusula, e $max(v_B)$ é o maior valor de v_B entre os átomos que aparecem como corpo nas cláusulas onde A é a cabeça.

No nosso exemplo acima, temos que os valores de v_D, v_E, v_F e v_G são iguais a zero, pois nenhum dos átomos aparecem como saída. Logo, temos que v_B e v_C são iguais a 1, e v_A é igual a 2. Mesmo com a inserção da cláusula $A \leftarrow A'$, o valor de v_A permanece o mesmo, porque $v_{A'}$ é igual a 0, então B e C continuam sendo átomos com maior valor de v no corpo de cláusulas em que A é a cabeça. Como v_A é o maior dos valores, temos que $v_\mathcal{P} = v_A = 2$. Conforme vimos nas diferentes tabelas acima, duas execuções de $\mathcal{T}_\mathcal{P}$ foram suficientes para atingir o modelo estável do programa.

Lema 6.1 *Considerando um programa acíclico $\mathcal{P}$, e uma constante $v_\mathcal{P}$ calculada conforme descrito acima, $v_\mathcal{P}$ execuções do operador de conseqüência imediata $\mathcal{T}_\mathcal{P}$ sobre qualquer interpretação inicial são suficientes para realizar a computação da semântica de ponto fixo.*

Prova: Para verificação deste lema, podemos considerar o mesmo processo de indução realizada na prova do teorema 2.9. Como base da indução, consideramos os átomos A que não aparecem como cabeça em nenhuma cláusula, ou que aparecem como cabeça de cláusula com corpo vazio, que manterão uma interpretação constante no decorrer do cálculo do ponto fixo, e portanto nenhuma execução de $\mathcal{T}_\mathcal{P}$ é necessária para atingir a convergência destes átomos. Para o passo da indução, se considerarmos a condição após $max(v_B)$ execuções de $\mathcal{T}_\mathcal{P}$, temos que todos os átomos B, tais que B que aparecem como corpo das cláusulas em que um átomo A é cabeça, atingiram o ponto de convergência, e portanto apenas mais uma execução de $\mathcal{T}_\mathcal{P}$ é necessária para atingir a convergência de A. □

6.2 Propagação Temporal

Conforme descrito anteriormente, o modelo de representação temporal utilizado neste trabalho é baseado em uma abordagem seqüencial, onde o conhecimento sobre o passado é usado para inferir novas informações sobre o futuro. Nossa estratégia para representar conhecimento temporal, segundo esta abordagem, se baseia na propagação de valores entre consecutivos momentos, seguindo o fluxo de tempo. Como as definições semânticas relacionadas na seção 2.4 são baseadas no tempo imediatamente anterior ($t-1$), um primeiro passo para a representação desta semântica consiste em representar a integração dos valores em $t-1$ no processamento da rede, e conseqüentemente representar o operador $\bullet$, que representa estes valores.

Como mecanismo para propagação de informações através do tempo, optamos pelo uso de unidades de atraso, cujo comportamento é diretamente associado à abordagem escolhida. Buscando utilizar modelos de redes MLP para representação de conhecimento e inserir unidades de atraso para lidar com os aspectos temporais, escolhemos o modelo neural NARX. Neste modelo, a seqüência temporal é representada pela aplicação consecutiva de valores de entrada na rede. Cada momento no tempo é caracterizado por esta aplicação de valores e consequente processamento *feed forward* e cálculo do vetor de saída pela rede MLP. Neste modelo, dois tipos de unidades de atraso podem ser utilizadas, antes de um neurônio de entrada da rede: Unidades em ligações recorrentes e unidades recebendo valor diretamente da entrada.

O uso destas unidades de atraso permitem uma representação adequada para a propagação de valor de um momento $t-1$ para um momento t, e desta forma, permite que um neurônio representando um átomo $\bullet\alpha$ receba o valor corretamente computado para α no momento anterior, representando corretamente a semântica do operador $\bullet$. No decorrer do trabalho, diferentes abordagens foram consideradas para esta representação.

A primeira abordagem consiste na utilização de ligações recursivas, unicamente. Esta abordagem é intuitivamente apropriada para a aplicação do valor associado a um átomo $\bullet\alpha$ sempre que α aparece como cabeça em alguma cláusula no programa original. Caso isto não aconteça, a rede gerada não apresenta um neurônio de saída corretamente representando α e, desta forma, o valor de $\bullet\alpha$ também não será corretamente aplicado. Uma solução para este problema consiste em estender a solução proposta para o *missing links issue* de forma que uma cláusula $\alpha \leftarrow \alpha'$ também seja inserida sempre que um átomo $\bullet\alpha$ apareça num programa $\mathcal{P}$, e α não seja cabeça de nenhuma cláusula.

Note que, neste caso, o valor de $v_{\mathcal{P}}$ pode ser incrementado, tendo em vista que uma cláusula é inserida e a cabeça desta cláusula aparece como corpo em outra. Nossa segunda abordagem para a representação do operador $\bullet$ consiste num procedimento semelhante à primeira, porém sempre que uma cláusula $\alpha \leftarrow \alpha'$ for inserida para um átomo α que não aparece como cabeça em nenhuma cláusula do programa original, todas as ocorrências de α no corpo de outras cláusulas sejam renomeadas para α'. Desta forma, mesmo que uma cláusula seja inserida, o átomo que aparece como cabeça não será usado como corpo de nenhuma outra cláusula, e o valor de v_P não será incrementado.

Nossa terceira abordagem consiste em explorar melhor os recursos das arquiteturas NARX para reduzir o número total de neurônios na rede. Esta abordagem utiliza a inserção de unidades de atraso para em um neurônio de entrada para aplicar o valor de $\bullet\alpha$ sempre que α não aparece como cabeça de nenhuma cláusula no programa original, e desta forma evitando a inserção de cláusulas $\alpha \leftarrow \alpha'$ cujo objetivo seja meramente a propagação temporal. Portanto, para cada átomo na forma $\bullet^n\alpha$, podemos considerar duas formas de inserir unidades de atraso para a representação no modelo conexionista:

- Se existe um átomo na forma $\bullet^i\alpha$ que apareça como cabeça de uma cláusula (no programa original), sendo que $0 \leq i < n$, cria-se uma ligação recursiva entre o neurônio de saída representando o átomo $\bullet^{max(i)}\alpha$, onde $max(i)$ é o maior valor de i, e o neurônio de entrada representando $\bullet^n\alpha$, tal que esta ligação apresenta $n-i$ unidades de atraso.

- Se nenhum átomo nesta forma aparece como cabeça de nenhuma cláusula no programa, adiciona-se n unidades de atraso antes do neurônio de entrada representando $\bullet^n\alpha$, para receber o valor de α aplicado à entrada no ponto $t-n$ do tempo.

No primeiro experimento do próximo capítulo, ilustramos um exemplo de construção de arquiteturas utilizando as três abordagens mencionadas. Apresentamos, na figura 6.3, um algoritmo para tradução de programas lógicos para redes no modelo NARX, de forma a preservar a semântica do operador $\bullet$.

```
●-based_Translation(P)
    CorrecaoMissingLinks(P);
    N ← TraducaoCILP(P);
    para cada in_α ∈ Neuronios(N) faça
        se (α = ●^n β) então
            se ∃i < n(out_●^i β ∈ neurons(N)) então
                j ← maximum(i);
                InsereLigacaoAtrasada(N, n − j, out_●^j β, in_α);
            senão
                InsereEntradaAtrasada(N, n, in_α);
    fim
    return N;
end
```

Figura 6.3: Tradução de programas baseados no operador $\bullet$

Lema 6.2 *Uma rede neural $\mathcal{N}$ gerada através da aplicação do algoritmo de tradução $\bullet$-based_Translation sobre um programa temporal acíclico $\mathcal{P}$ computa o operador de conseqüência imediata $\bullet\mathcal{T}_\mathcal{P}$ de $\mathcal{P}$.*

Prova: Podemos provar esta afirmação de forma indutiva. Para um primeiro ponto no tempo (t = 1), dadas as devidas inicializações para átomos $\bullet\alpha$, temos que a computação do operador $\bullet\mathcal{T}_\mathcal{P}$ é semelhante à do operador $\mathcal{T}_\mathcal{P}$ tradicional. Além disso, dada a convergência de $\mathcal{T}_\mathcal{P}$ para programas acíclicos, o valor do ponto fixo também é calculado corretamente. Para o passo indutivo, podemos considerar que, para todos os momentos t' tais que $t' < t$ e para todo átomo α, ou a rede $\mathcal{N}$ apresenta um neurônio de saída out_α representando corretamente $\mathcal{F}_\mathcal{P}^{t'}(\alpha)$, ou valor de α é dado pela entrada. Portanto, para cada átomo $\bullet^n\alpha$ em t, se um valor de $\bullet^i\alpha (i < n)$ aparece representado na saída da rede, a ligação recorrente com $n-i$ unidades de atraso propaga o valor correto para o neurônio $in_{\bullet^n\alpha}$. Caso contrário, ou seja, se $\bullet^i\alpha$ não é representado na saída da rede para nenhum $i < n$, o valor de entrada do neurônio $in_{\bullet^n\alpha}$ é fornecido corretamente pela cadeia de n unidades de atraso recebendo de entrada de α. Dada a convergência de $\bullet\mathcal{T}_\mathcal{P}$, temos que a rede atinge o ponto fixo, para possibilitar a propagação correta para os próximos pontos no tempo. □

6.3 Representação temporal

A estratégia definida na última seção define a utilização de recursos da arquitetura neural para propagação de valores através do tempo, e computação do operador de conseqüência imediata $\bullet\mathcal{T}_\mathcal{P}$. Para estender este sistema e permitir a representação semântica dos demais operadores temporais, nossa estratégia consiste em inserir cláusulas no programa original de forma que átomos baseados nos demais operadores temporais possam apresentar sua semântica definida em função de informações sobre o presente ou sobre outros átomos na forma $\bullet\alpha$.

Lema 6.3 *Considerando um programa temporal $\mathcal{P}_1$, gerado através da inserção de cláusulas para cada operador temporal (figura 6.5) em um programa original $\mathcal{P}$, temos que, para todo átomo α de $\mathcal{P}$, $\mathcal{ST}_\mathcal{P}(I^t_\mathcal{P})(\alpha)$ é verdadeiro se, e somente se, $\bullet\mathcal{T}_{\mathcal{P}1}(I^t_\mathcal{P})(\alpha)$ também for verdadeiro.*

Prova: A tradução realizada pelo algoritmo somente insere cláusulas representando exatamente as definições semânticas dos operadores. Podemos verificar isso através da análise individual de todas as cláusulas inseridas. Citando, por exemplo, o caso do operador $\mathbb{S}$, temos que a primeira cláusula inserida ($\beta\mathbb{S}\gamma \leftarrow \gamma$) representa exatamente a primeira definição do item *e.* da definição 2.13 e, tendo em vista que o átomo $\bullet\alpha$ representa corretamente a informação sobre a semântica do átomo α no momento $t-1$, a cláusula $\beta\mathbb{S}\gamma \leftarrow \beta, \bullet(\beta\mathbb{S}\gamma)$ representa corretamente a segunda opção da definição formal de $\mathbb{S}$.

($\rightarrow$)Assumindo que $\mathcal{ST}_\mathcal{P}(I^t_\mathcal{P})(\alpha)$ seja verdadeiro, temos duas possibilidades. Se $\bullet\mathcal{T}_\mathcal{P}(I^t_\mathcal{P})(\alpha)$ for verdadeiro, então a inserção de novas cláusulas não altera este fato, e $\bullet\mathcal{T}_{\mathcal{P}1}(I^t_\mathcal{P})(\alpha)$ também será verdadeiro. Caso contrário, a interpretação positiva de α é definida pela regra semântica de um dos operadores, e neste caso uma cláusula representando este operador será inserida pelo algoritmo, e a interpretação da conjunção dos literais no corpo desta cláusula será verdadeiro, portanto $\bullet\mathcal{T}_{\mathcal{P}1}(I^t_\mathcal{P})(\alpha)$ também será verdadeiro.

($\leftarrow$) Se $\mathcal{ST}_\mathcal{P}(I^t_\mathcal{P})(\alpha)$ for falso, então α não é interpretado como verdadeiro devido à semântica de nenhum operador temporal. Logo, nenhuma as cláusulas inseridas pelo algoritmo causam efeito à interpretação de α, e portanto $\bullet\mathcal{T}_{\mathcal{P}1}(I^t_\mathcal{P})(\alpha)$ também será falso.□

Teorema 6.4 *Para um programa temporal acíclico $\mathcal{P}$, e um programa temporal $\mathcal{P}_1$ = Tratamento_Logico($\mathcal{P}$), temos que a seguinte implicação é válida: Se $\mathcal{P}_1$ é acíclico, então uma rede neural $\mathcal{N}$ = $\bullet$-based_Translation($\mathcal{P}_1$) computa, em $v_{\mathcal{P}1}$ execuções, o ponto fixo do operador $\mathcal{ST}_\mathcal{P}$ de $\mathcal{P}$.*

Prova: A prova deste teorema segue diretamente dos lemas 2.12, 6.1, 6.2 e 6.3. Note que a condição de $\mathcal{P}_1$ ser acíclico é necessária pelo fato que a inserção de cláusulas para representação dos operadores temporais pode causar violação na natureza acíclica do programa, o que invalidaria os demais resultados mostrados. □

6.4 Comparação entre SCTL e CTLK

No capítulo 5 mostramos diferentes extensões do CILP para representação de conhecimento expresso em lógicas não clássicas. Mais especificamente, ilustramos o caso do sistema CTLK, que apresenta uma abordagem para representação de lógicas temporais e epistêmicas. Podemos apontar diversas diferenças com relação a esta abordagem para representação temporal e o uso do SCTL.

A primeira diferença claramente percebida consiste na estratégia para representação do fluxo de tempo. Em nosso trabalho, buscamos uma representação serial do fluxo de tempo, considerando uma única rede, incrementada com unidades de atraso, para permitir a propagação de informações através do tempo. Esta representação é baseada nume representação única para todos os diferentes estados de tempo, não havendo limites para o número de pontos no tempo a serem representados. Além disto, o uso de uma única rede acarreta em uma complexidade menor do sistema como um todo, isto é, um número menor de neurônios para uma representação.

Por outro lado, o sistema CTLK considera uma representação paralela do tempo, onde cada ponto na linha do tempo é referenciado por um rótulo e apresenta uma rede neural específica para sua representação. Uma das grandes vantagens desta estratégia é a identificação de cada cláusula com um ponto de tempo específico, enquanto as cláusulas no SCTL são consideradas independentemente do momento. Uma solução para representar este tipo de identificação num sistema SCTL consiste em usar um átomo t^* específico para marcar um ponto t no tempo. Desta forma, para representar uma semântica similar a uma cláusula rotulada na forma $t : \alpha \leftarrow \lambda_1, ..., \lambda_n$, consideraríamos o uso de uma cláusula na forma $\alpha \leftarrow t^*, \lambda_1, ..., \lambda_n$.

Outra vantagem da representação paralela utilizada pelo CTLK está na representatividade do sistema, que permite inferências temporais tanto do passado para o futuro (como no sistema SCTL) quanto no sentido contrário. Considerando, por exemplo, um programa lógico onde as cláusulas $\{\bigcirc B \leftarrow A, C \leftarrow B, D \leftarrow \bigcirc C\}$ são consideradas em dois pontos de tempo consecutivos t_1 e t_2, podemos verificar que, se A for verdadeiro em t_1, o átomo D também será interpretado positivamente, graças a inferências nos dois sentidos.

Considerando uma representação SCTL deste programa, temos que todas as cláusulas são consideradas em todos os momentos no tempo. Porém, considerando A como verdadeiro no momento t_1, temos que $\bigcirc B$ é verdadeiro. Porém, durante a computação de t_1, o sistema não apresenta recursos para considerar a cláusula $C \leftarrow B$ em t_2, então o átomo D será interpretado negativamente. Uma opção para contornar este problema no sistema consiste na inserção de cláusulas específicas para a realização destas inferências. Neste caso, se inserirmos uma cláusula $\bigcirc C \leftarrow \bigcirc B$, notamos que o sistema executa a mesma inferência realizada pelo CTLK.

Por fim, outra importante diferença com relação às duas abordagens consiste na gama de operadores representáveis. O sistema CTLK provê representação para programas lógicos baseados unicamente no uso do operador de próximo momento no futuro ($\bigcirc$). Já o sistema SCTL apresenta uma estratégia comprovadamente correta para representação da semântica de um conjunto considerável de operadores temporais, tanto para representação do passado quanto para representação do futuro.

Vale lembrar, porém, que a estratégia utilizada pelo sistema CTLK para representação do operador $\bigcirc$, que consiste em ligações entre neurônios de uma rede representando o tempo t para uma rede representando $t+1$, intuitivamente pode ser estendido para representar também os operador de passado $\bullet$, mantendo a mesma idéia original do sistema. Desta forma, considerando uma definição indutiva dos demais operadores lógicos temporais, de mesma forma que utilizado pelo SCTL, é possível estender o sistema temporal de representação do modelo CTLK para a representação do mesmo conjunto de operadores descrito neste trabalho.

Para ilustrar melhor a relação entre estas diferentes abordagens para representação de tempo, vamos considerar um exemplo descrito por Fagin et al. (1995), e utilizado nos trabalhos de D'Avila Garcez et al. (2004) e de D'Avila Garcez e Lamb (2006) para

ilustrar a representação de conhecimento usando lógicas não clássicas, que consiste na representação do raciocínio de um agente para o *Muddy Children Puzzle*. Este exemplo pode ser descrito da seguinte forma (D'AVILA GARCEZ; LAMB, 2006, p.18):

> Um número n de crianças (confiáveis e inteligentes) estão brincando em um jardim. Um certo número k destas crianças ($k \leq n$) tem lama em seus rostos. Cada criança consegue ver se o rosto dos outros estão sujos, mas não o seu próprio. Agora, considere a seguinte situação: A pessoa que está tomando conta das crianças anuncia que ao menos uma delas está suja, e pergunta se alguma delas sabe se tem lama em seu próprio rosto.
>
> Para compreensão do puzzle, vamos considerar os casos em que $k = 1$, $k = 2$ e $k = 3$. Se $k = 1$, a criança que está suja responde no primeiro momento que sim, tendo em vista que ela não consegue ver nenhuma outra criança com lama. Todas a outras crianças respondem não. Se $k = 2$, suponha que as crianças 1 e 2 estão sujas. Num primeiro momento, todas as crianças só podem responder não. Isto faz com que 1 pense da seguinte forma: se 2 tivesse dito sim no primeiro momento, ela seria a única criança com lama. Como 2 disse não, ela deve estar vendo mais alguém sujo, e como eu não estou vendo ninguém sujo além de 2, esta pessoa deve ser eu. A criança 2 pode raciocinar de maneira análoga e também responder sim, neste segundo momento. Se $k = 3$, suponha que as crianças 1, 2 e 3 estão com lama em seus rostos. Todas as crianças só podem responder não nas primeiras duas vezes. Novamente, isto faz com que a criança 1 pense o seguinte: Se 2 ou 3 tivessem dito sim no segundo momento, eles seriam as duas únicas crianças sujas. Portanto, deve haver uma terceira pessoa com lama no rosto. Como eu só consigo ver 2 e 3 com lama, esta terceira pessoa deve ser eu. As crianças 2 e 3 podem raciocinar analogamente e concluir que elas estão com lama no rosto também.

Na tabela 6.4, ilustramos a representação do raciocínio da criança 1 em um ambiente com 3 crianças, considerando uma descrição para o sistema CTLK, de acordo com D'Avila Garcez e Lamb (2006), e uma descrição para o sistema SCTL. Mais uma vez, vale ressaltar que foge do escopo deste trabalho a formalização e representação de operadores epistêmicos, este exemplo é utilizado meramente para ilustração dos aspectos temporais relacionados. A notação p_i indica que a criança i está com lama no rosto, e q_i indica que existem i crianças com lama no rosto. Consideraremos que os átomos na forma $K_i\alpha$ indicam que o agente i sabe que α é verdadeiro. A figura 6.4 mostra a representação neural destes programas, ilustrando a considerável redução espacial da rede proporcionado pela representação seqüencial de tempo no sistema conexionista.

Tabela 6.4: Representação de um agente no 'Muddy Children Puzzle'

$K_1p_1 \leftarrow K_1q_1, K_1\neg p_2, K_1\neg p_3$	$K_1p_1 \leftarrow K_1q_1, K_1\neg p_2, \sim K_1\neg p_3$
$K_1p_1 \leftarrow K_1q_2, K_1\neg p_2$	$K_1p_1 \leftarrow K_1q_2, K_1\neg p_2$
$K_1p_1 \leftarrow K_1q_2, K_1\neg p_3$	$K_1p_1 \leftarrow K_1q_2, K_1\neg p_3$
$K_1p_1 \leftarrow K_1q_3$	$K_1p_1 \leftarrow K_1q_3$
$t_1 : \bigcirc Kq_2 \leftarrow\sim K_1p_1, \sim K_2p_2, \sim K_3p_3$	$K_1q_2 \leftarrow\sim \bullet K_1p_1, \bullet K_2p_2, \bullet K_3K_3, \bullet K_1q_1$
$t_2 : \bigcirc Kq_3 \leftarrow\sim K_1p_1, \sim K_2p_2, \sim K_3p_3$	$K_1q_3 \leftarrow\sim \bullet K_1p_1, \bullet K_2p_2, \bullet K_3K_3, \bullet K_1q_2$

No próximo capítulo, mostramos diversos exemplos de representação de programas temporais baseados no sistema SCTL, mostrando a aplicabilidade de algoritmos de trei-

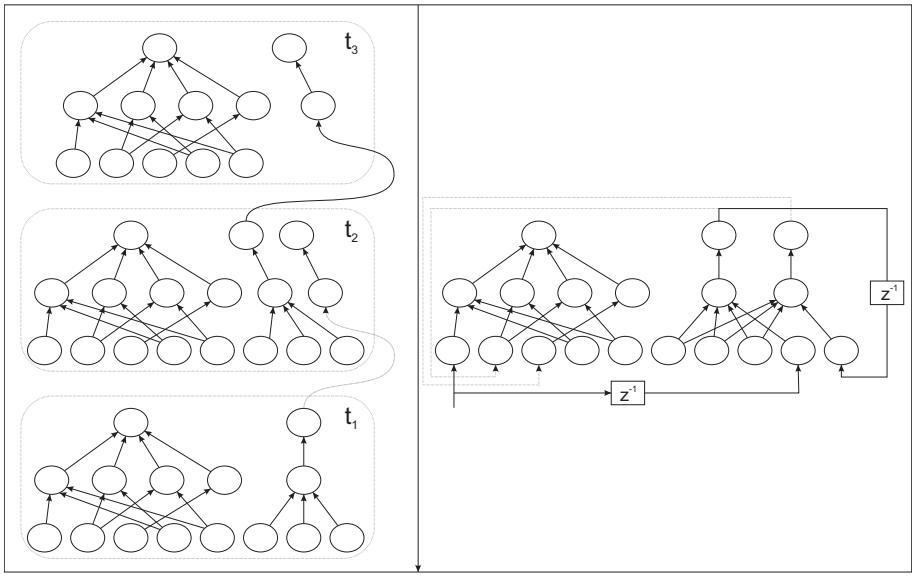

Figura 6.4: Ilustração das arquiteturas CTLK e SCTL para o Muddy Children Puzzle

namento sobre as redes geradas para realização de aprendizagem empírica. Por questão de comparação, vale lembrar que, enquanto o sistema SCTL foi definido de forma a permitir aprendizagem de aspectos temporais, sistemas como o CTLK (e outros baseados no uso de conjuntos de rede) apresentam definição de rotinas de aprendizagem aplicadas a cada rede individual no sistema, sem considerar a aprendizagem das relações entre elas. Como estas relações definem a semântica temporal dos operadores, podemos considerar que as rotinas de aprendizagem apresentadas até agora para o sistema CTLK não foram suficientemente verificadas na literatura até o momento presente.

Tratamento_Logico($\mathcal{P}$)
  para cada $\alpha \in atoms(\mathcal{P})$ **faça**
    if $\alpha = \blacksquare\beta$ **then**
      $InsereClausula(\mathcal{P}, \Box\beta \leftarrow \beta, \bullet\beta)$;
    end
    if $\alpha = \blacklozenge\beta$ **then**
      $InsereClausula(\mathcal{P}, \blacklozenge\beta \leftarrow \beta)$;
      $InsereClausula(\mathcal{P}, \blacklozenge\beta \leftarrow \bullet\blacklozenge\beta)$;
    end
    if $\alpha = \beta\mathbb{S}\gamma$ **then**
      $InsereClausula(\mathcal{P}, \beta\mathbb{S}\gamma \leftarrow \gamma)$;
      $InsereClausula(\mathcal{P}, \beta\mathbb{S}\gamma \leftarrow \beta, \bullet(\beta\mathbb{S}\gamma))$;
    end
    if $\alpha = \beta\mathbb{Z}\gamma$ **then**
      $InsereClausula(\mathcal{P}, \beta\mathbb{Z}\gamma \leftarrow \gamma)$;
      $InsereClausula(\mathcal{P}, \beta\mathbb{Z}\gamma \leftarrow \beta, \bullet(\beta\mathbb{Z}\gamma))$;
    end
    if $\alpha = \circ\beta$ **then**
      $InsereClausula(\mathcal{P}, \beta \leftarrow \bullet\circ\beta)$;
    end
    if $\alpha = \Box\beta$ **then**
      $InsereClausula(\mathcal{P}, \beta \leftarrow \Box\beta)$;
      $InsereClausula(\mathcal{P}, \Box\beta \leftarrow \bullet\Box\beta)$;
    end
    if $\alpha = \Diamond\beta$ **then**
      $InsereClausula(\mathcal{P}, \beta \leftarrow \bullet\Diamond\beta, \sim\bullet\beta)$;
    end
    if $\alpha = \beta\mathbb{U}\gamma$ **then**
      $InsereClausula(\mathcal{P}, \beta\mathbb{U}\gamma \leftarrow \bullet(\beta\mathbb{U}\gamma), \sim\bullet(\gamma))$;
      $InsereClausula(\mathcal{P}, \gamma \leftarrow \beta\mathbb{U}\gamma, \sim\beta)$;
    end
    if $\alpha = \beta\mathbb{W}\gamma$ **then**
      $InsereClausula(\mathcal{P}, \beta\mathbb{W}\gamma \leftarrow \bullet(\beta\mathbb{W}\gamma), \sim\bullet(\gamma))$;
      $InsereClausula(\mathcal{P}, \beta \leftarrow \beta\mathbb{W}\gamma, \sim\gamma)$;
    end
  fim
end

Figura 6.5: Conversão Lógica

7 APLICAÇÕES E EXEMPLOS

7.1 Algoritmo de Aprendizagem

Neste capítulo, ilustraremos vários casos de teste sobre o sistema gerado para ilustrar os efeitos da aplicação de algoritmos de treinamento sobre as arquiteturas geradas no modelo SCTL. O algoritmo de aprendizagem utilizado é baseado no Backpropagation (RUMELHART; HINTON; WILLIAMS, 1986) e em diferentes extensões para redes temporais (ELMAN, 1990; WAIBEL et al., 1990; WERBOS, 1990).

Devemos ressaltar que, durante a computação da semântica de ponto fixo, o comportamento recorrente no modelo é definido para a retropropagação do erro da mesma forma que para sua etapa feedforward, ou seja, para um mesmo erro aplicado na saída, $\nu_{\mathcal{P}}$ retropropagações serão consideradas. Além disso, as unidades de atraso colocadas em ligações recorrentes realizarão uma retropropagação do último valor de δ, no ponto de tempo anterior, calculado no neurônio de entrada a qual está conectada.

O algoritmo descrito em 7.1 descreve detalhadamente o comportamento da rede durante o treinamento. Três tipos de unidades de entrada são considerados: $Input(I)$, que indica que o valor de entrada do neurônio é recebido externamente, $Contexto(I)$ que indica que o neurônio recebe informação de uma ligação recorrente sem atrasos e $Temporal(I)$ que indica que o neurônio recebe uma entrada atrasada. Da mesma forma, três flags são usados para neurônios de saída: $Output(O)$, $Contexto(O)$ e $Temporal(O)$, indicando que o valor de saída é utilizado como saída da rede, realimentação recorrente ou realimentação temporal, respectivamente.

Outras notações utilizadas para o algoritmo são:

- $EntradaRede(I)$: valor de entrada aplicado à rede, associado a um neurônio de entrada I;
- $SaidaRede(O)$: valor esperado na saída da rede, associado a um neurônio O;
- $Erro(O)$: valor de erro no neurônio de saída O, usado para o Backpropagation;
- $ValorRecorrente(I)$: último valor de ativação da unidade se saída conectada recorrentemente na entrada de I;
- $DeltaRecorrente(O)$: último valor de δ da unidade de entrada conectada recorrentemente à saída de O;
- $ValorAtrasado(I)$: valor armazenado na unidade de atraso conectada a I

- *DeltaAtrasado*(O): valor de δ no neurônio de entrada conectado a uma unidade de atraso (ou cadeia de unidades de atraso) em ligação recorrente partindo de O, calculado com os devidos atrasos;

- $Feedforward(\mathcal{N})$: realiza propagação feedforward tradicional de valores na rede $\mathcal{P}$;

- *Backprop*($\mathcal{N}$): Executa a correção de erros de acordo com o algoritmo Backpropagation tradicional, de acordo com o erro nos neurônios de saída;

- $CorrigeAtrasos(\mathcal{N})$: Atualiza os valores de ativação e δ a serem propagados pelas unidades de atraso da rede $\mathcal{N}$ nos próximos estados de tempo.

Execução da rede $\mathcal{N}$
  para cada *ponto no tempo* **faça**
    para cada $I \in NeuroniosEntrada(\mathcal{N})$ **faça**
      se $Input(I)$ **então** $Entrada(I) = EntradaRede(I)$;
      se $Contexto(I)$ **então** $Entrada(I) = -1$;
      se $Temporal(I)$ **então** $Entrada(I) = ValorAtrasado(I)$;
    fim
    $Feedforward(\mathcal{N})$;
    repita
      para cada $I \in NeuroniosEntrada(\mathcal{N})$ **faça**
        se $Input(I)$ **então** $Entrada(I) = EntradaRede(I)$;
        se $Contexto(I)$ **então** $Entrada(I) = ValorRecorrente(I)$;
        se $Temporal(I)$ **então** $Entrada(I) = ValorAtrasado(I)$;
      fim
      $Feedforward(\mathcal{N})$;
    até $(v_{\mathcal{P}} - 1)$ *vezes* ;
    para cada $O \in NeuroniosSaida(\mathcal{N})$ **faça**
      se $Output(O)$ **então** $Erro(O) = SaidaRede(O) - Ativacao(O)$;
      se $Contexto(O)$ **então** $Erro(O) = 0$;
      se $Temporal(O)$ **então** $Erro(O) = DeltaAtrasado(O)$;
    fim
    $Backprop(\mathcal{N})$;
    repita
      para cada $O \in NeuroniosSaida(\mathcal{N})$ **faça**
        se $Output(O)$ **então** $Erro(O) = 0$;
        se $Contexto(O)$ **então** $Erro(O) = DeltaRecorrente(O)$;
        se $Temporal(O)$ **então** $Erro(O) = 0$;
      fim
      $Backprop(\mathcal{N})$;
    até $(v_{\mathcal{P}} - 1)$ *vezes* ;
    $CorrigeAtrasos(\mathcal{N})$
  fim
end

Figura 7.1: Descrição do treinamento das redes SCTL

7.2 XOR temporal

O primeiro exemplo consiste na variação temporal do problema do 'ou exclusivo' (XOR) proposto por Elman (1990). O reconhecimento da função XOR é um problema clássico na área de redes neurais, desde o trabalho de Minsky e Papert (1969), que utilizaram este exemplo para provar que o perceptron elementar era incapaz de identificar classes não linearmente separáveis. A variação temporal proposta por Elman consiste no reconhecimento do próximo valor em uma seqüência temporal de bits, formada por subseqüências compostas por dois bits aleatórios e um terceiro definido pela aplicação da XOR sobre os dois primeiros. Abaixo, mostramos a tabela verdade da função XOR e um exemplo de série da XOR temporal.

Tabela 7.1: Seqüência associada ao problema do XOR temporal

in	0	0	**0**	1	0	**1**	1	1	**0**	0	1	**1**
out	0	**0**	1	0	**1**	1	1	**0**	0	1	**1**	?

Para este experimento, consideramos dois casos diferentes. No primeiro caso, o valor de erro é calculado e retropropagado na rede apenas quando o segundo bit de cada subseqüência é aplicado na entrada (ou seja, para predição do terceiro bit). Nos demais passos, o erro aplicado na saída é nulo, e desta forma a rede só realiza alteração dos pesos nos casos em que a informação de saída está de acordo com o objetivo da tarefa de aprendizagem. Num segundo caso, consideraremos que o erro de saída é retropropagado para todos os bits da seqüência, e desta forma a rede receberá dados com um ruído considerável para a aprendizagem (aprox. 33.3%).

Para cada experimento, consideramos a representação do problema pelo programa lógico definido pelas cláusulas $O \leftarrow I, \sim \bullet I$ e $O \leftarrow \sim I, \bullet I$, traduzidos para redes neurais utilizando as três abordagens propostas na seção 6.2 para representação do operador $\bullet$. Nas duas primeiras abordagens, utilizamos duas inicializações de pesos para com relação aos neurônios inseridos para propagação temporal: Inserção dos neurônios com pesos e bias aleatórios, e inserção dos neurônios com a definição dos pesos e bias dada pelo algoritmo.

Além disso, para cada arquitetura, três inicializações diferentes dos pesos relacionados com os neurônios de representação da XOR foram consideradas: Inicialização aleatória, inicialização baseada na tradução de apenas uma cláusula (sendo o outro neurônio oculto inicializado com pesos e bias aleatórios) e inicialização com o programa completo. Na figura 7.2 ilustramos as arquiteturas utilizadas, relacionadas com as três abordagens para representação de $\bullet$.

Para verificar a aplicação de rotinas de aprendizagem sobre as diferentes redes, consideramos a utilização do processo de *10-fold cross validation*, ou seja, o conjunto de padrões foi dividido em 10 subconjuntos, de forma que para cada uma das dez etapas de validação, um destes subconjuntos é selecionado para o teste do desempenho da rede treinada através da aplicação de um algoritmo de aprendizagem empírica utilizando os demais 9 subconjuntos. Para cada arquitetura diferente, este processo foi executado duas vezes sob um conjunto de 3000 bits, ou seja, 20 validações diferentes utilizando 2700 bits para treinamento e 300 para teste. Utilizamos, para cada validação individual, 500 épocas (aplicações de todo o conjunto de dados), considerando uma taxa de aprendizagem fixa de 0.3 para os neurônios, e sem a utilização de taxa de *momentum*.

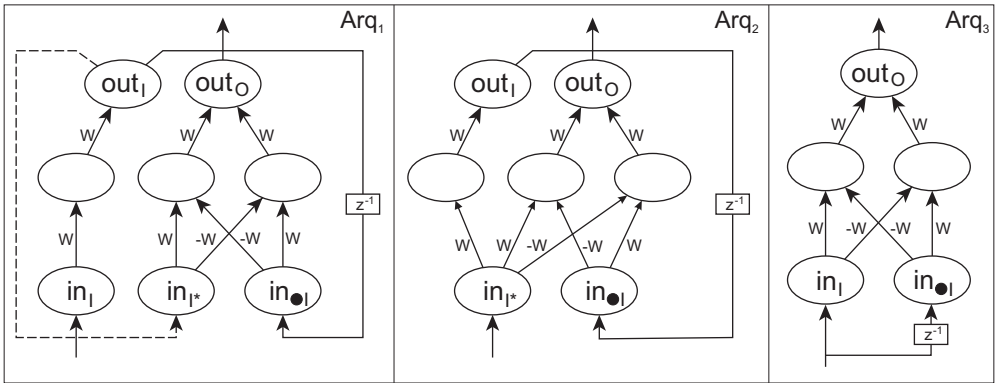

Figura 7.2: Redes utilizadas para experimento com XOR

A figura 7.3 ilustra os resultados obtidos sobre o experimento sem ruído. Neste gráfico, ilustramos a média do valor de RMSE (Root Mean Squared Error) sob o conjunto de teste. O valor de RMSE é definido como a raiz quadrada da média dos erros quadrados obtidos. Os, valores de saída da rede, definidos no intervalo [-1..1] foram normalizados para o intervalo [0..1] antes do cálculo do erro. Os rótulos Arq_1B e Arq_2B ilustram o uso das arquiteturas geradas através do programa com a cláusula $I \leftarrow I'$, enquanto as arquiteturas Arq_1A e Arq_2A consideram a inicialização aleatória dos pesos e bias associados. No gráfico, são mostrados as médias, dentre todas as validações, dos erros calculados. Para cada validação, selecionamos os erros na melhor época do treinamento (época com erro mais baixo) e sobre a última época, com o objetivo de ilustrar, também, a curva de convergência do algoritmo (se a diferença dos valores for considerável, significa que a os pesos da rede não convergiram para o melhor valor).

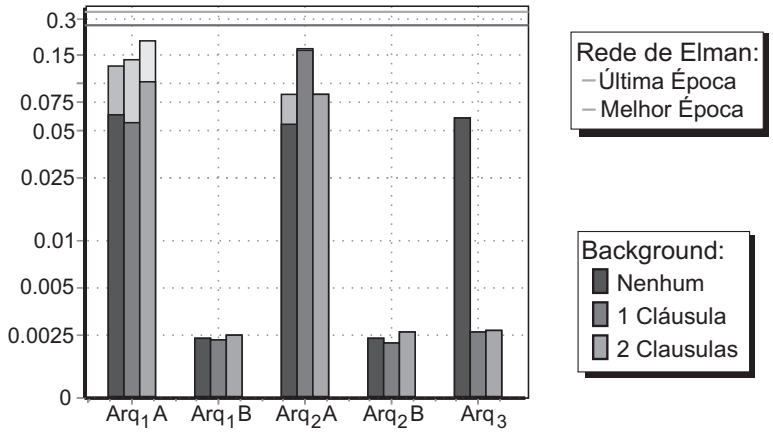

Figura 7.3: Erro das arquiteturas para experimento XOR sem ruído

A partir deste gráfico, podemos notar a influência da inserção de conhecimento temporal sobre o desempenho do sistema. Com exceção da rede gerada através da terceira abordagem, todas as redes inicializadas com a representação correta da propagação temporal de valores apresentaram um valor de erro desprezível. Outro aspecto importante consiste na considerável diferença entre o desempenho deste sistema e o desempenho obtida por experimentos semelhantes realizadas sobre a rede de Elman. Os resultados da rede de Elman estão ilustrados pelas linhas na parte superior da tabela. Vale lembrar que os procedimentos para validação das redes foi diferente do proposto no artigo original

de Elman (1990). Enquanto o artigo original utiliza uma única validação para ilustrar o resultado da aplicação da rede, os gráficos das figuras 7.3 e 7.4 ilustram a média entre 20 validações diferentes. Tal procedimento busca avaliar o comportamento médio da rede em diferentes aplicações, e apresentam um valor de erro consideravelmente maior do que o caso de uma única aplicação bem sucedida do treinamento. Isto ocorre porque, dada a aleatoriedade dos valores iniciais de pesos, a rede não atingiu a convergência esperada em algumas das validações, o que levantou o valor médio dos erros.

Os experimentos com ruído foram executados de forma semelhante aos anteriores, e os resultados, considerando a mesma métrica utilizada para avaliação, é ilustrada na figura 7.4. Mais uma vez podemos notar a melhoria do desempenho com a inserção do conhecimento temporal, e a comparação com o resultado da rede de Elman. Outro aspecto importante que merece ser destacado neste gráfico consiste no impacto da inserção de conhecimento de background sobre a tolerância a ruídos da rede.

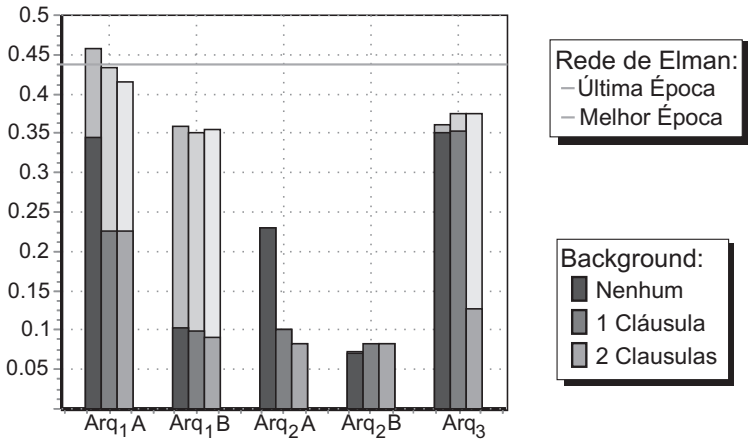

Figura 7.4: Erro das arquiteturas para experimento XOR com ruído

7.3 Operadores Temporais

Uma série de experimentos foi realizada para avaliação da representação e aprendizagem de operadores temporais. Primeiramente, consideramos o caso dos operadores ■ e ♦. Para tais operadores, consideramos a aplicação de seqüências de 10 bits (1 para 'verdadeiro' e 0 para 'falso'), considerando como saída desejada os valores dos operadores considerados desde o início desta seqüência. Todas as 1024 seqüências possíveis foram consideradas, em um processo de '8-fold cross validation', onde cada validação consistia em um conjunto de treinamento com 896 e um conjunto de teste com 128 destas seqüências, agrupadas aleatoriamente.

As arquiteturas utilizadas estão de acordo com a figura 7.5 Num primeiro momento, os experimentos foram realizados separadamente para cada operador, considerando apenas o conjunto de neurônios necessários para representação do operador específico. Para cada operador, duas redes diferentes foram consideradas, uma rede gerada pela tradução das cláusulas que representam o operador, e outra rede com a mesma arquitetura porém pesos e bias aleatórios. Os valores de erro (RMSE) obtidos neste experimento são mostrados na primeira coluna de valores da tabela 7.2. As demais configurações utilizadas para a rede foram semelhantes às utilizadas no experimento com a função XOR.

Outro experimento foi realizado sobre a arquitetura completa, considerando diferentes

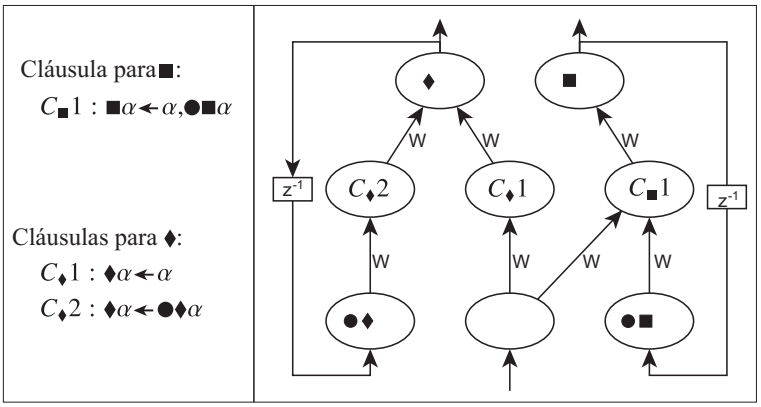

Figura 7.5: Programa lógico e arquitetura para representação do operador ♦ e ■

inicializações de pesos para representação de diferentes níveis de conhecimento de 'background'. Para representar a ausência de conhecimento, a arquitetura foi gerada com todos os pesos e bias aleatórios, e a inserção de cada cláusula representando o conhecimento de um operador é representada através da inicialização dos pesos e biases relacionados. Os valores destes experimentos são ilustrados na coluna da direita da tabela 7.2. Em ambos os experimentos, podemos notar que, com exceção das cláusulas representando o operador ♦, a inserção de conhecimento melhorou o desempenho da rede. De qualquer forma, os resultados obtidos para todos os experimentos apresentaram um valor de erro baixo.

Tabela 7.2: Resultados dos experimentos com os operadores ■ e ♦

Experimento individual		Experimento completo	
Experimento	RMSE	Experimento.	RMSE
■(Sem cláusulas)	$3.07 \cdot 10^{-2}$	Sem cláusulas	5.5×10^{-2}
■(Com cláusulas)	$2.03 \cdot 10^{-3}$	Cláusula ■	$2.38 \cdot 10^{-2}$
♦(Sem cláusulas)	$3.06 \cdot 10^{-3}$	Cláusula ♦	$1.46 \cdot 10^{-1}$
♦(Com cláusulas)	$3.84 \cdot 10^{-3}$	Todas cláusulas	$4.78 \cdot 10^{-4}$

Outro experimento realizado foi com relação ao operador *since*. Neste caso, o processo de 10-fold cross validation foi realizado sobre um conjunto de 1000 padrões, onde a entrada é definida por dois valores (A e B) e a saída é dada pelo operador $A\mathbb{S}B$. Um exemplo desta seqüência é descrito na tabela 7.3 A representação deste operador é dada por duas cláusulas ($(\alpha \mathbb{S} \beta) \leftarrow \beta$ e $(\alpha \mathbb{S} \beta) \leftarrow \alpha, \bullet \alpha \mathbb{S} \beta$), traduzidas para rede neural, considerando diferentes inicializações de pesos para representação de conhecimento de background.

Tabela 7.3: Exemplo de uma seqüência para o operador 'Since'

Time:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
α :	F	F	T	T	F	T	T	T	T	F
β :	F	T	F	F	F	F	F	T	F	T
$\alpha \mathbb{S} \beta$:	F	T	T	T	F	F	F	T	T	T

As diferentes representações de conhecimento utilizadas e os resultados dos experimentos sobre duas execuções do processo de *10-fold cross validation* estão descritos na

tabela 7.4. As demais configurações usadas para as redes neurais foram as mesmas dos experimentos anteriores. Mais uma vez, notamos que as redes apresentaram um excelente desempenho. Porém, a inserção de conhecimento de background apresentou um efeito desprezível sobre este desempenho.

Tabela 7.4: Resultados experimentos com operador 'Since'

Set of rules	RMSE
$(\alpha \mathbb{S} \beta) \leftarrow \beta$; $(\alpha \mathbb{S} \beta) \leftarrow \alpha, \bullet \alpha \mathbb{S} \beta$	7.22×10^{-3}
$(\alpha \mathbb{S} \beta) \leftarrow \beta$	7.21×10^{-3}
$(\alpha \mathbb{S} \beta) \leftarrow \alpha, \bullet \alpha \mathbb{S} \beta$	7.07×10^{-3}
No rules	7.09×10^{-3}

7.4 O problema dos filósofos glutões

Para uma análise mais profunda da representação e aprendizagem de conhecimento temporal em nosso modelo, consideramos um caso de teste clássico para ilustrar a sincronização de processos, utilizado principalmente em sistemas distribuídos. O problema dos 'filósofos glutões', proposto originalmente por Dijkstra (1971) pode ser descrito da seguinte maneira: *n filósofos estão sentados ao redor de uma mesa circular. Cada filósofo passa o tempo alternadamente pensando e comendo. No centro da mesa existe um grande prato de macarrão, e cada filósofo precisa de dois garfos para comer. Porém, o número de garfos sobre a mesa é igual ao número de filósofos. Um garfo é colocado entre cada par de filósofos, e eles concordam entre si que cada um vai utilizar apenas os garfos imediatamente à sua esquerda e à sua direita.*

Para representação do problema, consideramos um sistema descrito da seguinte maneira. O ambiente envia, aleatoriamente, mensagens $Fome_i$ e $Satisf_i$ para cada agente i. A primeira mensagem, indica para o agente que ele deve buscar alocar os recursos necessários para comer. Tendo alocado os recursos, o agente deve comer até que o ambiente envie a mensagem $satisf_i$, quando o agente deve parar de comer e liberar os garfos. O agente interage com o ambiente através de 5 ações diferentes. As ações $PegaEsq_i$ e $PegaDir_i$, indicam que o agente tenta alocar os garfos à sua esquerda e direita, respectivamente. Quando a tentativa é bem sucedida, o agente responde com uma mensagem $AlocEsq_i$ ou $AlocDir_i$. A ação $Come_i$ indica que o agente está comendo, e as ações $LargaEsq_i$ e $LargaDir_i$ representam a desalocação dos garfos por parte do agente. Note que cada agente é responsável pelo armazenamento da informação sobre seus estados (como por exemplo estar com fome, possuir um garfo, etc). Esta informação não é fornecida pelo ambiente, com o intuito de requerer que cada agente apresente um mecanismo de 'memória de curto prazo' para armazenamento de seu estado.

Para a descrição de cada agente (filósofo) no sistema, consideramos uma política bem definida de alocação de recursos, onde cada agente tenta primeiramente alocar o garfo da esquerda para depois alocar o da direita. Tal estratégia vai de acordo com a definição do ambiente para resolução de conflitos, que define que quando dois agentes tentam pegar um garfo, o agente da esquerda tem prioridade. Desta forma, o agente que detém um recurso tem prioridade, reduzindo o risco de deadlock. A única forma de acontecer deadlock, no caso dos agentes seguirem a política definida, é quando todos os agentes tentam alocar o garfo da esquerda ao mesmo tempo. Neste caso, o ambiente é dotado de um mecanismo para escolher aleatoriamente um agente que não receberá o garfo. O

Tabela 7.5: Comportamento de um agente no problema dos filósofos glutões

$PegaEsq_1 \mathbb{W} AlocEsq_1 \leftarrow Fome_1$ $PegaDir_1 \mathbb{W} AlocDir_1 \leftarrow AlocEsq'_1$ $Come_1 \mathbb{W} Satisf_1 \leftarrow AlocDir'_1$ $LargaEsq_1 \leftarrow Satisf'_1$ $LargaDir_1 \leftarrow Satisf'_1$
$PegaEsq_1 \mathbb{W} AlocEsq_1 \leftarrow \bullet(PegaEsq_1 \mathbb{W} AlocEsq_1), \sim \bullet AlocEsq_1$ $PegaEsq_1 \leftarrow PegaEsq_1 \mathbb{W} AlocEsq_1, \sim AlocEsq'_1$ $PegaDir_1 \mathbb{W} AlocDir_1 \leftarrow \bullet(PegaDir_1 \mathbb{W} AlocDir_1), \sim \bullet AlocDir_1$ $PegaDir_1 \leftarrow PegaDir_1 \mathbb{W} AlocDir_1, \sim AlocDir'_1$ $Come_1 \mathbb{W} Satisf_1 \leftarrow \bullet(Come_1 \mathbb{W} Satisf_1), \sim \bullet Satisf_1$ $Come_1 \leftarrow Come_1 \mathbb{W} Satisf_1, \sim Satisf'_1$
$AlocEsq_1 \leftarrow AlocEsq'_1$ $AlocDir_1 \leftarrow AlocDir'_1$ $Satisf_1 \leftarrow Satisf'_1$

comportamento do agente é descrito na forma de um programa em lógica temporal, como definido na tabela 7.5. A região de cima da tabela mostra o programa original. A região central mostra as cláusulas inseridas para caracterizar a semântica do operador $\mathbb{W}$. Na região inferior, encontramos cláusulas inseridas para resolução do *missing links issue* na propagação temporal dos valores, de acordo com a segunda abordagem descrita na seção 6.2.

Para os experimentos envolvendo aprendizagem, consideraremos arquiteturas de rede representando três diferentes configurações de conhecimento de background. Os agentes FK (Full Knowledge) são representados através da tradução direta do programa ilustrado na tabela 7.5. Os agentes PK apresentam apenas as cláusulas de propagação de valores e representação de operadores temporais, contidas nas regiões central e inferior da tabela. E os agentes NK não apresentam nenhum conhecimento de background, ou seja, utilizam a mesma arquitetura de rede dos demais, porém todos os pesos são inicializados aleatoriamente. Com o intuito de verificar o resultado da tradução, a arquitetura para este exemplo é mostrada na figura 7.6.

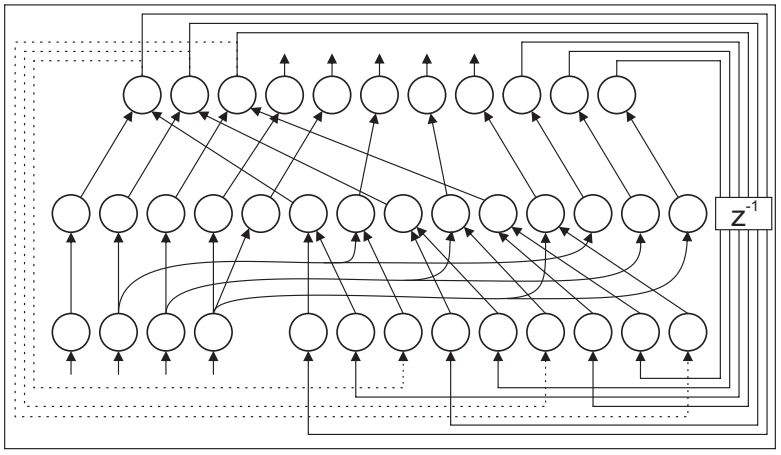

Figura 7.6: Arquitetura representando um agente (filósofo)

O primeiro experimento de aprendizagem consiste no uso de aprendizagem offline,

ou seja, o agente aprende com o ambiente sem interferir nele. A execução do ambiente com três agentes com comportamento definidos pelo programa lógico é realizada, e um agente externo 'observa' as ações de um dos agentes inseridos no ambiente, aprendendo seu comportamento. Para tal experimento, 500 séries de treinamento-teste foram executadas, sendo cada série formada por 200 aplicações de valores para treinamento e 200 para teste. Note que este experimento considera uma execução normal do ambiente, com os valores de $fome$ e $satisf$ sendo gerados aleatoriamente no decorrer da execução, sem nenhuma adaptação para o treinamento. As redes para representar o conhecimento dos agentes apresentam a mesma configuração para aprendizagem descrita nos experimentos anteriores.

Tabela 7.6: Experimentos com aprendizagem Offline

	FK	HK	NK
RMSE ≤ 0.2	0	73	137.88
RMSE ≤ 0.1	0	80	155.63
Menor Erro	0.0032	0.016	0.082
Último erro	0.0032	0.07	0.79

Na tabela 7.6 mostramos os resultados médios obtidos em 8 execuções do processo de treinamento-teste definido acima. Nas duas primeiras linhas, mostramos o número médio de épocas necessárias para atingir o valor de RMSE descrito (0.2 na linha superior e 0.1 na linha inferior). Nas demais linhas, mostramos as médias dos valores de erro obtido na melhor época e na última época de treinamento dos agentes. Vale lembrar que, diferentemente dos experimentos acima, estes valores de erro são calculados diretamente no intervalo de saída da rede, ou seja, entre -1 e 1. A figura 7.7 ilustra a evolução do erro em função do tempo para um caso de treinamento de cada configuração de agente.

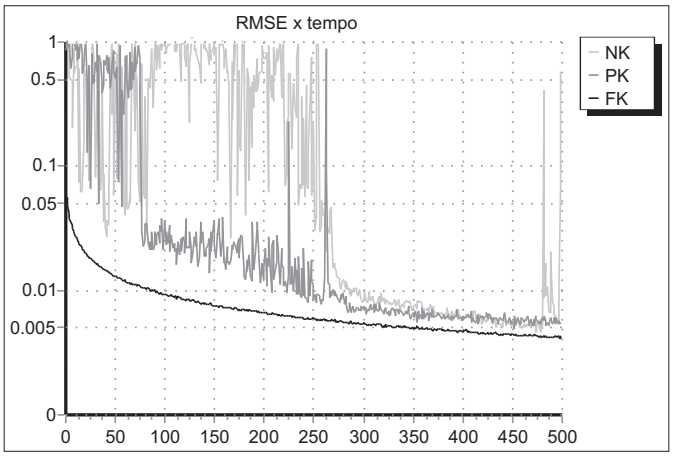

Figura 7.7: Evolução do erro considerando aprendizagem offline

O mesmo ambiente com três agentes foi considerado para avaliação da aprendizagem online. Neste caso, dois agentes seguem o padrão de comportamento definido pelo programa lógico, e um terceiro agente com diferentes configurações iniciais de conhecimento é inserido no ambiente, interagindo com os demais. Neste caso, após cada ação deste agente aprendiz, o sistema aplica na saída da rede os valores representando a ação que deveria ter sido tomada, de acordo com a política padrão, permitindo a aprendizagem supervisionada. O experimento foi realizado durante 100.000 momentos para cada

agente, e a evolução do erro para um experimento de cada configuração são mostrados na figura 7.8.

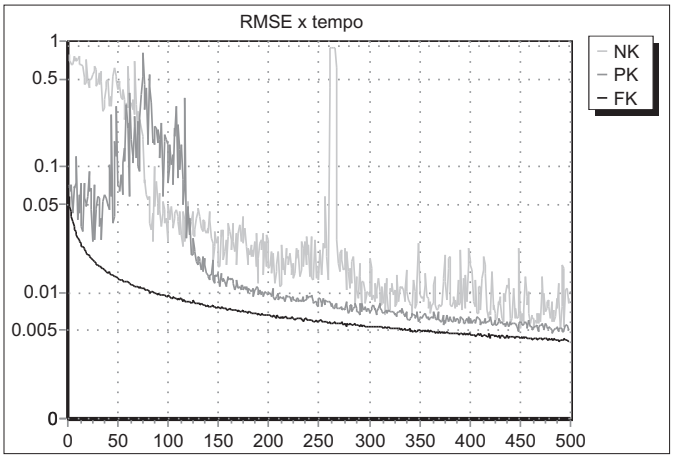

Figura 7.8: Evolução do erro considerando aprendizagem offline

Neste gráfico, os valores de erro são agrupados em conjuntos de 200 momentos consecutivos, e o valor de RMSE é calculado sobre estes conjuntos, visando a similaridade com o gráfico ilustrando a aprendizagem Offline. Para avaliar a execução do sistema, e a sincronização entre os agentes, consideramos como métrica a razão entre o número de agentes comendo por intervalo de tempo e o número de agentes desejando estar comendo (agentes que não receberam o sinal $satisf$ desde a última vez que receberam o sinal $fome$). O gráfico com a evolução da taxa de alocação de recursos é definida na figura 7.9.

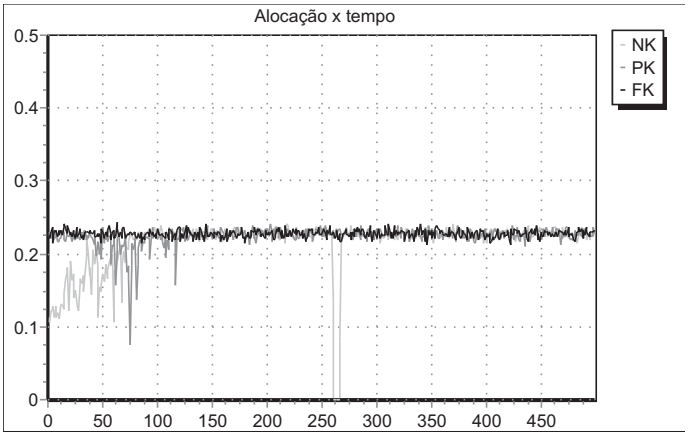

Figura 7.9: Evolução da taxa de alocação de recursos

Em ambos os casos de aprendizagem, podemos notar um considerável efeito da inserção do conhecimento na convergência da aprendizagem e desempenho do agente com relação ao erro de saída. Por outro lado, notamos também que esta aprendizagem se reflete diretamente no desempenho do sistema como um todo, no caso da aprendizagem offline. Analisando os gráficos nas figuras 7.8 e 7.9, verificamos que o melhor desempenho do sistema como um todo se dá no momento que o agente apresenta menor taxa de erro. Especificamente, se notarmos a região do gráfico ao redor dos pontos 260 e 270 (representando os momentos entre 52.000 e 54.000), para o agente sem conhecimento de

background, podemos verificar claramente que um desvio no processo de aprendizagem acarretou num bloqueio do sistema devido à alocação imprópria de recursos.

8 CONCLUSÕES

Neste trabalho, mostramos um conjunto de investigações com o intuito de prover estratégias para a representação de conhecimento temporal, descrito de forma lógica, em sistemas conexionistas, bem como para a aplicação de rotinas de aprendizagem sobre as redes neurais consideradas neste processo.

Com relação à representação do conhecimento temporal, apresentamos o modelo SCTL, que contempla um algoritmo de tradução de programas em lógica temporal para redes neurais com recursos próprios para propagação de valores no tempo, mostrando uma prova formal da equivalência semântica das representações simbólica e conexionista. Conforme as comparações realizadas com outro modelo existente para esta representação, verificamos que o SCTL apresenta diversas vantagens, como a redução na complexidade da rede gerada, representação natural do fluxo de tempo e adequação à aplicação de algoritmos de aprendizagem temporal. Por outro lado, mostramos também que grande parte das desvantagens do SCTL podem ser contornadas através de adaptação na modelagem do conhecimento simbólico.

O trabalho também mostrou uma estratégia para a realização de aprendizagem empírica e supervisionada sobre o modelo. Além de mostrar a efetividade da aplicação deste processo de aprendizagem, foi demonstrado através de experimentos, os efeitos positivos da inserção conhecimento simbólico em um modelo neural, como melhoria no desempenho e convergência do algoritmo e aumento na tolerância a ruídos.

Apesar de contemplar diferentes aspectos envolvendo a integração de sistemas neurais e simbólicos sob uma perspectiva temporal, este trabalho consiste apenas num passo inicial nesta direção. Diferentes esforços podem ser apontados como trabalhos futuros no sentido de fortalecer esta integração. Com relação à representação de conhecimento simbólico, podemos citar como passos importantes a extensão do modelo de representação, para considerar um conjunto maior de inferências temporais, e a integração com outras formas de representação de conhecimento, como lógicas modais, epistêmicas ou intuicionistas.

Também vale mencionar que diferentes aspectos com relação a aprendizagem empírica podem ser fortalecidos. Entre eles, uma exploração maior dos diferentes algoritmos de aprendizagem temporal baseados no Backpropagation, bem como nas variações de configuração para estes algoritmos. Além disso, para aumentar a aplicabilidade do modelo temporal, o estudo de algoritmos para aprendizagem por reforço surge como uma possibilidade natural.

Para completar o ciclo Representação - Aprendizagem - Extração de Conhecimento, a investigação e desenvolvimento de rotinas para extração de regras temporais consiste em um tema de pesquisa importante para o modelo SCTL, bem como para redes neurais temporais em geral. A utilização de lógicas temporais para extração de conhecimento, bem

como de outras lógicas não clássicas, pode surgir como uma possibilidade interessante para aumentar a expressividade e a clareza das regras extraídas de ambientes conexionistas.

Por fim, vale ressaltar que os alguns dos métodos propostos neste trabalho são extensíveis a outros modelos de representação e estruturas conexionistas. A utilização de ligações recorrentes e unidades de atraso para propagação de valores no tempo, por exemplo, pode ser integrada em diferentes modelos de redes neurais, e desta forma servir como mecanismo para auxiliar na representação temporal em outros modelos. Desta forma, num contexto geral da integração neuro-simbólica, este trabalho desempenha um papel estratégico para futuras pesquisas e desenvolvimentos, no que diz respeito à incorporação de uma dimensão temporal para inferência e aprendizagem.

REFERÊNCIAS

ALLEN, J. F. Mantaining Knowledge about temporal intervals. **Communications of the ACM**, New York, v.26, n.3, p.832–843, 1983.

ANDREWS, R.; DIEDERICH, J.; TICKLE, A. B. A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. **Knowledge-based Systems**, [S.l.], v.8, n.6, p.373–389, 1995.

ARKOUDAS, K. Specification, Abduction, and Proof. In: ATVA, 2004. **Autpmated Technology for Verification and Analysis:** proceedings. Berlin: Springer, 2004. p.294–309. (Lecture Notes in Computer Science, v.3299).

BARRINGER, H. et al. METATEM: an introduction. **Formal Asp. Comput.**, [S.l.], v.7, n.5, p.533–549, 1995.

BEN-ARI, M.; MANNA, Z.; PNUELI, A. The temporal logic of branching time. In: ACM SIGPLAN-SIGACT SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES, POPL, 8., 1981. **Proceedings...** New York: ACM Press, 1981. p.164–176.

BENCH-CAPON, T. J. M. Persuasion in Practical Argument Using Value-based Argumentation Frameworks. **Journal of Logic and Computation**, [S.l.], v.13, n.3, p.429–448, 2003.

BLACKBURN, P.; RIJKE, M. de; VENEMA, Y. **Modal Logic**. Cambridge, UK: Cambridge University Press, 2001.

BORGES, R. V.; LAMB, L. C.; D'AVILA GARCEZ, A. S. Combining Architectures for Temporal Learning in Neural-Symbolic Systems. in: INTERNATIONAL CONFERENCE ON HYBRID INTELLIGENT SYSTEMS, HIS, 6., 2006. **Proceedings ...** Los Alamitos, CA: IEEE Computer Society, 2006. p. 46.

BORGES, R. V.; LAMB, L. C.; D'AVILA GARCEZ, A. S. Towards Reasoning about the past in Neural Symbolic Systems. In: INTERNATIONAL WORKSHOP ON NEURAL-SYMBOLIC LEARNING AND REASONING IJCAI, 3., 2007. **Proceedings...** [S.l.: s.n.], 2007.

BRODA, K.; GABBAY, D. M.; LAMB, L. C.; RUSSO, A. **Compiled Labelled Deductive Systems**: a uniform presentation of non-classical logics. Baldock, Hertfordshire, England: Research Studies Press, 2004.

BROWNE, A.; SUN, R. Connectionist Variable Binding. In: WERMTER, S.; SUN, R. (Ed.). **Hybrid Neural Systems**. Heidelberg: Springer Verlag, 2000.

BROWNE, A.; SUN, R. Connectionist inference models. **Neural Networks**, [S.l.], v.14, n.10, p.1331–1355, 2001.

CLOUSE, D. S. et al. Time-Delay Neural Networks: representation and induction of finite-state machines. **IEEE Transactions on Neural Networks**, [S.l.], v.8, n.5, p.1065–1070, 1997.

D'AVILA GARCEZ, A. S.; BRODA, K.; GABBAY, D. M. Symbolic knowledge extraction from trained neural networks: a sound approach. **Artificial Intelligence**, [S.l.], v.125, n.1-2, p.155–207, 2001.

D'AVILA GARCEZ, A. S.; BRODA, K.; GABBAY, D. M. **Neural-Symbolic Learning Systems**: foundations and applications. [S.l.]: Springer-Verlag, 2002. (Perspectives in Neural Computing).

D'AVILA GARCEZ, A. S.; GABBAY, D. M.; LAMB, L. C. Value-based Argumentation Frameworks as Neural-Symbolic Learning Systems. **Journal of Logic and Computation**, [S.l.], v.15, n.6, p.1041–1058, 2005.

D'AVILA GARCEZ, A. S.; LAMB, L. C. Neural-Symbolic Systems and the Case for Non-Classical Reasoning. In: ARTËMOV, S. N.; BARRINGER, H.; d'AVILA GARCEZ, A. S.; LAMB, L. C.; WOODS, J. (Ed.). **We Will Show Them! Essays in honour of Dov Gabbay**. [S.l.]: College Publications, 2005. p.469–488.

D'AVILA GARCEZ, A. S.; LAMB, L. C. A connectionist computational model for epistemic and temporal reasoning. **Neural Computation**, [S.l.], v.18, n.7, p.1711–1738, 2006.

D'AVILA GARCEZ, A. S.; LAMB, L. C.; BRODA, K.; GABBAY, D. M. Applying connectionist modal logics to distributed knowledge representation problems. **International Journal on Artificial Intelligence Tools**, [S.l.], v.13, n.1, p.115–139, 2004.

D'AVILA GARCEZ, A. S.; LAMB, L. C.; GABBAY, D. M. A Connectionist Inductive Learning System for Modal Logic Programming. In: IEEE INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING, ICONIP, 2002, Singapore. **Proceedings...** [S.l.: s.n.], 2002.

D'AVILA GARCEZ, A. S.; ZAVERUCHA, G. The Connectionist Inductive Learning and Logic Programming System. **Applied Intelligence**, Hingham, MA, USA, v.11, n.1, p.59–77, 1999.

DIJKSTRA, E. W. Hierarchical Ordering of Sequential Processes. **Acta Inf.**, [S.l.], v.1, p.115–138, 1971.

ELMAN, J. L. Finding Structure in Time. **Cognitive Science**, [S.l.], v.14, n.2, p.179–211, 1990.

FAGIN, R. et al. **Reasoning about Knowledge**. [S.l.]: MIT Press, 1995.

FEIGENBAUM, E. A. Some challenges and grand challenges for computational intelligence. **J. ACM**, New York, NY, USA, v.50, n.1, p.32–40, 2003.

FISCHER, M.; GABBAY, D.; VILA, L. (Ed.). **Handbook of temporal reasoning in artificial intelligence**. [S.l.]: Elsevier, 2005.

FODOR, J. A.; PYLYSHYN, Z. W. Connectionism and cognitive architecture: a critical analysis. In: PINKER, S.; MEHLER, J. (Ed.). **Connections and Symbols**. Cambridge, MA: MIT Press, 1988. p.3–72.

GABBAY, D. M.; HODKINSON, I.; REYNOLDS, M. **Temporal logic**: mathematical foundations and computational aspects. New York, NY, USA: Oxford University Press, 1994.

GALTON, A. Temporal Logic. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. [S.l.: s.n.], 2003.

GELFOND, M.; LIFSCHITZ, V. The Stable Model Semantics for Logic Programming. In: ICLP, 1988. **Proceedings...** [S.l.]: MIT Press, 1988. p.1070–1080.

HALPERN, J. Y. et al. On the unusual effectiveness of logic in computer science. **Bulletin of Symbolic Logic**, [S.l.], v.7, n.2, p.213–236, 2001.

HAYKIN, S. **Neural Networks**: a compreensive foundation. 2nd ed. [S.l.]: Prentice Hall, 1999.

HILARIO, M. An Overview of Strategies for Neurosymbolic Integration. In: WORKSHOP ON CONNECTIONIST-SYMBOLIC INTEGRATION: FROM UNIFIED TO HYBRID APPROACHES, IJCAI, 1995. **Proceedings...** [S.l.: s.n.], 1995.

HITZLER, P.; HÖLLDOBLER, S.; SEDA, A. K. Logic programs and connectionist networks. **J. Applied Logic**, [S.l.], v.2, n.3, p.245–272, 2004.

HöLLDOBLER, S.; KALINKE, Y. Towards a new massively parallel computational model for logic programming. In: WORKSHOP ON COMBINING SYMBOLIC AND CONNECTIONIST PROCESSING, ECAI, 1994, Amsterdan. **Proceedings...** [S.l.: s.n.], 1994. v.2, p.68–77.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, Oxford, UK, v.2, n.5, p.359–366, 1989.

HUTH, M. R. A.; RYAN, M. D. **Logic in Computer Science**: modelling and reasoning about systems. Cambridge, UK: Cambridge University Press, 2000.

KASABOV, N. K. **Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering**. Cambridge, MA, USA: MIT Press, 1996.

KUPFERMAN, O.; PNUELI, A. Once and For All. In: IEEE SYMPOSIUM ON LOGIC IN COMPUTER SCIENCE, LICS, 1995. Washington, DC, USA. **Proceedings...** [S.l.]: IEEE Computer Society, 1995. p.25.

LAMPORT, L. "Sometime"is sometimes "not never": on the temporal logic of programs. In: ACM SIGPLAN-SIGACT SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES, POPL, 1980. **Proceedings...** New York: ACM Press, 1980. p.174–185.

LLOYD, J. W. **Logic for learning**: learning comprehensible theories from structured data. Berlin: Springer, 2003.

MARTINELLI, E. **Extração de conhecimentos de Redes Neurais Artificiais**. 1999. Dissertação (Mestrado em Ciência da Computação) — Instituto de Ciências Matemáticas e da Computação, Universidade de São Paulo, São Carlos, SP.

MCCARTHY, J. Programs with Common Sense. In: TEDDINGTON CONFERENCE ON THE MECHANIZATION OF THOUGHT PROCESSES, 1959, London. **Proceedings...** [S.l.]: Her Majesty's Stationary Office, 1959. p.75–91.

MICHALSKI, R. S. Learning strategies and automated knowledge acquisition: an overview. **Computational models of learning**, London, UK, p.1–19, 1987.

MINSKY, M. L.; PAPERT, S. A. **Perceptron**. Cambridge, MA: MIT Press, 1969.

NEWELL, A.; SIMON, H. A. Computer science as empirical inquiry: symbols and search. **Commun. ACM**, New York, NY, USA, v.19, n.3, p.113–126, 1976.

NOBRE, C. M. et al. Extração de Conhecimento: uma comparação entre os métodos clássico e conexionista. In: SIMPóSIO BRASILEIRO DE REDES NEURAIS, SBRN, 5., 1998, Belo Horizonte, MG. **Anais...** Belo Horizonte: Sociedade Brasileira de Computação, 1998. p.126–131.

ORGUN, M. A.; MA, W. An Overview of Temporal and Modal Logic Programming. In: INTERNATIONAL CONFERENCE ON TEMPORAL LOGIC, ICTL, 1., 1994. **Temporal Logic**. Berlin: Springer-Verlag, 1994. p.445–479. (Lecture Notes in Computer Science, v. 827).

PAGE, M. Connectionist modelling in psychology: a localist manifesto. **Behavioural and Brain Sciences**, [S.l.], v.23, p.443–512, 2000.

PNUELI, A. The Temporal Logic of Programs. In: IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, FOCS, 18., 1977. **Proceedings...** [S.l.]: IEEE Computer Society, 1977. p.46–67.

ROSEMBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, [S.l.], v.65, p.386–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: RUMELHART, D. E; McCLELLAND, J. L. **Parallel distributed processing:** explorations in the microstructure of cognition. Cambridge: The MIT Press, 1986. v.1, p.318–362.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence**: a modern approach. [S.l.]: Pearson Education, 2003.

SCHIFFMANN, W.; JOOST, M.; WERNER, R. **Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons**. Koblenz, Germany: [s.n.], 1994.

SIEGELMANN, H. T.; HORNE, B. G.; GILES, C. L. **Computational capabilities of recurrent NARX neural networks**. College Park, MD, USA: [s.n.], 1995.

SIEGELMANN, H. T.; SONTAG, E. D. On the computational power of neural nets. In: ANNUAL WORKSHOP ON COMPUTATIONAL LEARNING THEORY, COLT, 5., 1992. **Proceedings...** New York: ACM Press, 1992. p.440–449.

SMOLENSKY, P. On The Proper Treatment of Connectionism. **Behavioral and Brain Sciences**, [S.l.], v.11, n.1, p.1–23, 1988.

TOWELL, G. G.; SHAVLIK, J. W. Knowledge-based artificial neural networks. **Artificial Intelligence**, Essex, UK, v.70, n.1-2, p.119–165, 1994.

VALIANT, L. G. Three problems in computer science. **J. ACM**, New York, NY, USA, v.50, n.1, p.96–99, 2003.

WAIBEL, A. et al. Phoneme recognition using time-delay neural networks. In: WAIBEL, A.; LEE, K. F. (Ed.). **Readings in speech recognition**. San Mateo: Morgan Kaufmann, 1990. p.393–404.

WERBOS, P. Backpropagation through time: what it does and how to do it. **Proceedings of the IEEE**, Washington, DC, USA, v.78, n.5, p.1550–1560, 1990.

WERMTER, S.; SUN, R. **Hybrid Neural Systems**. [S.l.]: Springer, 2000.