HAROLDO ISSAO GUIBU

Tecnologias adaptativas aplicadas na flexibilização de redes neurais artificiais e redes de Petri

São Paulo
2018

HAROLDO ISSAO GUIBU

Tecnologias adaptativas aplicadas na flexibilização de redes neurais artificiais e redes de Petri

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências.

São Paulo
2018

HAROLDO ISSAO GUIBU

Tecnologias adaptativas aplicadas na flexibilização de redes neurais artificiais e redes de Petri

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências.

Área de Concentração: Engenharia Elétrica. Computação e Sistemas Digitais.

Orientador: Prof. Dr. João José Neto

São Paulo

2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência do orientador.

São Paulo, _____de _____de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Guibu, Haroldo Issao
Tecnologias Adaptativas aplicadas na flexibilização de Redes Neurais Artificiais e Redes de Petri. / H.I. Guibu – versão corr.- São Paulo, 2018, 120 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Autômatos Adaptativos 2. Redes de Petri Adaptativas 3. Redes Neurais Artificiais 4. Aprendizado de Máquinas 5.Sistemas de Apoio à Tomada de Decisão .Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas.

AGRADECIMENTOS

Agradeço a Deus por realizar milagres a partir de nossos erros e sempre oferecer uma alternativa quando enveredamos pelos caminhos errados.

Agradeço à minha família, em particular a Emi, Aquemi, Akira, Massumi e Sumie, por me apoiarem incondicionalmente.

Agradeço aos meus colegas do IFSP professores Alexandre Ventieri, José Augusto Pinheiro Christino e professora Sara Dereste dos Santos por compartilharem de forma generosa nossas aulas e ao professor Alberto Akio Shiga por seu apoio à qualificação docente.

Agradeço aos meus colegas do Laboratório de Linguagens e Técnicas Adaptativas, em especial ao Paulo Cereda e ao Newton Miura por me ajudarem de todas as formas e por uma convivência extremamente agradável.

Agradeço ao professor Dr. João José Neto por anos de incentivos e paciência e por incorporar a mão de Deus, evitando que suas “ovelhas” caiam em despenhadeiros e armadilhas.

Muitíssimo Obrigado.

Deus abençoe e ilumine a todos.

Ora et Labora (Reza e Trabalha)

Preceito da Regra São Bento.

Trabalha como se tudo dependesse de ti e confia como se tudo dependesse de Deus

Santo Inácio de Loyola.

RESUMO

O objetivo deste trabalho é o estudo das tecnologias adaptativas aplicadas às redes neurais artificiais e às redes de Petri. Além disso, uma metodologia é proposta para estas aplicações a partir da definição de uma rede de Petri colorida adaptativa.

Inicialmente, as redes neurais artificiais são estudadas do ponto de vista da extração de regras. Uma das críticas recorrentes às redes neurais artificiais é a característica de “caixa preta” das soluções, significando que as soluções escondem o mecanismo de funcionamento, deixando em dúvida a razão de seu funcionamento. A extração de regras a partir das redes neurais artificiais objetiva apresentar uma solução equivalente baseada em regras que para os especialistas em uma determinada área seja mais inteligível ou transparente.

Outro ponto importante é a inserção de regras nas redes neurais artificiais. Esta inserção é possível a partir da versão baseada em regras das redes neurais artificiais. Um especialista humano em uma área muitas vezes cria um conjunto de regras que o auxiliam na compreensão do problema. Se estas regras forem inseridas ao conjunto de regras obtidas através dos dados, o novo conjunto de regras conterá ao mesmo tempo o conhecimento humano e o conhecimento extraído dos dados. As tecnologias adaptativas de extração e inserção de regras tornam as soluções mais flexíveis.

As redes de Petri são, em certo sentido, complementares às redes neurais artificiais pois foram criadas para tratar os “Sistemas a Eventos Discretos” ou sistemas sequenciais, enquanto que as redes neurais artificiais possuem uma natureza combinatória. Muitas extensões foram propostas à redes de Petri ao longo dos anos e entre estas extensões aparecem associações de redes de Petri e redes neurais artificiais.

Nestas associações, muitas técnicas desenvolvidas para as redes neurais artificiais foram incorporadas às redes de Petri como, por exemplo, as diversas formas de aprendizado. Utilizando a característica das redes de Petri de modelagem de sistemas sequenciais, a fase de treinamento das redes neurais artificiais pode ser controlada pela rede de Petri. Neste trabalho, a incorporação de regras à rede de Petri é examinada assim como a sua aplicação a sistemas de apoio à decisão e a sistemas de manufatura flexível.

Palavras-chave: Autômatos Adaptativos. Redes de Petri Adaptativas. Redes Neurais. Aprendizado de Máquinas. Sistemasde Apoio à Decisão.

ABSTRACT

The objective of this work is the study of adaptive technologies applied to artificial neural networks and Petri nets. In addition, a methodology is proposed for these applications from the definition of an adaptive color Petri net.

Initially, artificial neural networks are studied from the point of view of rule extraction. One of the recurring criticisms of artificial neural networks is the "black box" feature of the solutions, meaning that the solutions hide the working mechanism, casting doubt on the reason for its operation. The extraction of rules from the artificial neural networks aims to present an equivalent solution based on rules that for the experts in a given area is more intelligible or transparent.

Another important point is the insertion of rules in artificial neural networks. This insertion is possible from the rule-based version of artificial neural networks. A human expert in an area often creates a set of rules that aid in understanding the problem. If these rules are inserted into the set of rules obtained from the data, the new set of rules will contain at the same time the human knowledge and the knowledge extracted from the data. Adaptive rule extraction and insertion technologies make solutions more flexible.

Petri nets are, in a sense, complementary to artificial neural networks as they were designed to treat "Discrete Event Systems" or sequential systems, while artificial neural networks have a combinatorial nature. Many extensions have been proposed to the Petri nets over the years and among these extensions appear associations of Petri nets and artificial neural networks.

In these associations, many techniques developed for artificial neural networks were incorporated into Petri nets, such as the various forms of learning. Using the Petri nets feature of sequential modeling, the training phase of artificial neural networks can be controlled by the Petri net. In this work, the incorporation of rules into the Petri net is examined as well as its application to decision support systems and flexible manufacturing systems.

Keywords: Adaptive Automata. Adaptive Petri Networks. Artificial Neural Networks. Machine Learning. Decision Support Systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANFIS	Adaptive Neural Fuzzy Inference System
FIS	Fuzzy Inference System
RBF	Rede Neural tipo Radial Basis Function
RNA	Rede Neural Artificial
RP	Rede de Petri
RPAM	Rede de Petri Auto-Modificável
RPC	Rede de Petri Colorida
RPCA	Rede de Petri Colorida Adaptativa

Sumário

1 INTRODUÇÃO

1.1 Motivação

Em seu livro “O Algoritmo Mestre” (DOMINGOS, 2017), Domingos apresenta uma alegoria sobre aprendizagem de máquinas como um território divido entre cinco tribos: os Analogistas, os Simbolistas, os Conexionistas, os Bayesianos e os Evolucionários.

Domingos utiliza esta divisão para explicar as diversas abordagens utilizadas no desenvolvimento de algoritmos de aprendizagem descritas a seguir.

- Analogistas: A aprendizagem é obtida pela extrapolação de julgamentos de similaridade e eles são influenciados pela psicologia e pela otimização matemática. O algoritmo mestre desta tribo é a Máquina de Vetores de Suporte.
- Simbolistas: Esta tribo encara a aprendizagem como o inverso da dedução e se inspiram na filosofia, na psicologia e na lógica. O algoritmo mestre desta tribo é a dedução inversa.
- Conexionistas: Eles procuram realizar a engenharia reversa do cérebro e são inspirados pela neurociência e pela física. Seu algoritmo mestre é a retro-propagação.
- Bayesianos: Eles acreditam que a aprendizagem é uma forma de inferência probabilística e suas raízes se encontram na estatística. Seu algoritmo mestre é a inferência Bayesiana.
- Evolucionários: Eles simulam a evolução no computador e suas ideias derivam da genética e da biologia evolucionária. Seu algoritmo mestre é a programação genética.

A Tabela 1-1 (DOMINGOS, 2017) resume as diferenças entre as tribos em relação à representação do conhecimento, da forma de avaliação dos erros, e da maneira como seus modelos são otimizados.

Tabela 1-1 Principais diferenças entre tribos de aprendizagem de máquinas

Critério	**Analogistas**	**Simbolistas**	**Conexionistas**	**Bayesianos**	**Evolucionários**
Otimização	Otimização Restrita	Dedução Inversa	Descida de Gradiente	Inferência Probabilística	Busca Genética
Avaliação	Margem	Precisão	Erro Quadrático	Probabilidade a Posteriori	Adaptabilidade
Representação	Máquina de Vetores de Suporte	Lógica	Redes Neurais	Modelos Gráficos	Programas Genéticos

Fonte: adaptado de Domingos (2017)

A Adaptabilidade aparece nesta tabela como uma maneira de se avaliar diversos programas concorrentes a fim de se determinar qual melhor se adaptou a um determinado ambiente. Conforme o método empregado pela natureza descrito por Darwin, os evolucionistas determinam que o programa mais apto, com melhor desempenho, deve gerar novas versões que serão cada vez melhores.

O objetivo desta tese é a aplicação das tecnologias adaptativas de uma forma distinta da acima descrita e em áreas diferentes das dos evolucionários, observando que o termo adaptatividade tem sido utilizado na engenharia ao longo dos anos e seu conceito tem variado dependendo da área onde este termo é empregado.

Em certas áreas da engenharia, um dispositivo é denominado adaptativo quando algum parâmetro que rege seu comportamento pode ser modificado durante a operação do mesmo. Para estas áreas, o acerto ou ajuste dos parâmetros durante sua operação torna o dispositivo adaptável ao meio.

Em outras áreas, um dispositivo é denominado adaptativo quando o conjunto de regras que determina seu comportamento é modificado, seja pela inclusão de novas regras, seja pela exclusão de regras antigas. Para estas áreas, é a mudança do conjunto de regras durante sua operação que torna o dispositivo adaptável ao meio ou ao conjunto de dados sobre os quais opera.

No Laboratório de Linguagens e Técnicas Adaptativas (LTA) a visão adotada é a da adaptação através da modificação do conjunto de regras. Um dos trabalhos fundamentais do LTA é a definição de dispositivos dirigidos por regras de Neto (NETO, 2001). Neste trabalho, Neto envolve um dispositivo convencional com uma camada adaptativa que coloca em evidência o controle do dispositivo convencional.

O que é posto em evidência é o conjunto de regras responsável pela operação do dispositivo. Se novas regras são adicionadas a este conjunto de regras como resposta aos dados analisados, o dispositivo sofre uma adaptação aos dados e ao meio que produziu estes dados.

Na área de Redes Neurais Artificiais o termo Adaptatividade é utilizado para a modificação dos pesos sinápticos de uma dada rede e também para a modificação das conexões, ou seja, de sua arquitetura. Estes pesos são valores que definem o grau de influência de uma entrada ou de outra variável interna na decisão ou resposta de um neurônio artificial.

Existem diversos tipos de Redes Neurais Artificiais, com ou sem recorrência nas conexões, mas para a maioria delas sua estrutura ou arquitetura é fixa e a adaptatividade fica por conta da modificação da força de suas conexões que são definidas pelos pesos sinápticos.

Em virtude dessa diferença conceitual sobre adaptatividade, as Redes Neurais Artificiais não têm despertado muito interesse na pesquisa do LTA. Entretanto, ao longo dos anos, diversas pesquisas estabeleceram equivalências entre alguns tipos de redes neurais e alguns sistemas baseados em regras especialmente sistemas de inferência nebulosa.

Uma destas equivalências foi demonstrada por Jang (JANG, 1993) entre Redes Neurais de Funções de Base Radial (RBF - Radial Basis Function) e Sistemas de Inferência Nebulosa, que são sistemas baseados em regras, ou seja, alterações do conjunto de regras em um Sistema de Inferência Nebulosa seriam equivalentes a mudanças na arquitetura ou nos pesos das Redes Neurais de Funções de Base Radial e vice-versa.

Do ponto de vista da Inteligência Computacional, estas equivalências relacionam duas de suas áreas denominadas Computação Neural (CN) e Computação Nebulosa ou Fuzzy (CF). A terceira área da Inteligência Computacional é a Computação Evolutiva (CE) (ANDRADE, 2002) (CAVERSAN, 2006).

As Redes Neurais Artificiais são utilizadas com sucesso em diversas áreas da engenharia, mas ainda são criticadas pelos especialistas de diversas outras por sua falta de transparência.

Os especialistas em alguma área médica querem saber, por exemplo, por que o diagnóstico d_1 foi produzido como resposta aos dados fornecidos e não o diagnóstico d_2 para esta situação. De certa forma, na alegoria de Domingos citada acima, são simbolistas questionando conexionistas o significado de seus resultados.

Para resolver este problema, pontes entre estas áreas foram criadas na forma de diversas técnicas denominadas de “Extração/Inserção de Conhecimento” ou “Extração/Inserção de Regras” nos últimos vinte e cinco anos a fim de tornar mais transparentes para os simbolistas os mecanismos que comandam as Redes Neurais Artificiais.

A “Inserção de Conhecimento” é a tarefa inversa de “Extração do Conhecimento”. Seu objetivo é a modificação de uma Rede Neural Artificial ou de um Sistema de Inferência Nebulosa a partir dos conhecimentos de um especialista em um dado ramo.

Normalmente, o especialista é capaz de estabelecer regras às vezes de forma precisa e outras vezes de forma imprecisa ou nebulosa sobre sua área de especialização. A inclusão destes conhecimentos na rede neural supera o problema da falta de dados para treinamento da mesma rede.

A figura 1-1 representa estas transformações de forma gráfica.

-

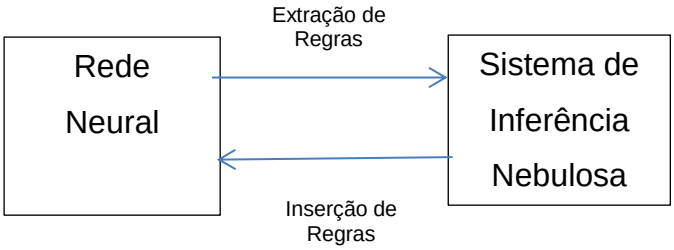


Figura 1-1 Extração e Inserção de Regras em uma Rede Neural a partir de um Sistema de Inferência Nebulosa

No diagnóstico de falhas incipientes de transformadores de potência, Castro (CASTRO, 2004) desenvolveu uma metodologia para a extração transparente de regras nebulosas a partir de uma rede neural artificial treinada com os dados das falhas dos transformadores.

Nessa mesma área de conhecimento, Silva (SILVA, 2013) extraiu regras difusas (nebulosas) a partir de Mapas Auto-Organizáveis de Kohonen (Self-Organizing Maps – SOM) (KOHONEN, 1982).

Nos dois casos acima mencionados, o conjunto de regras nebulosas constitui um conhecimento considerado inteligível por parte dos especialistas da área, produzindo confiança nos resultados dos diagnósticos de falhas, ao contrário dos resultados apresentados diretamente da rede neural treinada.

1.2 Objetivos

O primeiro objetivo desta tese é pesquisar as aplicações das redes neurais artificiais e das redes de Petri com a finalidade de realçar seus pontos fortes e mapear alternativas que poderiam ser implementadas através do uso de tecnologias adaptativas, tornando-as mais flexíveis do ponto de vista das aplicações.

O segundo objetivo é a definição de um dispositivo adaptativo capaz de estabelecer uma estrutura comum para as redes de Petri e as redes neurais artificiais. Trata-se da combinação de redes de Petri de alto nível com capacidade de auto-modificação ou de reconfiguração. A partir destas redes de Petri reconfiguráveis,

implementações de redes neurais artificiais e de sistemas de apoio à decisão com controle flexível são propostos e exemplos de aplicação são apresentados.

1.3 Estrutura do trabalho

No segundo capítulo desta tese a redes neurais artificiais e os mecanismos de extração e inserção de conhecimentos são analisados com ênfase na adaptatividade apresentada.

No terceiro capítulo as redes de Petri ordinárias e de alto nível são apresentadas com ênfase nas versões reconfiguráveis.

No quarto capítulo a versão de uma rede de Petri colorida adaptativa de alto nível e a metodologia de sua utilização são propostos e exemplos de aplicações são apresentados e discutidos.

Finalmente, no quinto capítulo as conclusões deste trabalho são apresentadas e propostas para trabalhos futuros são indicadas.

2 FLEXIBILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS

2.1 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é um modelo computacional com inspiração biológica constituída de Neurônios Artificiais interligados com o objetivo de implementar funções complexas como, por exemplo, reconhecimento de padrões, classificação de dados, aproximação de funções, previsão, otimização e controle.

2.1.1 Rede Perceptron e Rede Perceptron multicamadas

Em 1958 Rosenblatt criou o modelo denominado Perceptron, também conhecido como Neurônio Artificial (SILVA, SPATTI e FLAUSINO, 2010), que se mostrou capaz de reconhecer padrões geométricos como letras e números.
Na Figura 2-1 está esquematizada a estrutura genérica de um Neurônio Artificial com n entradas (X_1, X_2, ..., X_n) e uma saída (Y).

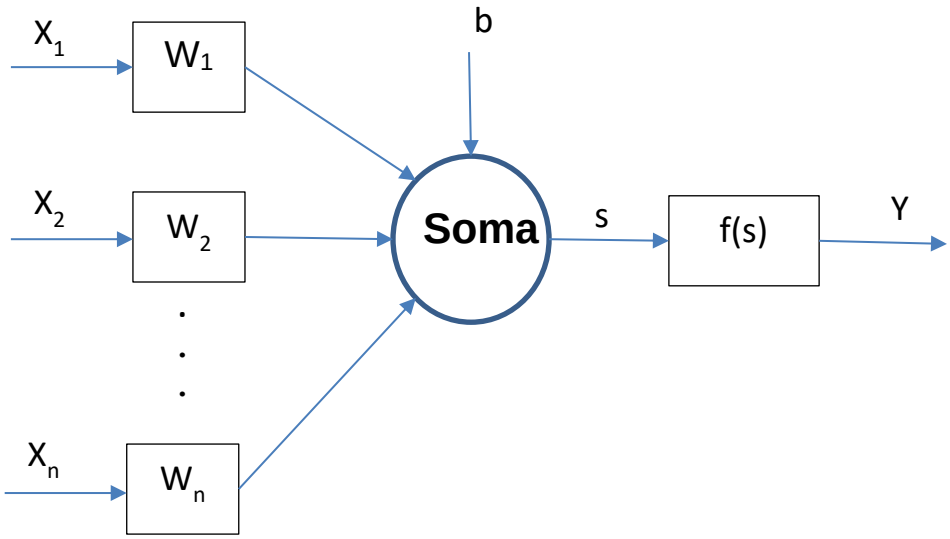


Figura 2-1 Neurônio Artificial

Matematicamente o Neurônio Artificial é descrito como:

$$s = b + \sum_{i=1}^{n} W_i * X_i \quad (2.1)$$

$$Y = f(s) \quad (2.2)$$

W_i são os pesos sinápticos associados às entradas.
b é uma constante de polarização da soma ponderada.

O resultado da soma ponderada de todas as entradas e da constante de polarização é submetido à função f chamada de função de ativação.

A função de ativação f pode ser a função degrau de Heaviside (eq 2.3), pode ser a função sinal (eq. 2.4), a função rampa simétrica (eq.2.5), a função logística (eq. 2.6) ou a função tangente hiperbólica (eq. 2.7).

$$f(s) = \begin{Bmatrix} 1\ se\ s \geq 0 \\ 0\ se\ s < 0 \end{Bmatrix} \quad \text{função degrau de Heaviside} \tag{2.3}$$

$$f(s) = \begin{Bmatrix} 1\ se\ s > 0 \\ 0\ se\ s = 0 \\ -1\ se\ s < 0 \end{Bmatrix} \quad \text{função sinal} \tag{2.4}$$

$$f(s) = \begin{Bmatrix} 1\ se\ s > a \\ s\ se - a \leq s \leq a \\ -1\ se\ s < a \end{Bmatrix} \quad \text{função rampa simétrica} \tag{2.5}$$

$$f(s) = \frac{1}{1+e^{-ks}} \quad \text{função logística} \tag{2.6}$$

$$f(s) = \frac{1-e^{-ks}}{1+e^{-ks}} \quad \text{função tangente hiperbólica} \tag{2.7}$$

A saída da função de ativação é igual a 1 quando o neurônio dispara e é igual a 0 quando o neurônio não dispara.

Em 1969, Minsky e Papert (MINSKY e PAPERT, 1969) publicaram um estudo no qual demonstraram que o Perceptron de Rosenblatt não poderia resolver o problema do OU exclusivo.

Este problema foi resolvido através da criação do modelo de Perceptron Multicamadas (PMC). Neste modelo, entre a camada formada pelas entradas de dados e a camada formada pelas saídas foram introduzidas novas camadas de neurônios artificiais denominadas camadas escondidas que possibilitaram uma partição mais complexa do espaço das entradas.

Se os neurônios artificiais estão interligados formando uma rede neural com uma única camada, esta rede é denominada de Perceptron de Camada Única. Se uma ou mais camadas intermediárias são introduzidas na rede, mas sem ligações realimentadas (redes feedforward) esta rede é denominada de Perceptron Multicamadas e se a rede neural possuir múltiplas camadas com ligações recorrentes elas são denominadas de redes recorrentes.

A figura 2-2 apresenta uma rede neural artificial tipo Perceptron Multicamadas com uma camada escondida.

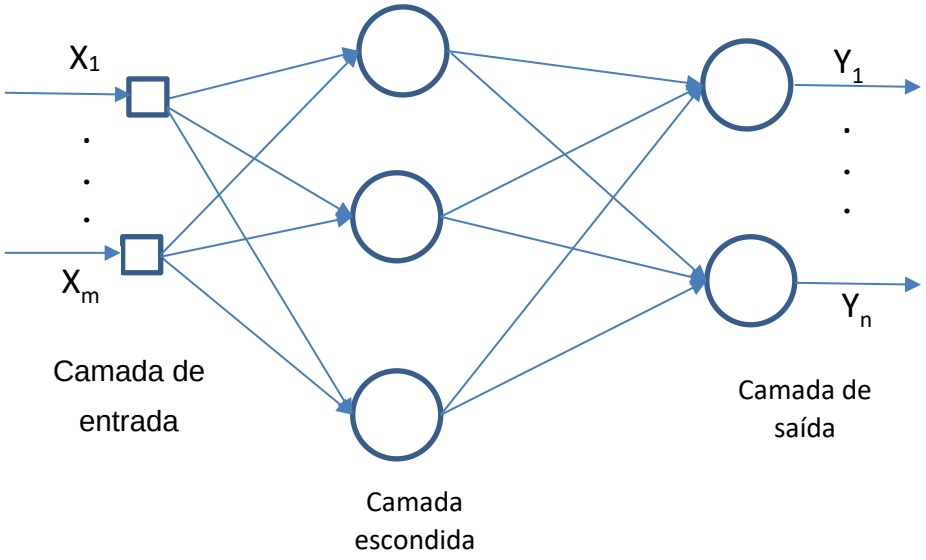


Figura 2-2 Rede Neural Artificial Feedforward com uma camada escondida

Para exemplificar a solução do problema do OU exclusivo com a inclusão de uma camada escondida apresenta-se a seguir a implementação das funções lógicas E e OU no Perceptron de camada única e a função lógica OU exclusivo no Perceptron Multicamadas.

A Figura 2-3 apresenta a função lógica OU entre as variáveis X_1 e X_2 .
A reta $X_1 + X_2 = 0{,}5$ divide o espaço das variáveis X_1 e X_2 em duas regiões, a região A e a região B.

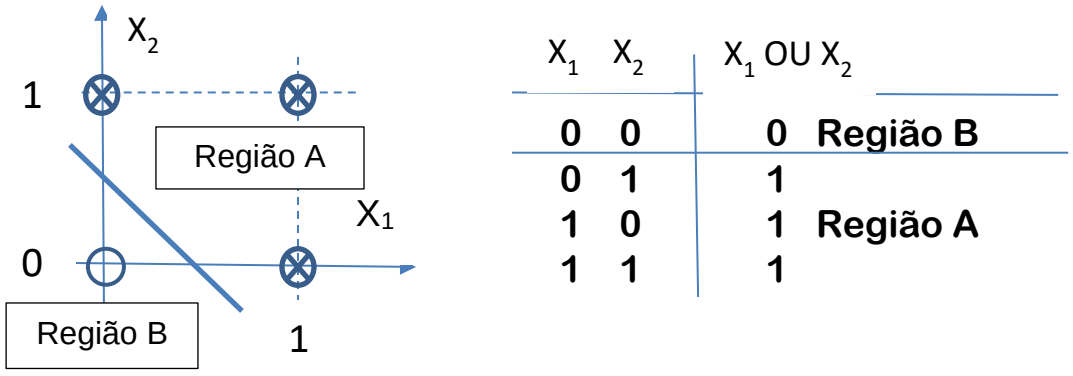


Figura 2-3 Função lógica OU linearmente separável

A região A é definida pela inequação $X_1 + X_2 > 0{,}5$ e a região B é definida pela inequação $X_1 + X_2 < 0{,}5$.

A Figura 2-4 mostra um neurônio artificial no qual os pesos sinápticos e a constante de polarização foram escolhidos de tal forma que as regiões A e B da Figura 2-3 são separadas pela reta $X_1 + X_2 = 0{,}5$.

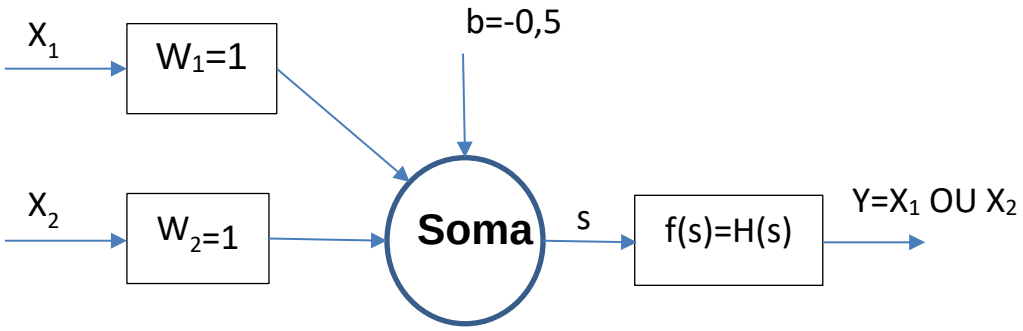


Figura 2-4 Neurônio artificial que implementa a função lógica OU

A função H(s) é a função degrau de Heaviside e $s = X_1 + X_2 - 0{,}5$.
Se $s > 0$ (região A) então $Y = 1$.
Se $s < 0$ (região B) então $Y = 0$.
Observe que infinitas retas conseguem separar a região com resultados iguais a 1 da região com resultado igual a 0, ou seja, infinitas combinações de pesos sinápticos e de constantes de polarização poderiam ser utilizadas.
De forma análoga, a Figura 2-5 apresenta o espaço das variáveis X_1 e X_2 sendo dividido em duas regiões pela reta $X_1 + X_2 = 1{,}5$.

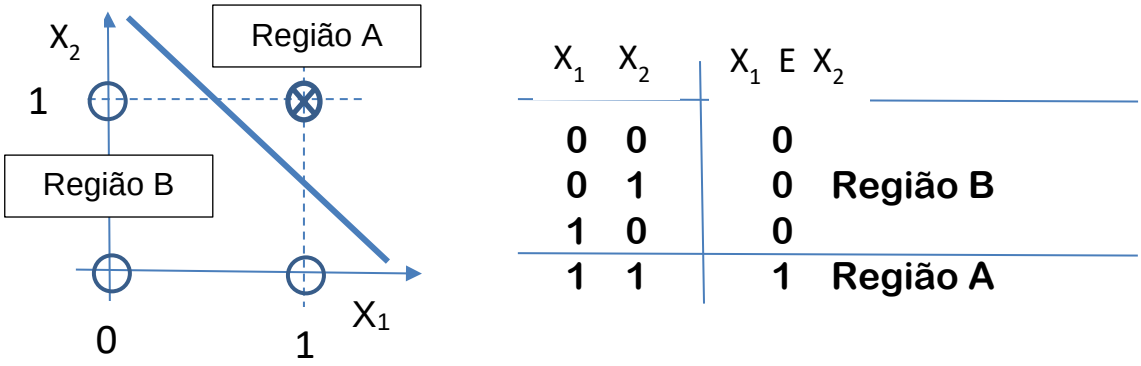


Figura 2-5 Função lógica E linearmente separável

A Figura 2-6 esquematiza um neurônio artificial no qual a reta de separação entre as regiões A e B é dada por $s = X_1 + X_2 - 1{,}5$ e a função de ativação é o degrau de Heaviside.

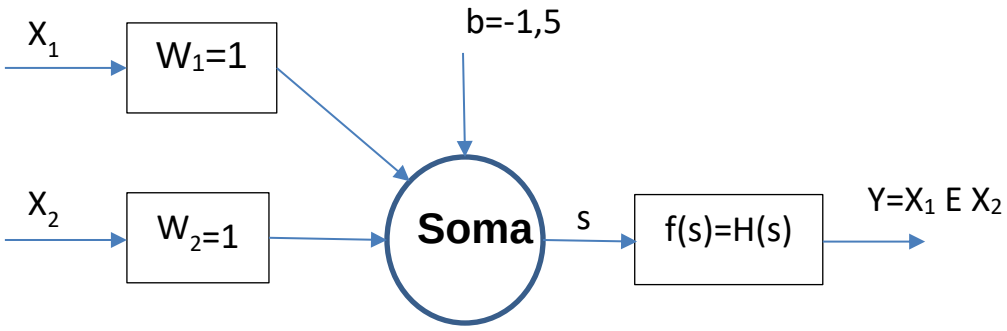
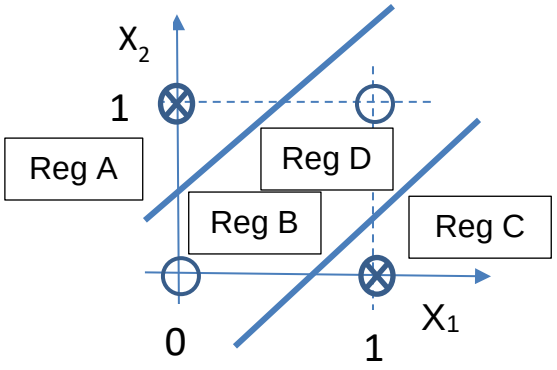


Figura 2-6 Neurônio artificial que implementa a função lógica E.

Se $s > 0$ (região A) então $Y = 1$.
Se $s < 0$ (região B) então $Y = 0$.

Analogamente ao caso anterior, existem infinitas retas que são capazes de separar os resultados 0's dos resultados 1's, ou seja, existem infinitas combinações de pesos sinápticos de constates de polarização capazes de produzir o mesmo resultado lógico da função E.
A Figura 2-7 apresenta a função lógica OU Exclusivo e três regiões distintas com os resultados da função.
Não existe uma reta capaz de separar os 0's dos 1's, indicando que um neurônio artificial é incapaz de implementar a função OU Exclusivo.



X_1	X_2	X_1 OEx X_2	
0	0	0	Região B ou D
0	1	1	Região A
1	0	1	Região C
1	1	0	Região B ou D

Figura 2-7 Função lógica OEx (OU exclusivo)

Para resolver este problema, uma nova camada de neurônios artificiais é necessária para dividir o espaço das entradas X_1 e X_2 .

A reta $-X_1 + X_2 = 0{,}5$ separa as regiões A e B enquanto que a reta $X_1 - X_2 = 0{,}5$ separa as regiões C e D.

A região A é descrita pela inequação $s = -X_1 + X_2 - 0{,}5 > 0$.

A região B é descrita pela inequação $s = -X_1 + X_2 - 0{,}5 < 0$.

A região C é descrita pela inequação $s = X_1 - X_2 - 0{,}5 > 0$.

A região D é descrita pela inequação $s = X_1 - X_2 - 0{,}5 < 0$.

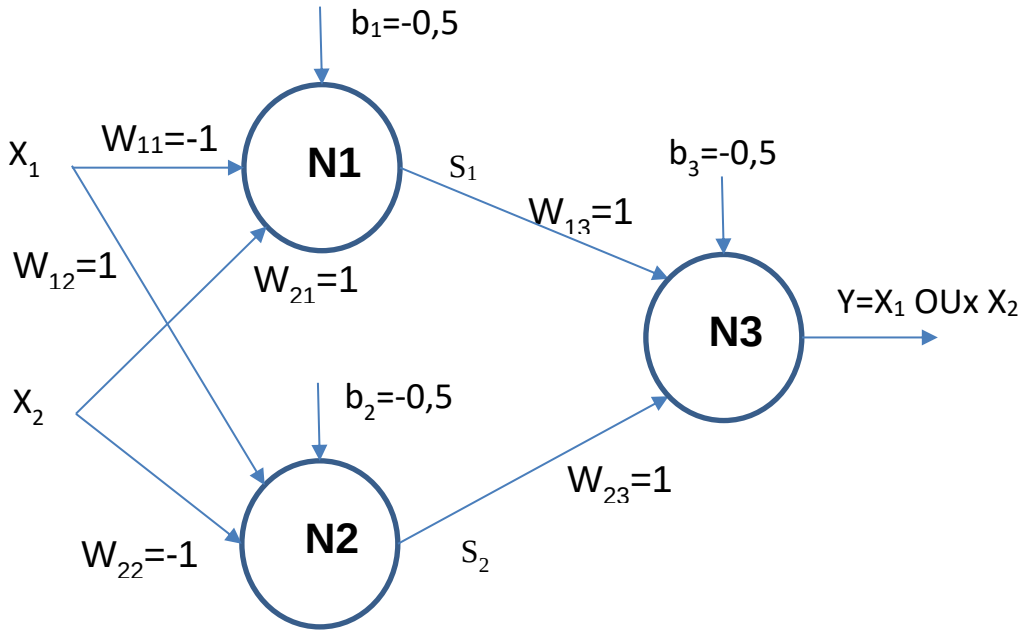


Figura 2-8 Função lógica OU Exclusivo (OUx) implementado com três neurônio artificiais

A Figura 2-8 apresenta a função OU Exclusivo implementada por três neurônios artificiais. O neurônio N1 separa a região A da região B, gerando um 1 para a região A e um 0 para a região B.

O neurônio N2 separa a região C da região D, gerando um 1 para a região C e um 0 para a região D.

O neurônio N3 realiza a função OU entre os resultados do neurônio N1 e do neurônio N2, gerando um 1 para as regiões A e C e um 0 para a intersecção das regiões B e D.

Em termos de regras, a função OU Exclusivo é descrita por:

Se $(-X_1 + X_2 - 0{,}5 > 0)$ OU $(X_1 - X_2 - 0{,}5 > 0)$ então Y=1, caso contrário, Y=0.

2.1.2 Rede de Função de Base Radial - RBF (Radial Basis Function)

A rede RBF difere de muitas maneiras da rede PMC, mas a mais importante é a maneira como cada rede divide o espaço constituído pelas variáveis de entrada para a classificação dos sinais.

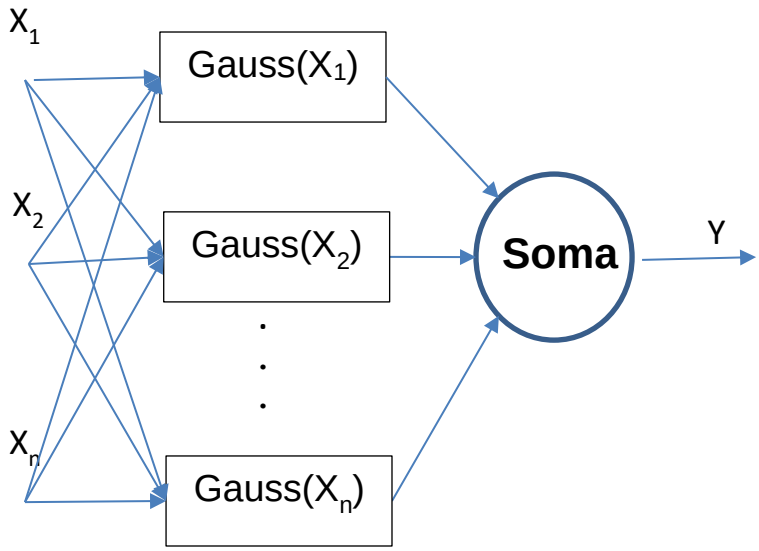
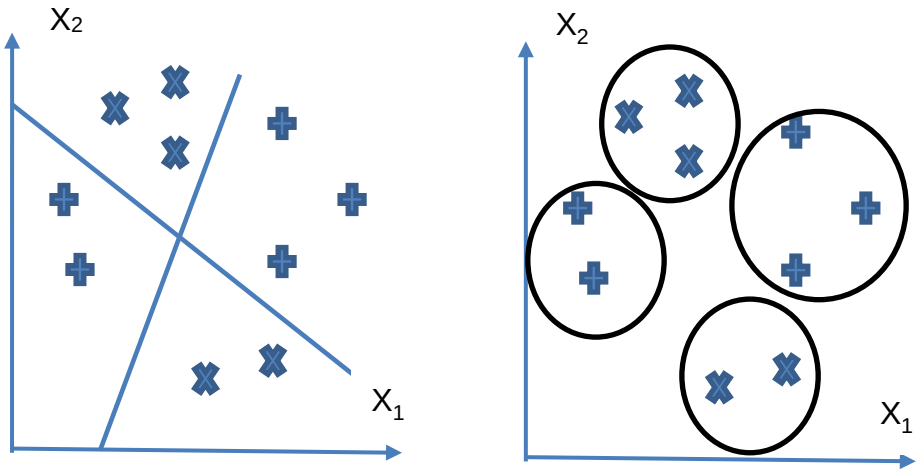


Figura 2-9 Esquema simplificado de uma rede RBF

A Figura 2-9 mostra n entradas X_n acionando n funções gaussianas cujas saídas são somadas para produzir a saída Y.



a) Rede PMC b) Rede RBF

Figura 2-10 Partição do espaço de entradas

Enquanto as redes PMC partilham o espaço de entradas através de retas, planos ou hiperplanos dependendo da quantidade de variáveis de entrada, as redes RBF estabelecem regiões de sensibilidade diferenciada denominados de campos receptivos (ver Figura 2-10).
Na Figura 2-10-b, os círculos representam a área cuja resposta é uma gaussiana na qual o centro do círculo representa o ponto de maior sensibilidade (resposta igual a 1) e a circunferência de raio σ representa o conjunto de pontos no qual a resposta decaiu para aproximadamente 60% do máximo. O valor de σ representa o desvio padrão da gaussiana.

2.2 Extração de regras ou conhecimento das Redes Neurais Artificiais

O trabalho pioneiro na área de extração de regras a partir de Redes Neurais Artificiais foi publicado por Gallant (GALLANT, 1988) que apresentou uma rotina para extrair regras proposicionais de uma rede simples. Depois deste trabalho, diversos autores apresentaram algoritmos que representam uma RNA como um conjunto de regras, tanto regras da lógica proposicional como regras da lógica nebulosa.

Na definição de Craven (CRAVEN, 1996), a tarefa de extração de regras consiste em produzir, para uma dada rede neural treinada, uma descrição da hipótese da rede que seja compreensível e que se aproxime do comportamento previsível da rede.

Segundo Castro (CASTRO, 2004), além da explicação do funcionamento de uma rede neural, a extração de regras possui outras vantagens listadas a seguir:

- Descoberta de características importantes da entrada;
- Melhoria na generalização da RNA;
- Descoberta de conhecimento;
- Aquisição de conhecimento para sistemas especialistas;
- Validação do conhecimento adquirido.

Andrews et al (ANDREWS, DIEDERICH e TICKLE, 1995) definiram uma taxonomia para avaliar os algoritmos de extração de regras que é a mais utilizada pelos pesquisadores da área. Esta taxonomia engloba cinco critérios de classificação:

1. O poder de expressão das regras extraídas. Três grupos de regras são sugeridos:
    - Regras simbólicas convencionais (booleana, proposicional);
    - Regras baseadas em conjuntos nebulosos e lógica nebulosa;
    - Regras expressas em lógica de primeira ordem.

2. A qualidade da regra extraída. Quatro medidas de qualidade são sugeridas: a fidelidade, a precisão, a consistência e a compreensibilidade.

3. A transparência. Este critério classifica a técnica de extração de regras em “decomposta” (ou caixa branca), em “pedagógica”(ou caixa preta) e “eclética”.
    - A técnica decomposta considera a extração de regras como um processo de busca que mapeia a estrutura interna de uma rede neural treinada em um conjunto de regras. A extração de regras analisa os valores numéricos da rede tais como os valores de ativação dos neurônios escondidos e dos neurônios de saída, e os pesos das conexões entre os neurônios e forma as regras a partir desses valores. As regras são extraídas separadamente para cada neurônio escondido e para cada neurônio de saída, e o sistema de regras da rede total é obtido em um processo separado de reescrita do conjunto de regras.
    - A técnica pedagógica não desmonta a arquitetura da rede neural treinada, mas considera a RNA como uma “caixa preta” e as regras extraídas descrevem a relação global entre as variáveis de entrada e as variáveis de saída.
    - A técnica eclética incorpora elementos tanto da técnica decomposta quanto da técnica pedagógica.

4. Complexidade do Algoritmo. A quantidade de cálculos necessários para a realização da tarefa (complexidade temporal) e a quantidade de memória utilizada (complexidade espacial) são geralmente usadas como medida de eficiência do algoritmo. A complexidade temporal é o fator mais importante uma vez que os algoritmos de extração frequentemente se baseiam em testes de um grande número de combinações de entradas e parâmetros da rede tais como: número de camadas da RNA, quantidade de neurônios por camada, conexões entre camadas, quantidade de exemplos de treinamento, quantidade de atributos de entrada e valores por atributo de entrada.

5. Portabilidade ou generalidade. Este critério avalia a técnica de extração de regras sob o ponto de vista da aplicabilidade do algoritmo considerando as diversas arquiteturas de RNA e de sistemas de treinamento.

2.2.1 Regras Booleanas de extração usando a abordagem por decomposição

Os primeiros trabalhos nesta área focalizaram na busca e extração de regras booleanas convencionais no nível das unidades individuais no interior das RNAs treinadas. A ideia básica é a busca por conjuntos de pesos contendo uma única conexão de valor positivo suficiente para garantir que a polarização da unidade em análise seja ultrapassada independentemente dos valores das outras conexões.

Se uma conexão que satisfaz este critério for encontrada, então ela é escrita como uma regra. A busca continua para os subconjuntos de dois elementos e assim sucessivamente, sendo as regras extraídas no nível das unidades individuais agrupadas para formar uma base de regras compostas que valem para a RNA como um todo.

Um esquema do algoritmo “Subset” desenvolvido por Towell e Shavlik (TOWELL e SHAVLIK, 1993) é o seguinte:

Para cada unidade das camadas escondida ou da saída:
Extrair até S_P subconjuntos das ligações com pesos positivos para os quais a somatória dos pesos é maior que o valor de polarização da unidade;
Para cada elemento p dos S_P subconjuntos:
Procurar por um conjunto S_N de um conjunto de atributos negativos tal que a somatória dos pesos de p mais a somatória dos pesos de N-n (onde N é o conjunto de todos os atributos negativos e n é um elemento de S_N) excedem o limiar na unidade;
Com cada elemento n do conjunto S_N , formar a regra: "Se p e Não n, então é verdadeiro o conceito designado pela unidade".

O problema com este tipo de algoritmo é que o tempo para descoberta de todos os possíveis subconjuntos cresce exponencialmente com o tamanho da rede.
Um segundo exemplo de técnica por decomposição é o algoritmo M-of-N (TOWELL e SHAVLIK, 1993). O conceito do algoritmo M-of-N é a formação de regras do tipo:
Se (M dos seguintes N antencedentes for VERDADEIRO) então ...

As fases do algoritmo M-of-N são:

Passo 1 : Gerar uma RNA e treiná-la usando a retropropagação. Formar grupos de pesos similares para cada unidade escondida e de saída.
Passo 2: Tornar todos os pesos das conexões de todos os membros de um grupo iguais à media do grupo.
Passo 3: Eliminar todos os grupos que não afetam de modo significativo a saída da unidade.
Passo 4: Mantendo todos os pesos das conexões constantes, otimizar as polarizações (biases) de todas as unidades escondidas e de saídas através do algoritmo de retropropagação.
Passo 5: Criar uma única regra para cada unidade escondida e para cada unidade de saída. A regra consiste em um limiar dado pelas polarizações e pelos pesos dos antecedentes especificados pelas

2.2.2 Extração de Regras Booleanas usando a abordagem “Pedagógica”

Um exemplo da técnica pedagógica de extração de regras foi desenvolvido por Thrun (THRUN, 1993) e denominado VIA (Validity Interval Analysis – Análise do Intervalo de Validade). As entradas são mapeadas diretamente para as saídas. O algoritmo usa o procedimento de geração-e-teste para extrair regras simbólicas a partir de RNAs treinadas com retropropagação.

Os passos básicos do algoritmo VIA são os seguintes:

Passo 1: Atribuir intervalos arbitrários para todas as unidades de uma RNA. Estes intervalos constituem as restrições dos valores das entradas e dos valores de ativação das saídas.

Passo 2: Refinar os intervalos através de iterativamente detectar e excluir os valores de ativação que são provavelmente inconsistentes com os pesos e polarizações (biases) da rede.

Passo 3: O resultado do Passo 2 é um conjunto de intervalos que são considerados consistentes ou inconsistentes. Um intervalo é dito inconsistente se não existe um padrão de ativação que pode satisfazer as restrições impostas pelos intervalos de validade iniciais.

Segundo Thrun, seu método é parecido com uma análise de sensibilidade, pois ele caracteriza uma saída de uma RNA treinada através de uma variação sistemática dos padrões de entrada e do exame das mudanças de classificação efetuada pela RNA, ou seja, quão sensível é o classificador para as mudanças na entrada.

Outro exemplo da abordagem “Pedagógica” é o algoritmo RULENEG, que foi desenvolvido por Pop et al (POP, HAYWARD e DIEDERICH, 1994). O foco deste algoritmo é a extração de regras conjuntivas (regras tipo E). A origem desta técnica é a observação de que cada regra simbólica no cálculo proposicional pode ser expressa como uma disjunção (função lógica OU) de conjunções (funções lógicas E).

Além disso, uma regra conjuntiva é válida somente quando todos os antecedentes na regra são VERDADEIROS, ou seja, modificando o valor de um dos antecedentes, o consequente da regra se modifica. Dado uma RNA treinada e os padrões usados na fase de treinamento, as regras aprendidas pela RNA são extraídas de acordo com o seguinte algoritmo:

```
Inicializar o conjunto de regras como vazio.
  Para cada padrão s do conjunto de treinamento
     Encontrar a classe C para s usando a RNA; “C=RNA(s)”
     Se s não é classificada pelas regras existentes
        Inicializar uma nova regra r para a classe C
        Para cada entrada i da rede
       Fazer uma cópia s’ de s
       Negar a i-esima entrada em s’
       Encontrar a classe C’ para s’ usando a RNA; “C’=RNA(s’)”
       Se C não é igual  a C’
              Incluir a i-esima entrada e seu valor a r
      Fim do laço para cada entrada
       Adicionar a regra r ao conjunto de regras
  Fim do laço para cada padrão
```

RULENEG foi projetado para extrair somente uma regra conjuntiva por padrão de entrada, mas ele é capaz de extrair todas as regras aprendidas com os padrões.

2.2.3 Extração de Regras Nebulosas

Em paralelo com o desenvolvimento das técnicas para extração de regras booleanas das RNAs, foram desenvolvidas técnicas para extrair regras nebulosas dos sistemas denominados neuro-fuzzy. Um sistema neuro-fuzzy típico compreende três componentes distintos.

O primeiro é um conjunto de procedimentos para inserir o conhecimento de um especialista na forma de regras nebulosas dentro de uma estrutura de RNA (fase de inicialização). A maior diferença nesta etapa é a geração de representações que correspondem às funções de pertinência.

O segundo componente é o processo de treinamento da RNA. Neste caso, o foco está na sintonia das funções de pertinência de acordo com os padrões dos dados de treinamento.

O terceiro componente do processo é a análise e a extração do conhecimento refinado embutido na forma de um conjunto de funções de pertinência modificadas. Entre os diversos trabalhos nesta área, podemos citar o FALCON (Fuzzy Adaptive Learning Control Network) (HORIKAWA, FURUHASHI e UCHIKAWA, 1992), o ANFIS (Adaptive Network-based Fuzzy Inference System) (JANG, 1993) , o SIN (Sistema de Inferência Nebulosa, 1994) (ANDRADE, 2002), o NEFCON (Neuro-Fuzzy Control) (NAUCK e KURSE, 1997), o EFuNN (Evolving Fuzzy Neural Networks) (KASABOV, 1998) .

O FALCON é uma RNA de 5 camadas, distribuídas da seguinte forma:
Camada 1 – Contêm os neurônios das Entradas Linguísticas;
Camada 2 – Contêm os neurônios dos Termos de Entrada (fazem a Fuzzificação). Cada neurônio representa uma função de pertinência;
Camada 3 – Contêm os neurônios das Regras Nebulosas;

Camada 4 – Contêm os neurônios dos Termos de Saída, os consequentes das regras;

Camada 5 – Contêm os neurônios das Saídas Linguísticas.

O procedimento de operação da rede é o seguinte:

- Treinamento dos dados;
- Determinação dos centros e das larguras das funções de pertinência através do algoritmo de agrupamento auto-organizado;
- Determinação das regras nebulosas através do aprendizado competitivo;
- Eliminação de regras redundantes;
- Combinação de regras;
- Determinação das funções de pertinência ótimas através do algoritmo de retro-propagação.

O ANFIS de Jang também possui 5 camadas, mas a ideia é a implementação de um Sistema de Inferência Nebulosa tipo Takagi-Sugeno (JANG, 1993).

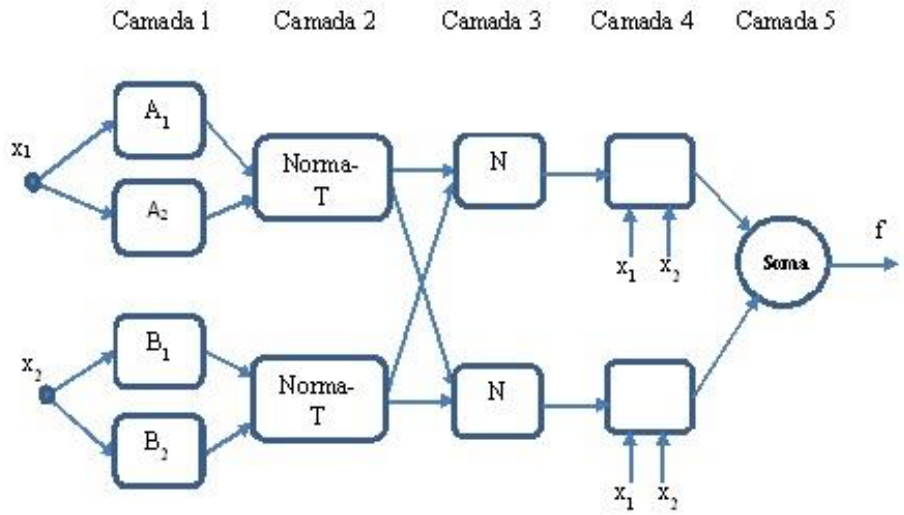


Figura 2-11– Arquitetura do ANFIS

A fuzzificação das variáveis de entrada é realizada na primeira camada;

A segunda camada calcula o antecedente da regra utilizando os operadores da norma-T (operação E difuso, que pode ser a operação MIN(mínimo) ou multiplicação;

A terceira camada normaliza os pesos da regra;

A quarta camada determina os parâmetros consequentes da regra;

A quinta camada, ou camada de saída, soma todos os sinais da camada anterior.

Para determinar os parâmetros da premissa (relacionados com as funções de pertinência), ANFIS utiliza o aprendizado por retro-propagação e para determinar os parâmetros do consequente, é utilizado o método dos mínimos quadrados.
Cada iteração no processo de aprendizagem do ANFIS possui duas partes:
Na primeira parte os padrões de entrada são propagados e os parâmetros ótimos do consequente são estimados por um procedimento iterativo de mínimos quadrados, enquanto que os parâmetros da premissa são considerados fixos para o ciclo atual de treinamento;
Na segunda parte, os padrões são novamente propagados e agora os parâmetros da premissa são modificados via aprendizado por retro-propagação, enquanto que os parâmetro da parte consequente permanecem fixos.
O procedimento de aprendizado do ANFIS não possui um mecanismo que restrinja o tipo de modificação aplicado às funções de pertinência.

2.2.4 Extração de Regras a partir de Mapas Auto-Organizáveis (SOM – Self-Organizing Maps)

Em trabalho de 2010, Hung (HUNG, 2010) apresentou uma forma de extrair conhecimento de uma RNA, na forma de regras simbólicas, a partir de Mapas Auto-Organizáveis (SOM – Self-organizing feature map) de uma dimensão.

Os Mapas Auto-Organizáveis (SOM) (KOHONEN, 1982) são um modelo de agrupamento neural não-supervisionado. A maioria das outras pesquisas extraem regras a partir de SOM de duas dimensões, mas estes modelos normalmente produzem grupamentos de limites fechados.

Ultsch e Korus (ULTSCH e KORUS, 1995) propuseram um sistema de extração de regras denominado REGINA, que utiliza uma matriz de distância unificada denominada de Matriz-U (ULTSCH e SIEMON, 1990) para detectar e apresentar grupamentos formados por SOM bidimensional treinado. Este método calcula o vetor médio dos vetores unitários da vizinhança.

O SOM treinado é transformado em uma paisagem de “colinas” e “vales” baseado no algoritmo de bacia hidrográfica (VINCENT e SOILLE, 1991). Todas as amostras de entrada que caem em uma bacia comum pertencem a um mesmo grupamento

por sua similaridade e, deste modo, as fronteiras de decisão são encontradas . O maior problema deste método resulta na dificuldade do algoritmo de bacia hidrográfica no refinamento das bacias, que é feito por tentativa e erro.

O método de Hung (HUNG, 2009) é dividido em três tarefas: a primeira tarefa consiste na descoberta de limites de cada grupamento, a segunda tarefa consiste na seleção de atributos significativos para cada grupamento e a terceira tarefa consiste na geração de regras simbólicas para cada grupamento.
Na primeira tarefa, a questão é a definição do rótulo conceitual para cada unidade de saída do SOM. Hung utiliza a informação pré-rotulada de modo a incluir unidades vizinhas com o mesmo rótulo dominante em um mesmo grupamento. Na seleção de atributos significativos para cada grupo, é definido um valor limiar para caracterizar a importância do atributo e o cálculo efetuado é a diferença entre a matriz unificada do atributo e a do grupo.

A geração de regras para cada grupo da terceira etapa leva em conta a importância de cada regra de acordo com o número de atributos significativos. Em função dessa importância, as bordas de cada agrupamento são ajustadas. Silva apresenta um método para a extração de regras nebulosas a partir dos Mapas Auto-Organizáveis de Kohonen (SILVA, 2013).

A aplicação proposta para teste do método é o Diagnóstico de Faltas Incipientes em Transformadores. Esta mesma aplicação foi realizada por Castro (CASTRO, 2004), mas a extração de regras partiu de uma Rede Restrita de Perceptrons de Múltiplas Camadas. Pelo método de Castro denominado TFRENN (Transparent Fuzzy Rule Extraction from Neural Networks), a extração de regras para o Diagnóstico de Faltas Incipientes em Transformadores produziu um conjunto de 95 regras nebulosas e 100% de acerto em relação aos dados da base de testes .

Pelo método proposto por Silva, para o mesmo conjunto de dados utilizados por Castro, obteve-se 100% de acerto na base de testes com apenas 5 regras nebulosas.
A avaliação de Castro e de Silva sobre seus métodos de extração de regras é a seguinte:

Tabela 2-1 Comparação entre os métodos de extração de regras de Castro e de Silva

	Castro 2004 - TFRENN	Silva 2013
Abordagem	Transparente	Eclética
Portabilidade de Domínio	Independente	Independente
Formato das Regras	Nebulosa	Nebulosa
Precisão	Alta	Alta
Fidelidade	Alta	Alta
Facilidade de Compreensão	Depende do número de entradas	Alta
Complexidade do algoritmo	Baixa	Baixa

2.2.5 Relação entre as Redes Neurais de Funções de Base Radial e os Sistemas de Inferência Nebulosa

Em trabalho de 1993, Jang e Sun (JANG e SUN, 1993) demonstraram a equivalência funcional entre as redes de Funções de Base Radial (RBF - Radial Basis Functions Networks) e os sistemas de inferência nebulosa (FIS – Fuzzy Inference Systems).

A arquitetura de uma rede de Funções de Base Radial com n elementos na camada intermediária está esquematizada na Figura 2.12.

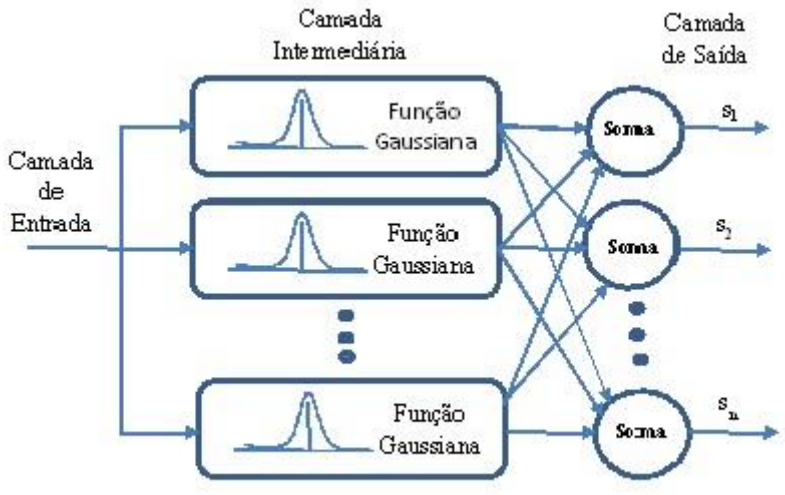


Figura 2-12 Rede de Funções de Base Radial (RBF – Radial Basis Functions)

Cada elemento da camada intermediária responde a uma área da entrada denominado campo receptivo.

A saída do i-esimo campo receptivo é:

$$w_i = g_i(\bar{x}) \tag{2.8}$$

onde $\bar{x}$ é o vetor n-dimensional de entrada , $\bar{c}_i$ é um vetor n-dimensional com os centros dos campos receptivos e $g_i(\bar{x})$ é tipicamente a função gaussiana

$$g_i(\bar{x}) = exp(-\frac{(\bar{x}-\bar{c})^2}{2\sigma_i^2}) \tag{2.9}$$

A saída de uma rede RBF pode ser calculada de duas maneiras: através da soma ponderada dos valores das funções associadas a cada campo receptivo, ou através da média ponderada destes mesmos valores.

$$f(\bar{x}) = \sum_{i=1}^{n} g_i w_i \qquad \text{modo soma ponderada;} \tag{2.10}$$

$$f(\bar{x}) = \frac{\sum_{i=1}^{n} g_i w_i}{\sum_{i=1}^{n} w_i} \qquad \text{modo media ponderada.} \tag{2.11}$$

Seguindo a argumentação de Jang e Sun no referido trabalho, um sistema de inferência nebulosa (FIS) é constituído de um conjunto de regras do tipo “Se – então”, de uma base de dados contendo as funções de pertinência de rótulos linguísticos, e de um mecanismo de inferência denominado “raciocínio nebuloso”.

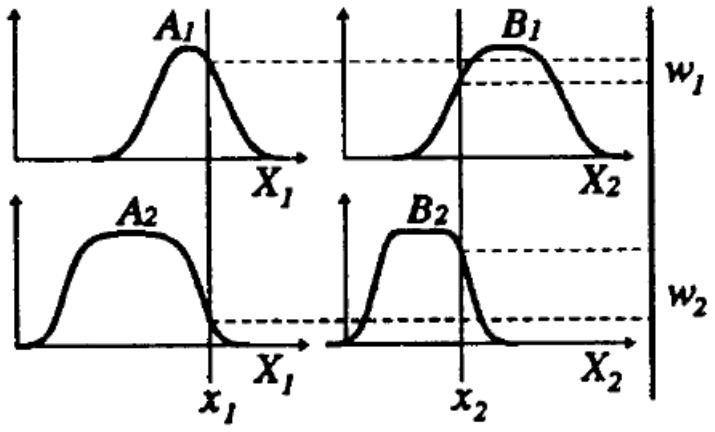
Como exemplo de regras, Jang e Sun apresentaram duas regras do tipo Takagi-Sugeno:

Regra 1: Se x_1 é A_1 e x_2 é B_1 , então $f_1 = a_1x_1 + b_1x_2 + c_1$ (2.12)

Regra 2: Se x_1 é A_2 e x_2 é B_2 ,

então $f_2 = a_2.x_1 + b_2.x_2 + c_2$ $f_2 = a_2x_1 + b_2x_2 + c_2$ (2.13)

Para estas regras, o mecanismo do raciocínio nebuloso pode ser esquematizado pela figura 2.13.



Fonte: (JANG e SUN, 1993)

Figura 2-13 Mecanismo de raciocínio nebuloso para as Regras 1 e 2.

O mecanismo de raciocínio nebuloso utiliza a funções de pertinência A_1, B_1 e A_2 , B_2 para calcular os pesos w_1 e w_2, respectivamente, através do operador T-norma , que costuma ser o operador min ou o operador multiplicação dos valores de pertinência das premissas.

$$w_i = \mu_{A_i}(x_1).\mu_{B_i}(x_2) \qquad \text{ou} \qquad w_i = \min\{\mu_{A_i}(x_1), \mu_{B_i}(x_2)\} \tag{2.14}$$

O cálculo da função de saída f para as duas regras fica:

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = w_1^n f_1 + w_2^n f_2 \tag{2.15}$$

Para n regras, considerando um vetor $\bar{x}$, a saída $f(\bar{x})$ pode ser escrita como:

$$f(\bar{x}) = \sum_{i=1}^{n} w_i f_1 \qquad \text{para saída como soma ponderada,} \tag{2.16}$$

ou

$$f(\bar{x}) = \frac{\sum_{i=1}^{n} w_i f_1}{\sum_{i=1}^{n} w_i} \qquad \text{para saída como media ponderada.} \tag{2.17}$$

Pela semelhança entre as equações 2.10 e 2.12 e as equações 2.11 e 2.17, Jang e Sun concluíram pela equivalência funcional entre os dois sistemas desde que observadas as seguintes restrições:

- O número de unidades de campo receptivo deve ser igual ao número de regras nebulosas “Se-então”;
- A saída de cada regra nebulosa “Se-então” é composta de uma constante;
- As funções de pertinência de cada regra devem ser funções gaussianas com as mesmas variâncias;
- O operador norma-T é a multiplicação;
- Os dois sistemas devem utilizar o mesmo método para o cálculo das saídas (ou somas ponderadas ou medias ponderadas).

Essa equivalência funcional demonstrada por Jang e Sun gerou muitos trabalhos na área de extração de regras e também na inserção de regras.

McGarry et al (MCGARRY, MACINTYRE e ADDISON, 2001) argumentaram que em muitas situações de falhas, não existem dados disponíveis para o treinamento de uma rede neural ou, se existem, elas são escassas. Entretanto, um especialista da área costuma ter uma boa ideia do que deve acontecer com os valores dos parâmetros de entrada e de saída da rede neural.

Caso o especialista consiga expressar seu conhecimento na forma de regras, este seria por demais valioso para ser ignorado. A figura a seguir, adaptada de

(MCGARRY, WERMTER e MACINTYRE, 2001), resume as transformações necessárias para enriquecimento da base de conhecimentos via inserção de regras.

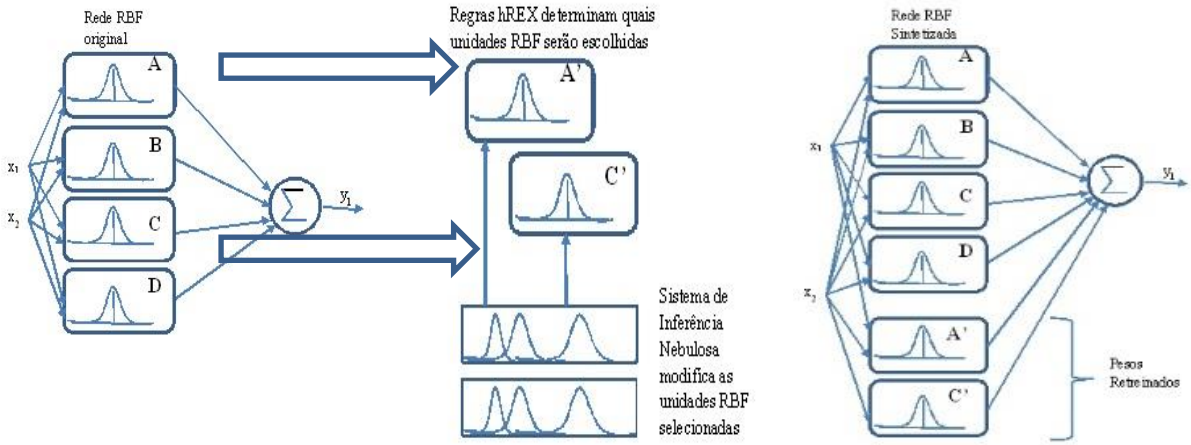


Figura 2-14 Processo de síntese do conhecimento

As cinco etapas para a conversão de regras nebulosas em uma rede RBF são:

- Aquisição do conhecimento, a partir dos especialistas da área e de outras referências para suprir a carência de dados;
- Definição dos conjuntos nebulosos, onde os parâmetros de entrada e de saída estão codificados como conjuntos nebulosos;
- Definição das regras nebulosas; estas regras determinam como os conjuntos nebulosos são usados no cálculo das saídas em função das entradas e como elas interagem para produzir novos parâmetros RBF;
- Determinação das unidades RBF escondidas que serão úteis, usando o algoritmo hREX (MCGARRY, WERMTER e MACINTYRE, 2001) para extrair regras simbólicas para selecionar os centros das RBFs que serão apropriados para cada classe.
- Utilizar o Sistema de Inferência Nebulosa onde os novos parâmetros RBF são calculados a partir dos centros identificados pelas regras do algoritmo hREX e da aplicação das regras nebulosas sobre os conjuntos nebulosos. As saídas do Sistema de Inferência Nebulosa são valores precisos para as novas localizações dos centros das unidades RBF.

Neste mesmo artigo de 2001, McGarry et al ilustram o princípio de inserção de conhecimento através de um sistema de reconhecimento de voz. A partir de dados coletados por Peterson e Barney sobre vozes humanas, foi desenvolvido um sistema de reconhecimento do sexo dos falantes. Os dados de Peterson e Barney consistem em 1520 exemplos com 4 características de entrada e 10 classes de saída.

O conhecimento do domínio disponível para estes dados afirma que uma voz feminina típica é 75% a 85% mais alta do que a voz masculina para o formante X1.
O conjunto de dados foi dividido em vozes masculinas e vozes femininas. O método utilizado foi o seguinte:
1. Treinar uma rede RBF usando somente vozes masculinas, com conjuntos separados para treinamento e para testes.
2. Testar a rede RBF masculina com vozes masculinas.
3. Testar a rede RBF masculina com vozes feminas.
4. Duplicar todas as unidades escondidas com as novas unidades com aumento entre 70-85% na localização dos centros de X1, sendo o valor exata descoberto empiricamente.
5. Modificar todos os pesos e polarizações entre as unidades escondidas e as de saídas para incluir as novas unidades.
6. Testar a rede RBF modificada com os dados das vozes femininas para testes.

Os resultados obtidos por este experimento foi o seguinte:
A rede treinada para vozes masculina obteve uma precisão de 87% para vozes masculinas de testes. Os dados com as vozes femininas para testes foram apresentados para esta rede, observando-se uma precisão de 42,2%. A heurística utilizada para modificar os novos centros foi a multiplicação dos valores existentes por 75%. Estes novos centros foram acrescentados à rede original, obtendo-se uma nova rede.

Esta nova rede obteve uma precisão para vozes femininas entre 10% e 16% maiores que a rede anterior.
Um segundo experimento de síntese de conhecimento foi realizado com dados sobre vibração, com o seguinte método:
1. Os dados de vibração foram divididos em dois grupos, os de motores de baixa velocidade (grupo a) e os de motores de alta velocidade (grupo b).
2. Uma rede RBF foi treinada com os dados do grupo de baixa velocidade e a precisão foi examinada com os dados de testes do mesmo grupo.
3. A rede RBF foi testada com os dados do grupo de alta velocidade e a precisão foi anotada.

4. As regras de domínio foram convertidas em conjuntos nebulosos e em regras nebulosas.
5. O algoritmo hREX foi utilizado para atribuir unidades para as classes de saídas.
6. Para cada uma das três classes de falhas consideradas, as unidades escondidas a elas atribuídas foram duplicadas usando-se o sistema de inferência nebulosa.
7. A rede RBF modificada teve seus pesos e valores de polarização das ligações entre as unidades escondidas e as de saídas recalculadas.
8. A rede modificada foi novamente testada com os dados de vibração dos motores de alta velocidade e a precisão foi anotada.

Os resultados observados foram os seguintes:
A rede RBF treinada com os dados para baixa velocidade obteve uma precisão de 94% para os dados de testes de motores de baixa velocidade.
A mesma rede produziu apenas 54% de precisão para os dados de testes de motores de alta velocidade.

Após modificar a rede RBF com novas unidades obtidas a partir de 18 regras nebulosas e execução do algoritmo hREX para associar as unidades escondidas com as classes de saídas, a nova rede foi testada com os dados de testes dos motores de alta velocidade. A taxa de precisão obtida subiu para 75%, o que demonstra os benefícios de se incluir o conhecimento de especialistas na falta de dados para calibração das redes RBF.

3 REDES DE PETRI

As redes de Petri foram criadas por Carl Petri na década de 60 com o objetivo de desenvolver uma linguagem para a comunicação entre máquinas sequenciais de forma gráfica e de fácil compreensão.

Posteriormente, diversos pesquisadores foram incluindo novas características à rede original, estendendo sua capacidade de descrever sistemas cada vez mais complexos, além de, em paralelo a estes novos tipos de redes, desenvolverem softwares capazes de implementá-los.

Neste capítulo é apresentada uma sequência da evolução das redes de Petri, ou seja, entre as dezenas de novos tipos de redes de Petri criadas ao longo dos anos, com foco naquelas que serviram de base para a rede de Petri proposta no próximo capítulo.

3.1 Redes de Petri Básicas

Neste item,será utilizada a notação de Cardoso e Valette (CARDOSO e VALETTE, 1997) em relação às definições formais.

3.1.1 Conceitos e Definições.

Uma rede de Petri é um grafo constituído por dois tipos de nós, os “Lugares” e as “Transições” interligados por “Arcos”.

Formalmente, uma rede de Petri é uma quádrupla

$$RP = (P, T, Pre, Post) \tag{3.1}$$

onde

- P é o conjunto finito de n lugares, n≥1;
- T é o conjunto finito de m transições, m≥1;
- Pre é uma aplicação de entrada, $Pre = P \times T \rightarrow \mathbb{N}$, onde $\mathbb{N}$ é o conjunto dos números naturais e os lugares precedem as transições;
- Post é uma aplicação de saída, $Post = T \times P \rightarrow \mathbb{N}$, onde $\mathbb{N}$ é o conjunto dos números naturais e os lugares são posteriores às transições;

Exemplo: Na Figura 3.1 está esquematizada uma rede de Petri básica que é descrita matematicamente por $RP = (P, T, Pre, Post)$ com $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2, t_3\}$, $Pre = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$ e $Post = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$, onde as linhas correspondem aos lugares p_1, p_2, p_3 e as colunas correspondem às transições t_1, t_2, t_3.

Nas matrizes Pre e Post, quando os elementos das matrizes são 0, significa que não existe uma conexão ou arco entre o lugar i e a transição j; onde os elementos são maiores que 0, significa que existe uma conexão entre o lugar e a transição, e a quantidade de fichas que fluem através do arco é exatamente o valor anotado.

No grafo associado à rede de Petri, os lugares são representados por círculos ou elipses e as transições são representadas por retângulos ou barras. Os arcos sempre conectam um lugar e uma transição ou uma transição e um lugar, mas nunca conectam um lugar com outro lugar ou uma transição com outra transição.

As fichas ou marcas circulam através do grafo saindo de um lugar para outro lugar indicando a mudança de estado do sistema ao longo do tempo. Os números associados aos arcos indicam a quantidade de fichas que devem fluir através dos mesmos. Quando nenhum número está indicado, adota-se a quantidade de 1 ficha.

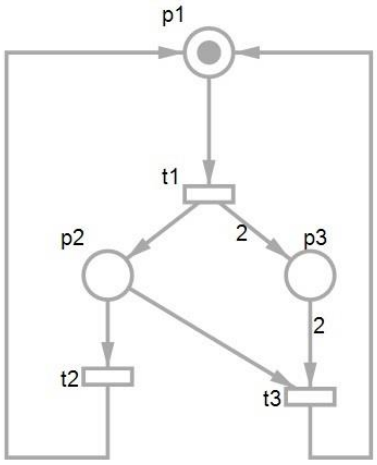


Figura 3-1 Rede de Petri Básica

3.1.2 Rede Marcada

Uma rede de Petri marcada é uma rede de Petri na qual cada lugar possui associado um valor correspondente à quantidade de fichas deste local em um dado instante do tempo.

Formalmente, uma rede marcada RM é uma dupla

$$RM = (RP, M) \tag{3.2}$$

onde

- RP é uma rede de Petri;
- M é a marcação inicial constituída por um vetor com n elementos, e n é a quantidade de lugares.

M(p) é a quantidade de fichas ou marcas contidas no lugar p.

No exemplo da Figura 3-1, o vetor M é dado por $M = \begin{bmatrix}1\\0\\0\end{bmatrix}$, indicando que, no instante inicial, o lugar p_1 possui 1 ficha e os lugares p_2 e p_3 não possuem fichas.

A evolução da rede ao longo do tempo é observada pela sequência de modificações do vetor M e esta sequência é calculada através da matriz de incidência Inc, que é obtida a partir das matrizes Pre e Post:

$$Inc = Post - Pre \tag{3.3}$$

No exemplo da Figura 3-1, a matriz Inc resultaria em:

$$Inc = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 1\\0 & 0 & 2\end{bmatrix} - \begin{bmatrix}0 & 1 & 1\\1 & 0 & 0\\2 & 0 & 0\end{bmatrix} = \begin{bmatrix}1 & -1 & -1\\-1 & 1 & 1\\-2 & 0 & 2\end{bmatrix} \tag{3.4}$$

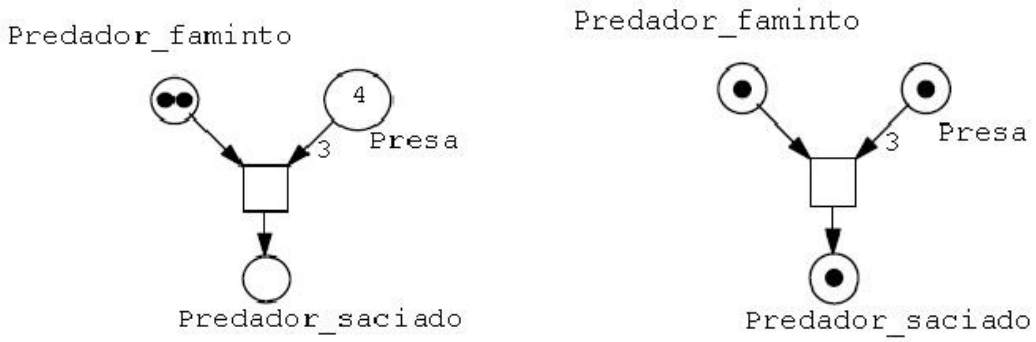
Para se referenciar a uma coluna associada a uma transição t, utiliza-se a notação $Pre(\cdot, t)$ para a matriz Pre, $Post(\cdot, t)$ para a matriz Post e $Inc(\cdot, t)$ para a matriz de Incidência.

3.1.3 Transição Habilitada

Uma transição estará habilitada a disparar quando a quantidade de fichas ou marcas nos lugares que antecedem a transição for maior ou igual ao peso do arco que conecta cada lugar a esta transição.

Matematicamente, esta condição é especificada como:

$$M(p) \geq Pre(p, t), \forall p \in P, \text{ ou ainda } M \geq Pre(\cdot, t) \tag{3.5}$$



(a) (b)

Figura 3-2 Ilustração de uma regra de transição

Na figura 3-2-a, no lugar “Predador_faminto” temos 2 predadores e no lugar “Presa” temos 4 presas. A transição t está habilitada, pois as duas fichas no lugar “Predador_faminto” superam o consumo de fichas associado ao arco entre este lugar e a transição (2>1), assim como a quantidade de fichas em “Presa” supera a quantidade de fichas indicada no arco correspondente (4>3). Depois do disparo de t, as fichas mudam como mostra a Figura 3-2-b e aí a transição t já não estará habilitada. Isso quer dizer que depois da transição, apesar de existir ainda uma ficha em “Predador_faminto”, a quantidade de fichas em “Presa” não permite um novo disparo de t (1<3).

Note-se que as transições assumem que cada lugar pode armazenar um número ilimitado de fichas, ou seja, que a rede de Petri tem capacidade infinita. No entanto, em muitos casos, é preferível que a modelagem tenha um limite superior para o número de fichas que podem ser armazenadas a cada local.

Estas redes são consideradas como redes de capacidade finita e são denotadas como (PN, M_0), onde M_0 representa a marcação inicial e a cada lugar p da rede possui uma capacidade máxima de fichas $k(p)$. Para que uma transição t de uma rede de capacidade finita esteja habilitada é empregada a regra de transição estrita que exige que cada lugar de saída p não exceda sua capacidade $k(p)$ depois do disparo de t.

3.2 Modelagem com redes de Petri

3.2.1 Exemplo de modelagem de Leitores e Gravador

Na Figura 3-3 estão esquematizados dois processos de leitura e processo de gravação que competem pelo mesmo recurso, no caso, uma base de dados. Os dois processos de leitura podem operar em paralelo, mas o processo de gravação exige que nenhum outro processo esteja acessando a base de dados.

Os lugares “Linat1” e “Linat2” indicam que os processos 1 e 2 de leitura estão inativos. Os lugares “Lesp1” e “Lesp2” indicam que os processos estão à espera da

liberação do recurso, indicado pelo lugar “Rec”. Os lugares “Lativ1” e “Lativ2” indicam que os processos estão ativos, ou seja, estão lendo a base de dados.

O pedido de leitura acontece através da ativação das transições t1 e t2 e, quando isso ocorre, a ficha de marcação sai do lugar inativo para o lugar de espera.

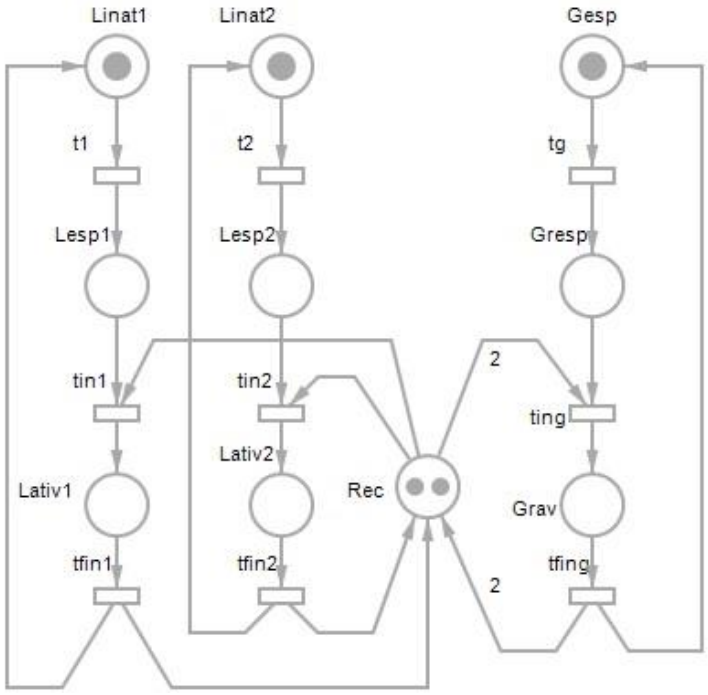


Figura 3-3 Processos de Leitura e Gravação

Se existir recurso disponível (ficha no lugar Rec), então tin1 ou tin2 iniciam a leitura, consumindo uma ficha cada uma das transições.

No caso do gravador, ocorrendo um pedido de gravação e estando o gravador inativo, a transição tg dispara, consome uma ficha e a transfere para o lugar Gresp, ou seja, estado de espera de gravação.

Para a transição ting (início da gravação) ser habilitada é necessário que o lugar Gresp contenha uma ficha e que o lugar Rec contenha duas fichas. Esta exigência de duas fichas ocorre para impedir que alguma leitura ocorra em paralelo com a gravação. Deixando o lugar Rec sem fichas é a garantia de que nenhuma leitura ocorra durante a gravação da base de dados.

As transições tfin1 e tfin2 finalizam as leituras e devolvem uma ficha cada para o lugar Rec. A transição tfing finaliza a gravação e devolve duas fichas ao lugar Rec.

3.3 Rede de Petri Auto-Modificável (RPAM)

Esta rede de Petri Auto-Modificável foi definida por Valk (VALK, 1978) como sendo uma extensão natural das redes de Petri básicas.

A definição de Valk para a rede de Petri Auto-Modificável é a seguinte:

$$RPAM = (P, T, Pre, Post, M_0) \tag{3.6}$$

onde

- P é o conjunto finito de n lugares, n≥1;
- T é o conjunto finito de m transições, m≥1;
- M_0 é a marcação inicial da $RPAM$, onde $M_0 = \begin{bmatrix} M_0(p_1) \\ M_0(p_2) \\ \vdots \\ M_0(p_n) \end{bmatrix}$ com $M_0(p_i) \in \mathbb{N}$

As aplicações Pre e Post são diferentes das definidas para as redes de Petri básica e suas novas definições são:

- Pre é uma aplicação de entrada, $Pre = P \times P_1 \times T \rightarrow \mathbb{N}$, onde $\mathbb{N}$ é o conjunto dos números naturais e $P_1 = P \cup \{1\}\, e\ 1 \notin P$;
- Post é uma aplicação de saída, $Post = T \times P_1 \times P \rightarrow \mathbb{N}$, onde é o conjunto dos números naturais e os lugares são posteriores às transições;

No caso de Pre, o primeiro parâmetro é um lugar de entrada da transição indicada pelo terceiro parâmetro. O segundo parâmetro representa um lugar ou então o número 1. Se o parâmetro for um lugar, então o peso do arco será dado pela marcação do lugar especificado, mas se o parâmetro for 1, então o peso do arco é o valor da função Pre(p,1,t).

No caso de Post, o terceiro parâmetro é um lugar de saída da transição indicada pelo primeiro parâmetro. O segundo parâmetro, tal como no caso Pre, representa um lugar ou o número 1. Se o parâmetro for 1, o peso do arco é o valor da função Post(t,1,p), caso contrário, o parâmetro é um lugar cuja marcação é o peso do arco que une a transição ao lugar.

Resumindo, para a função Pre

$Pre(p, 1, t) = peso\ do\ arco\ de\ p\ para\ t$ e

$Pre(1, q, t) = 1\ e\ o\ peso\ do\ arco\ será\ a\ quantidade\ de\ fichas\ contidas\ no\ lugar\ q$;

e para a função Post

$Post(t, 1, p) = peso\ do\ arco\ de\ t\ para\ p$ e

$Post(t,q,1) = 1$ *e o peso do arco será a qtde. de fichas contidas no lugar* q;
Se não existir um arco entre um lugar p e a transição t, então $Pre(p,q,t) = 0$ e se não existir um arco entre uma transição t e um lugar p, então $Post(t,q,p) = 0$.

Caso a função Pre seja definida por $Pre: P \times \{1\} \times T \rightarrow \mathbb{N}$, que é quando os arcos de entrada das transições não possuem dependência com a marcação de outros lugares da rede, então esta rede é denominada rede de Petri Auto-Modificável Posterior (RPAMP).

Uma RPAM se reduz a uma RP básica quando as funções Pre e Post forem definidas como
$Pre: P \times \{1\} \times T \rightarrow \mathbb{N}$;
$Post: T \times \{1\} \times P \rightarrow \mathbb{N}$.
Neste caso, nenhum arco possuirá dependência da marcação de algum outro lugar da rede.

Figura 3-4 Representação de Pre(p,q,t) e Post(t,q,p)

A marcação M(q) da RPAM é uma função de P para $\mathbb{N}$, ou seja, todos as fichas contidas em um dado momento em todos os lugares da rede constituem uma marcação. Para uma marcação M, define-se uma nova função auxiliar AM(q) tal que

$$AM(q) = \begin{cases} M(q), quando\ q \neq 1\ e\ q \in P \\ 1,\ quando\ q = 1 \end{cases} \quad (3.7)$$

Uma transição $t \in T$ estará habilitada para uma dada marcação M se

$$M(p) \geq \sum_{q \in P_1} Pre(p,q,t).AM(q), para\ \forall p \in P \quad (3.8)$$

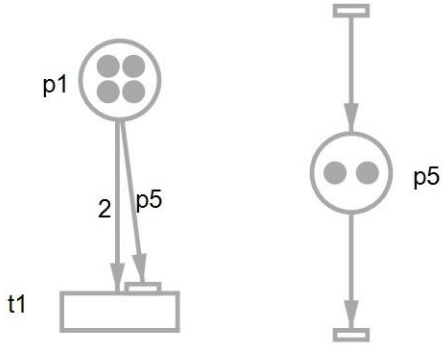


Figura 3-5 Transição Habilitada

A Figura 3-5 apresenta um exemplo de transição habilitada. A transição t1 depende da quantidade de fichas armazenadas em p1 e da quantidade de fichas armazenadas em p5.

Neste exemplo,

$M(p1) = 4$,

$Pre(p1, p5, t1) = 1$, $AM(p5) = M(p5) = 2$,

$Pre(p1,1, t1) = 2$ e $AM(1) = 1$, portanto

$M(p1) = 4 \geq Pre(p1,1, t1).AM(1) + Pre(p1, p5, t1).AM(p5) = 2.1 + 1.2 = 4$, ou seja, o lugar p1 possui fichas suficientes para serem consumidas pela transição t1.

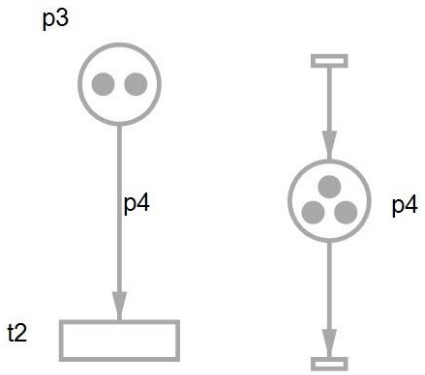


Figura 3-6 Transição Desabilitada

No caso do exemplo ilustrado pela Figura 3-6, a transição t2 resulta desabilitada em função da quantidade de fichas em p3 ser menor que a quantidade de fichas consumidas pela transição t2, especificada pela marcação do lugar p4.

$M(p3) = 2$,

$Pre(p3, p4, t2) = 1$, $AM(p4) = M(p4) = 3$,

$M(p3) = 2 < Pre(p3, p4, t2).AM(p5) = 1.3 = 3$, ou seja, a transição t2 consome 3 fichas que não estão disponíveis em p3.

Um exemplo de aplicação apresentado por Valk em seu artigo é a utilização da RPAM na modelagem de processos de leitores e gravadores operando sobre um mesmo recurso ou base de dados (Figura 3-7).

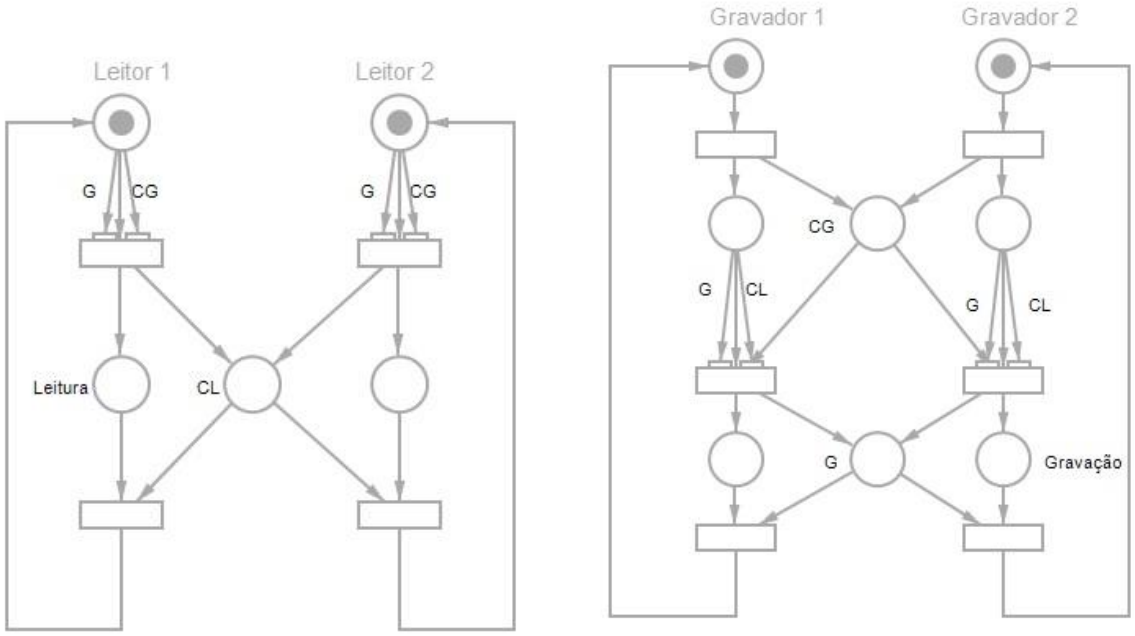


Figura 3-7 Aplicação da RPAM

Nesta RPAM, o lugar CL representa um contador de leitura, o lugar CG representa um contador de gravação e G indica que um processo de gravação está ocorrendo. O lugar G afeta diretamente as regras de habilitação tanto das transições de leitura como das transições de gravação.

3.3.1 Aplicações da RPAM em Manufatura Flexível

Sasaki (SASAKI, 1996) utilizou a RPAM para modelar o sistema de Manufatura Flexível esquematizado na Figura 3-8. O Robô recolhe peça na “Esteira de Entrada de Peças” e a coloca em uma das duas Máquinas disponíveis. Quando a peça estiver pronta, o Robô transporta a peça de uma das máquinas para a “Esteira de Saída de Peças”.

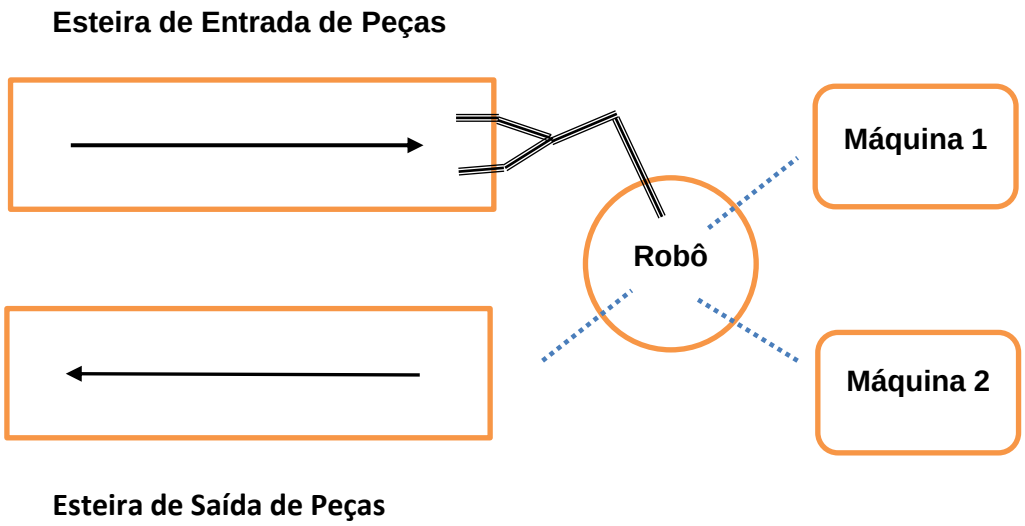


Figura 3-8 Esquema de Manufatura Flexível

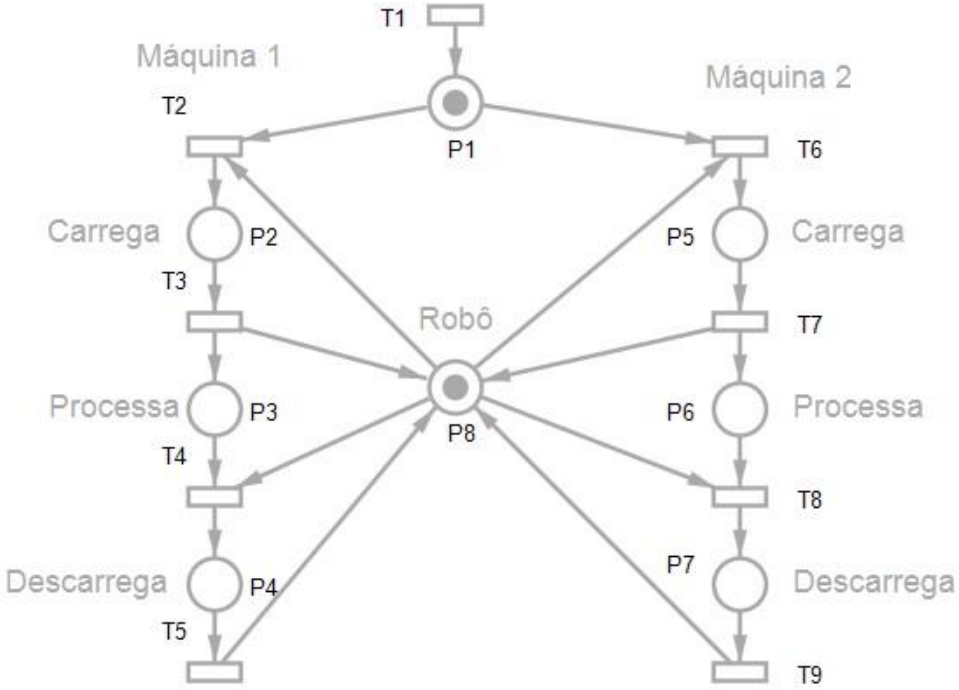


Figura 3-9 Modelo do Processo de Manufatura Flexível com auxílio de Robô

A rede de Petri correspondente está ilustrada na Figura 3-9 (adaptada de Sasaki). No lugar P1, uma peça está disponível. Se a transição T2 dispara, então o Robô carrega a peça na Máquina 1 e, em seguida, a transição T3 libera o Robô e a Máquina 1 inicia o processamento da peça.

Se outra peça aparecer na esteira de entrada de peças, o lugar P1 voltará a ser marcado com uma ficha e, desde que o lugar P5 esteja vago, a transição T6 dispara e o Robô transporta a peça para a Máquina 2, provocando o deslocamento da ficha de P1 para P5 na rede de Petri.

Após o carregamento da peça na Máquina 2, a transição T7 dispara liberando o Robô e a Máquina 2 inicia o processamento da peça. Quando uma máquina termina seu processamento, estando o Robô disponível (ficha no lugar Robô), a transição correspondente dispara (T4 para a Máquina 1 e T8 para a Máquina 2), provocando nova atuação do Robô para descarregamento da peça (lugar P4 para a Máquina1 e lugar P7 para a Máquina 2).

Esta rede de Petri não leva em consideração a possibilidade das máquinas não estarem disponíveis em virtude de quebras ou de manutenção preventiva. Sasaki desenvolveu uma nova rede de Petri incluindo a indisponibilidade das máquinas, utilizando a rede de Petri Auto-Modificável (Figura 3-10 adaptada de Sasaki) nos moldes definidos por Valk.

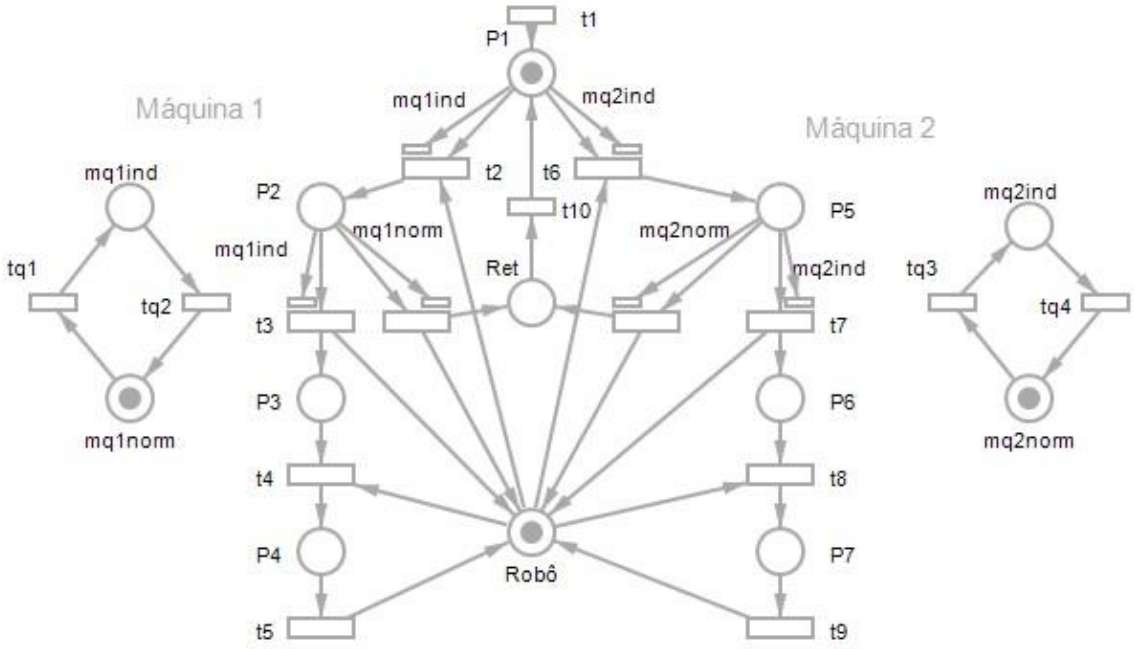


Figura 3-10 Modelo do Processo de Manufatura Flexível com RPAM

Nesta RPAM, os lugares mq1ind (máquina 1 indisponível) e mq1norm (máquina 1 normal) foram introduzidos para possibilitar a retirada da Máquina 1 da operação, mas sem alterar a rede de Petri que se adapta à situação pela alteração das regras de disparo t2 e t3.

Analogamente, os lugares mq2ind (máquina 2 indisponível) e mq2norm (máquina 2 normal) exercem a mesma função para a Máquina 2, provocando alteração das regras de disparo t6 e t7. Quando a Máquina 2 estiver indisponível, o lugar mq2ind conterá uma ficha e esta ficha aumenta de 1 para 2 a quantidade de fichas necessárias no lugar P1 para que a transição t6 seja habilitada. Como P1 só pode conter 1 ficha, na prática isto significa que a transição t6 jamais ocorrerá enquanto a Máquina 2 estiver indisponível.

Quando as máquinas estão em situação normal, a RPAM se comporta como a rede de Petri original.

3.4 Redes de Petri Coloridas

As Redes de Petri coloridas (RPC) estendem o vocabulário de redes de Petri básicas e adicionam características que as tornam adequadas para modelar grandes sistemas.

O conceito de cor foi introduzido basicamente para distinguir as fichas que circulam através das redes de Petri. No conceito original, só existe um tipo de ficha. Nas redes de Petri coloridas, as fichas possuem cor, ou seja, uma transição pode exigir não apenas uma certa quantidade de fichas para disparar, mas também que ela seja de uma determinada cor.

As RPC combinam os pontos fortes das redes de Petri comuns com os pontos fortes de uma linguagem de programação de alto nível chamada CPN ML, baseada na linguagem funcional SML. O Apêndice A desta tese contém uma descrição mais ampla e detalhada da linguagem CPN ML, que está associada ao simulador CPN Tools. Detalhes adicionais com ênfase em aplicações em processos e negócios podem ser encontrados em (AALST e STAHL, 2011) e em (JENSEN e KRISTENSEN, 2009).

Formalmente, uma RPC é uma 9-tupla definida como

$$RPC = (C, P, T, A, N, F, G, E, I) \tag{3.9}$$

onde

C – é um conjunto não-vazio de tipos denominados conjunto de cores;

P – é um conjunto finito de lugares;

T – é um conjunto finito de transições;

A – é um conjunto finito de arcos tal que $P \cap T = P \cap A = T \cap A = \emptyset$;

N – é a função Nó, que é definida a partir de A em $(P \times T) \cup (T \times P)$;

F – é a função colorida e é definida de P para C;

G – é a função de guarda definida para T na qual os resultados são booleanos e os tipos das variáveis envolvidas na função devem ser de um tipo contido em C;

E – é a função de expressão do arco definida para A e o tipo dos resultados deve ser o mesmo que a cor do lugar a este arco e a cor deve pertencer a C;

I – é a função de inicialização definida para P de modo que seu tipo deve ser idêntico ao do lugar associado.

As redes de Petri fornecem os primitivos para a interação do processo, enquanto a linguagem de programação fornece os primitivos para a definição de tipos de dados e as manipulações de valores de dados.

Assim, em um modelo RPC, as fichas podem ser codificadas como valores de dados de um rico conjunto de tipos (chamados conjuntos de cores) e inscrições de arco podem ser expressões computadas e não apenas constantes.

A Figura 3.11 mostra um exemplo de reação química sendo representada por Rede de Petri Colorida.

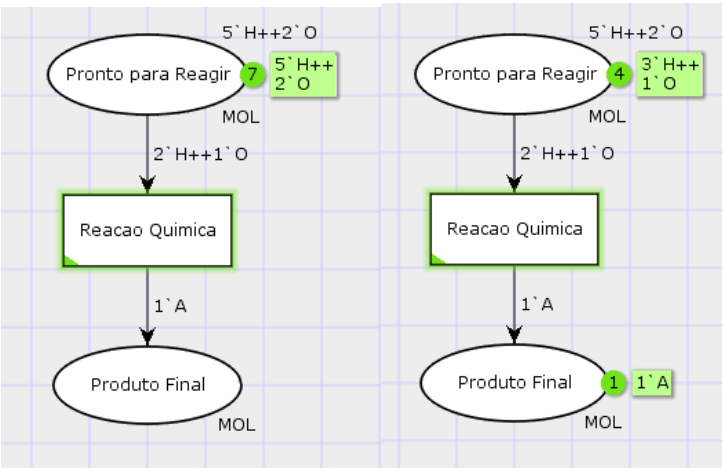


Figura 3-11 Reação Química para formação da molécula de água

Neste exemplo, o estado inicial “Pronto para Reagir” contém cinco fichas H e duas fichas O, que são mostradas no canto superior direito do lugar. O operador ++ é usado para representar uma operação de união multi-conjunto (multiset). O peso sobre o arco significa que duas fichas de hidrogênio e uma ficha de oxigênio são necessárias para a transição “Reação Química” ser ativada e disparada.

Uma vez que a transição é disparada, as fichas são retiradas do lugar de entrada e uma ficha A, ou água, é colocada no lugar “Produto Final”. Em uma rede Petri básica, as fichas são indistinguíveis, enquanto que na RPC, a todas as fichas são atribuídos um valor. Os valores das fichas têm um tipo e podem ser manipulados através do uso de uma linguagem de programação de alto nível.

Para cada estado de uma rede, o tipo de conteúdo deve ser especificado. Na Rede de Petri Colorida, cada local tem o tipo de ficha (MOL) rotulado na parte inferior direita. MOL é um tipo de dados que pode ser definido usando a linguagem de alto nível CPN ML. Em CPN ML todos os tipos de dados são referidos como conjuntos de cores.

Pode-se definir um conjunto de cores do seguinte modo:

```
colset MOL = with H | O | A;
```

A declaração acima indica que o conjunto de cores MOL (um tipo de enumeração) pode ser composto de valores H, O e A.
Observe que uma vez que as fichas podem ter identidades separadas, não precisamos mais manter o lugar do oxigênio separado do lugar para o hidrogênio. Mesmo neste exemplo simples, pode-se observar que os modelos de RPC tendem a ser muito mais compactos quando comparados à sua contraparte descrita em rede de Petri básica.

Em simulação, as fichas podem ser usadas para especificar entidades e atributos. As entidades efetuam as mudanças no estado do sistema, enquanto os atributos são características de uma dada entidade.

As entidades podem ser declaradas como um produto de vários atributos. Um exemplo de definição de uma cor de tipo composto conjunto é mostrado abaixo:

```
colset Empregado = ID * Salario;
```

A declaração acima descreve um tipo de dados de empregado, consistindo de uma identificação e do valor do salário. ID e Salario também são conjuntos de cores. ID,

neste caso, é composto de uma sequência de caracteres (uma string) e Salario é composto por um número real. As suas declarações são as seguintes:

colset ID = STRING;

colset Salario = REAL;

3.4.1 Entradas, Saídas e Estado

A entrada para uma simulação RPC é a marcação inicial da rede. As RPCs fornecem uma visão localizada de um sistema. As entradas alteram os estados individuais do sistema. As transições mudam os estados individuais, bem como o estado geral do sistema. A combinação dos estados, ou a marcação final da rede, pode ser considerada a saída da simulação.

3.4.2 Variáveis

CPN permite variáveis em diversas expressões. Todas as variáveis devem ser declaradas com seu tipo (conjunto de cores).Por exemplo,

Var contador: INT;

Declara uma variável denominada contador do tipo INT. As variáveis podem ser de tipos simples ou compostos. As variáveis são ligadas a valores de sua cor declarada definida pelo simulador, uma vez que ele tenta determinar se uma transição está ativada.

O escopo de uma variável é local para a transição e pode haver várias ligações simultaneamente ativas em transições diferentes. Estas ligações podem existir ao mesmo tempo porque têm diferentes escopos. A extensão de uma ligação de variável CPN é o disparo de uma transição particular.

Além disso, o CPN permite variáveis globais (chamadas variáveis “ref”) e constantes. As primeiras são declaradas usando a palavra-chave “globref”, enquanto que as constantes são declaradas usando a palavra-chave “val”.

3.4.3 Gerador de números aleatórios.

CPN Tools utiliza o gerador de números aleatórios do CPN ML. Isto pode ser particularmente útil em simulações para representar atrasos de um período de

tempo desconhecido. O CPN Tools fornece várias funções de distribuição aleatória incluindo discreto, exponencial e uniforme.

3.4.4 Relógio e Calendário

No simulador de CPN o relógio é representado como um contador inteiro. O tipo interno deste contador permite arbitrariamente grandes valores inteiros e não está restrito a uma representação de 32 bits. Dependendo do sistema que está sendo modelado, o valor do relógio pode ser interpretado como milissegundos, ou segundos, ou minutos, etc.

O calendário é representado pela distribuição de fichas com seus carimbos de tempo e transições associadas em um modelo cronometrado. O carimbo de data / hora representa o tempo mais cedo possível que uma ficha específica pode ser consumida por uma transição. Isso pode depender da execução de outros eventos também.

Uma vez que, é possível para os modelos ser do tipo atemporal, pode-se pensar de calendário como uma lista de eventos que podem ocorrer em alguma ordem que não é necessariamente determinada pelo tempo.

3.5 Redes de Petri Coloridas Auto-Modificáveis (RPCAM)

Este tipo de Rede de Petri foi definida por Guan e Liu (GUAN e LIU, 2003) para resolver problemas na área de sincronização de multimídia. Além de permitir a modificação da rede, existem dois tipos de camadas denominadas de controle e de apresentação. A rede da camada de apresentação interage com os dispositivos multimídia, enquanto que a rede da camada de controle possibilita a alteração da estrutura da rede da camada de apresentação. Deve-se observar que esta é uma rede de Petri adaptativa uma vez que seu comportamento e sua estrutura são modificáveis durante a execução da mesma.

Na RPCAM as fichas são divididas em dois tipos: fichas de cores e fichas de recursos. Além disso, existem dois tipos de fichas de cores: um associado a

comandos para realizar o controle, outro associado a um número que indica o número de iterações necessárias para executar.

Há também dois tipos de fichas de recursos usados para reproduzir recursos: um para frente que se move na mesma direção com arcos e um para trás que se move na direção oposta com arcos. Na Figura 3.12, um exemplo de RPCAM é ilustrado. Os mecanismos desenhados em linhas pontilhadas são criados depois que os comandos de controle associados à ficha colorida c são executados.

Quando a ficha colorida c aparece no lugar p_1, os comandos associados a ele serão executados. Na RPCAM existem alguns comandos primitivos que podem modificar a estrutura da rede, como criar um local, desativar uma transição, etc. Podem-se combinar vários desses comandos básicos em uma seqüência especial para lidar com diferentes interrupções de usuário / eventos de rede.

Existem duas abordagens para representar comandos de modificação: uma é usar uma ficha colorida para representar um tipo de interrupção (de usuário), assim uma ficha colorida representa uma combinação de vários comandos básicos. Outra é usar uma ficha colorida para representar um comando básico e um tipo de ficha colorida também indica a prioridade de um comando (ou seja, qual comando será executado primeiro).

No exemplo ilustrado na Figura 3.12, a ficha colorida c está associada a cinco comandos básicos: desativar a transição t_1, criar um lugar de apresentação p_3, criar um arco da transição t_1 para o lugar p_3, criar um arco do lugar p_3 para a transição t_2, ativar a transição t_1. A parte desenhada em linhas pontilhadas não existem no início, então a ficha colorida c é injetada em p_1 devido a algumas interrupções / eventos. A implementação da ficha de recurso no lugar p_1 será pausada, então os comandos de modificação associados a c serão executados.

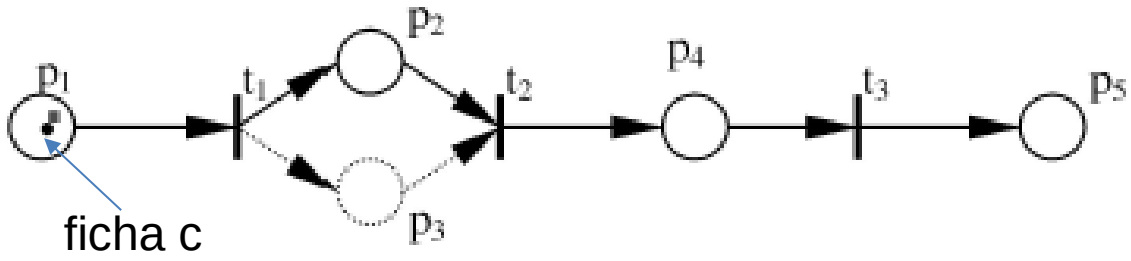


Figura 3-12 RPCAM - A Ficha Colorida c modifica a estrutura da rede pelo acréscimo do lugar p_3 e dos arcos t_1-p_3 e p_3-t_2 (GUAN e LIM, 2004)

O local e os arcos desenhados em linhas pontilhadas são criados sucessivamente e as definições da RPCAM são apresentadas a seguir.

3.5.1 Definição da Rede de Petri Colorida Auto-Modificável.

A RPCAM é uma 7-tupla

$$RPCAM = \{P, T, A, D, U, C, M\} \text{ onde } P \cap T \neq \emptyset. \quad (3.10)$$

$P = \{p_1, p_2, p_3, \cdots, p_m\}$ é um conjunto finito de lugares representando estados em um sistema de multimídia ou processos em uma rede de transmissão, onde $m > 0$.

$T = \{t_1, t_2, t_3, \cdots, t_n\}$ é um conjunto finito de transições representando pontos de sincronização em um sistema de multimídia / processos em uma rede de transmissão, onde $n > 0$.

$A: \{P \times T\} \cup \{T \times P\}$ é um conjunto finito de arcos representando a relação de fluxo.

$D: P \rightarrow R^+$ é um mapeamento do conjunto de lugares da apresentação para números reais não-negativos, representando a duração dos recursos em questão / velocidade de transmissão da rede.

$U = \{u_1, u_2, u_3, \cdots, u_l\}$ é o conjunto finito de cores; cada cor pode representar um tipo de interrupção do usuário / evento da rede, onde $l > 0$.

$C = \{c_1, c_2, c_3, \cdots, c_k\}$ é o conjunto finito de comandos (como definido na Tabela 3-1), onde $k > 0$ (por exemplo, c_4 – “Criar um lugar”, c_5 – “Eliminar um lugar”).

$M : P \rightarrow \{I_r^+, I_c^+\}$ é um mapeamento do conjunto de lugares para um conjunto de números inteiros e representa uma marca de uma rede, onde (I_r – conta o número

de fichas de recursos; I_c – conta o número de fichas coloridas); I_r^+= {0,1,2, . . .} e I_c^+ = {0,1,2, . . . }.

Na RPCAM, quando uma ficha colorida é injetada em um lugar, os comandos associados a ele serão disparados. Diferentes fichas coloridas podem ser associadas a diferentes interrupções do usuário / eventos de rede. O trabalho principal de um projetista é instalar os comandos básicos definidos e projetar a RPCAM com base em requisitos especiais de diferentes interrupções de usuário / eventos de rede.

Uma outra maneira de resolver este problema é permitir que um usuário altere a própria RPCAM usando esses conjuntos básicos de comandos.

3.5.2 Regras de Habilitação e Disparo da RPCAM

Uma transição em uma rede de Petri é habilitada se todos os seus lugares de entrada forem compostos por uma certa quantidade de fichas que é maior ou igual ao número de arcos de cada respectivo lugar para a transição e nenhuma interrupção o desabilita.

Se as condições mencionadas forem atendidas, a transição dispara e fichas serão removidas de cada um dos seus lugares de entrada e fichas serão criadas em cada um dos seus lugares de saída. A transição dispara instantaneamente se cada um de seus lugares de entrada não estiver associado a algum atraso ou contém alguma ficha desbloqueada.

No caso de um lugar estar associado um atraso, o local permanece no estado ativo por um intervalo especificado pelo atraso D após receber uma ficha. Durante este período, a ficha estará bloqueada.

Após o atraso D, a ficha ficará desbloqueada. Para a RPCAM, se nenhum usuário desabilitar a transição, ela obedecerá a essa regra de habilitação da rede Petri. Ao introduzir novos mecanismos, a ficha colorida e os comandos primitivos de controle, a rede RPCAM suporta aplicações multimídia interativas / distribuídas e pode

manipular o fluxo da rede Petri modelada. Além de apoiar as regras convencionais das redes de Petri, algumas novas regras da RPCAM são necessárias.

Regras de habilitação:

1. Criar fichas coloridas para controle: quando um usuário interromper ou ocorrer um evento da rede, as fichas coloridas correspondentes a esta interrupção serão criadas e injetadas nos lugares que contenham fichas de recursos. Os comandos associados a cada ficha colorida serão executados em ordem.
2. Quando uma ficha colorida é injetada em um lugar, a execução das fichas de recursos neste lugar será pausada. Ao mesmo tempo, o sistema não será capaz de atender outras interrupções.
3. Quando uma ficha colorida é criada, os comandos associados a ela serão executados. A estrutura da rede será modificada para satisfazer as demandas da interrupção correspondente.

Regra de disparo: quando todos os comandos associados a uma ficha colorida foram executados, essa ficha colorida será excluída. Ao mesmo tempo, as fichas de recursos normais serão retomadas. Em particular, a ficha colorida associada a um índice N será excluída quando o número do índice foi diminuído para zero. Então, a transição ativada irá disparar de acordo com as regras básicas de disparo da rede Petri.

Término: quando uma ficha colorida termina sua operação, todas as modificações feitas para a interrupção do usuário / evento da rede serão removidas por procedimento padrão, ou seja, a estrutura da rede será restaurada para a configuração anterior à interrupção do usuário / evento da rede ocorrer. Assim, as modificações de uma interrupção não afetarão a próxima apresentação. Se o usuário optar por preservar a modificação, a estrutura da rede alterada permanecerá.

Com a introdução destes novos mecanismos, a RPCAM perde a propriedade de segurança e conservação. Embora essas propriedades sejam sacrificadas, a RPCAM ganha um poder de maior expressividade no aspecto da programação geral e escalabilidade ao manipular eventos de usuários ou de rede.
A Tabela a seguir ilustra o conjunto de operações associadas às fichas coloridas.

Tabela 3-1 - Lista dos comandos associados às fichas coloridas da RPCAM

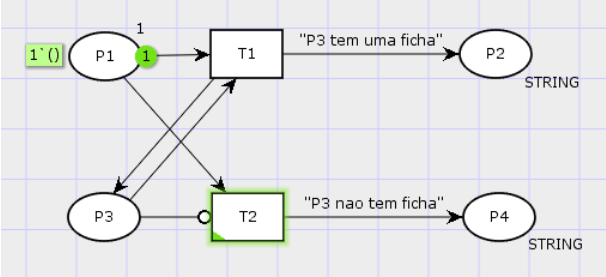
No	Mecanismo	Comando	Ação
1	Arco	Criar arco	Um arco é criado
2		Eliminar arco	Um arco é eliminado
3		Criar arco para descarte	Um arco para descartar algum recurso é criado
4	Lugar	Criar lugar	Um lugar é criado
5		Eliminar lugar	Um lugar é eliminado
6		Substituir lugar	Um lugar é substituído
7		Modificar duração	Modifica a duração associada a um lugar
8	Transição	Habilitar transição	Uma transição é capaz de disparar quando as condições para disparar forem preenchidas
9		Desabilitar transição	Uma transição não é capaz de disparar mesmo quando as condições para disparar forem preenchidas
10		Criar transição	Uma transição é criada
11		Eliminar transição	Uma transição é eliminada
12	Ficha colorida Ficha para frente Ficha para trás	Bloquear ficha	Para bloquear uma ficha
13		Desbloquear ficha	Para desbloquear uma ficha
14		Reverter	Para modificar a direção de uma ficha de recurso
15		Pausar ficha	Para parar a contagem decrescente se o lugar está associado com a duração ou impede a transição de disparar se o lugar não estiver associado a alguma duração
16		Continuar ficha	Para continuar uma ficha e recomeçar a contagem regressiva a partir do tempo em que ela foi parada
17		Criar ficha	Para criar uma ficha no locar indicado sem

		(colorida, normal)	nenhuma condição
18		Eliminar ficha (colorida, normal)	Para remover uma ficha
19		Desabilitar ficha	Permite que uma ficha rode sem nenhum efeito. Dado um sinal de áudio como no exemplo, deixa ele passar através da rede sem nenhum som.
20	Número	Incrementa valor	Incrementa o valor de um número associado a uma ficha colorida
21		Decrementa valor	Decrementa o valor de um número associado a uma ficha colorida

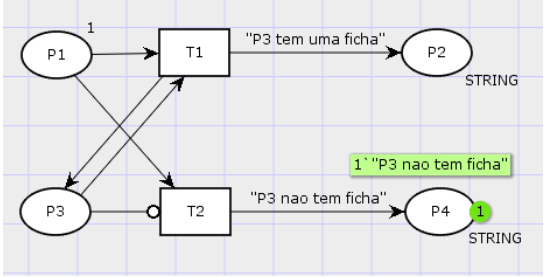
3.5.3 Expressividade da RPCAM

Agerwala (AGERWALA, 1974) e outros mostraram que um modelo de rede de Petri estendido com a capacidade de testar se um lugar não possui fichas pode simular uma máquina de Turing. Guan (GUAN e LIU, 2003) demonstrou usando o esquema da Figura 3.13 que a RPCAM possui esta a capacidade de testar se um lugar não possui fichas,ou seja, que a RPCAM tem a mesma expressividade que a Máquina de Turing.

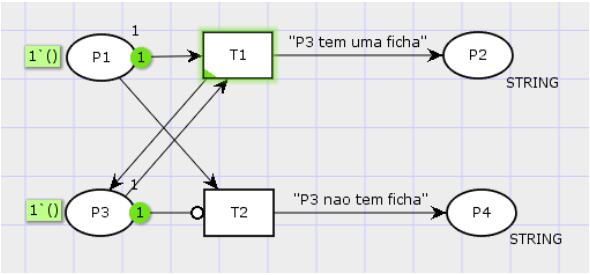
A Figura 3-13 foi produzida via rede de Petri Colorida, com um recurso de um arco especial para teste da situação de “Zero Fichas” no lugar de origem.Na Figura 3-13, considere P3 representando um lugar a ser testado, P1 representando um lugar para iniciar o teste de zero fichas, P4 representando “P3 não tem ficha” e P2 representando “P3 tem uma ficha”.



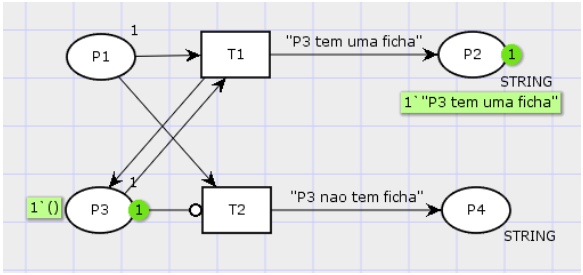
(a) P3 sem fichas, antes do disparo de T2



(b) depois de disparo de T2



(c) P3 com 1 ficha, antes do disparo de T1



(d) depois do disparo de T1

Figura 3-13 Teste de "Zero Fichas" no lugar P3 com arco inibidor

Quando P3 não possui fichas, a transição T2 fica habilitada e a transição T1 fica desabilitada. O arco simbolizado por  (arco inibidor) significa que a transição é possível quando não houver fichas no lugar de origem do arco, ou seja, é este arco que possibilita o teste de “Zero Fichas” na RPC. Quando P3 possui uma ficha, a transição T1 fica habilitada e a transição T2 fica desabilitada.

Na RPCAM de Guan, ao invés de utilizar este arco inibidor, uma ficha colorida c de comando no lugar P1 impede o disparo de T2 quando houver fichas em P3 e uma outra ficha colorida c (outro comando) habilita T2 em caso contrário.

No esquema de Guan, o início do teste de fichas em P3 é o apresentado na Figura 3-14.

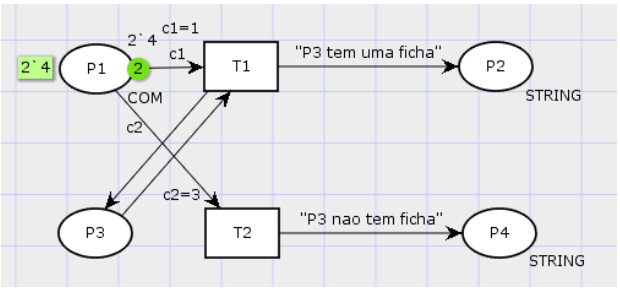


Figura 3-14 Teste de P3 original

A lista de comandos da ficha colorida c é a apresentada na Tabela 3.2

Tabela 3-2 Lista de comandos da ficha colorida c

1 – Habilita a transição $T1$
2 – Desbloqueia a ficha no lugar $P1$
3 – Habilita a transição $T2$
4 – Desabilita as transições $T1$ e $T2$
$P1$ representa um lugar da apresentação para começar o teste de zero
$P2$ significa que $P3$ possui uma ficha
$P3$ representa o lugar a ser testado
$P4$ significa que $P3$ não possui fichas (zero fichas)

A explicação a seguir ilustra o esquema de teste de P3 sem a utilização do arco inibidor.A marcação inicial é mostrada na Figura 3-14. No início, as transições T1 e T2 estão desabilitadas. Quando uma ficha chega no lugar P1, a ficha de recurso em P1 é pausada e o teste de zero começa. Os comandos de controle da ficha colorida c serão executados de acordo com a ordem dada.

A sequência de execução é importante, a transição T1 deve ser ativada antes da transição T2 e, caso contrário, produzirá resultados errados. Suponha que P3 não contém ficha, os primeiros comandos 1 e 2 são executados, a transição T1 é ativada e a ficha de recurso em P1 é desbloqueada porque não há ficha no lugar P3, então T1 não pôde disparar, nenhuma ficha será criada no lugar P2. Então, o comando 3 é executado, a transição T2 está habilitada. Como o lugar de entrada da transição T2 contém uma ficha de recurso, T2 dispara e a ficha de recurso no lugar P1 se moverá para o lugar P4.

Por fim, o comando 4 é executado, as transições T1 e T2 são desabilitadas. Como resultado, não há nenhuma ficha no lugar P2 e há uma ficha no lugar P4, indicando que P3 não contém ficha.
Repetindo toda a simulação da rede, mas desta vez com P3 contendo uma ficha. Quando a transição T1 está ativada, para todos os seus lugares de entrada que contêm fichas, T1 dispara e as fichas nos lugares P_1 e P_3 serão removidas e uma ficha aparecerá no lugar P2.

Depois que este comando for concluído, T2 será habilitada, mas agora não há nenhuma ficha no lugar P2 e T2 não vai dispar. Então, nenhuma ficha aparecerá no lugar P4, ou seja, a RPCAM consegue detectar zero fichas em um lugar e, consequentemente, tanto a RPCAM como a RPC conseguem modelar as Máquinas de Turing como demonstrou Agerwala em (AGERWALA, 1974).

3.6 Redes de Petri para Tomada de Decisão

3.6.1 Rede de Petri Nebulosa (RPN)

Um dos primeiros trabalhos focando a utilização de redes de Petri em problemas de Tomada de Decisão foi desenvolvido por Looney (LOONEY, 1988). Seu trabalho foi desenvolvido como uma evolução de um trabalho anterior (LOONEY e ALFIZE, 1987) no qual os autores propuseram uma maneira de desenvolver um raciocínio lógico através de matrizes booleanas conforme o exemplo por ele fornecido e adaptado a seguir.

Condições a serem transcritas:
C(1) – “A tensão da bateria está baixa”;
C(2) – “A bobina de partida não funciona”;
C(3) – “A partida não gira quando ligada”;
C(4) – “A partida gira, mas o motor não gira”.

Regras que relacionam as condições:
Regra(3,1)=1 representa $[C(1) \Rightarrow C(3)]$
Regra(4,2)=1 representa $[C(2) \Rightarrow C(4)]$

Matricialmente, estas regras são inseridas em uma matriz 4x4, com a diagonal igual a 1 e os demais elementos iguais a 0.

$$Regra = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Considerando que C(1) seja verdadeira, então esta condição é codificada em um vetor x, então

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.12}$$

e a consequência da aplicação das regras dado que a condição C(1) é verdadeira é calculada como

$$y = Regra.x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (3.13)$$

A matriz transposta possibilita o cálculo para trás, ou seja, conhecidos os sintomas pode-se calcular as possíveis causas desses sintomas.

No exemplo acima, conhecido o sintoma C(4), sua possível causa pode ser calculada através de

$$x = Regra^T.y = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad (3.14)$$

que pode ser interpretado como “A partida gira, mas o motor não gira” porque “A bobina de partida não funciona”.

Uma versão nebulosa deste esquema é obtida pela utilização de valores reais entre 0 e 1 denominados de grau de crença na veracidade das condições listadas.

Assim, as regras nebulosas correspondentes às duas listadas acima poderia ser:
Regra(3,1)=0,75 representa $[C(1) \Rightarrow 0{,}75.C(3)]$
Regra(4,2)=0,66 representa $[C(2) \Rightarrow 0{,}66.C(4)]$

Um grau de confiança de 50% na condição C(1) é representado por x(1)=0,5 e a equação utilizada para o cálculo das consequências fica

$$y = Regra.x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0{,}75 & 0 & 1 & 0 \\ 0 & 0{,}66 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 0{,}5 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0{,}5 \\ 0 \\ 0{,}5 \\ 0 \end{bmatrix} \quad (3.15)$$

Nesse cálculo, a operação lógica AND é substituída pela função MIN e a operação lógica OR é substituída pela função MAX, ou seja,
y(1)=Max(MIN(1, 0,5), MIN(0, 0), MIN(0, 0), Min(0,0))=0,5
y(2)=Max(MIN(0, 0,5), MIN(1, 0), MIN(0, 0), Min(0,0))=0
y(3)=Max(MIN(0,75, 0,5), MIN(0, 0), MIN(1, 0), Min(0,0))=0,5
y(4)=Max(MIN(0, 0,5), MIN(0,66, 0), MIN(0, 0), Min(1,0))=0

Também na lógica nebulosa a matriz transposta permite o cálculo para trás, ou o cálculo das causas dos sintomas observados. Supondo que o sintoma C(4) seja observado com um grau de crença de 90%, este cálculo para trás seria calculado como segue

$$x = Regra^T.y = \begin{bmatrix} 1 & 0 & 0{,}75 & 0 \\ 0 & 1 & 0 & 0{,}66 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0{,}9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0{,}66 \\ 0 \\ 0{,}9 \end{bmatrix} \quad (3.16)$$

Cada elemento do vetor x foi calculado da seguinte forma:
x(1)=Max(MIN(1, 0), MIN(0, 0), MIN(0,75, 0), Min(0, 0,9))=0
x(2)=Max(MIN(0, 0), MIN(1, 0), MIN(0, 0), Min(0,66, 0,9))=0,66
x(3)=Max(MIN(0, 0), MIN(0, 0), MIN(1, 0), Min(0, 0,9))=0
x(4)=Max(MIN(0, 0), MIN(0, 0), MIN(0, 0), Min(1, 0,9))=0,9

Este esquema matricial baseado em regras pode ser transposto para redes de Petri. Na Figura 3-15 está esquematizada uma rede de Petri de um sistema baseado nas seguintes regras:
Regra 1: $(C1 \wedge C2) \Rightarrow C4$
Regra 2: $C4 \Rightarrow C6$
Regra 3: $C5 \Rightarrow C3$
Regra 4: $C5 \Rightarrow C1$
Regra 5: $C6 \Rightarrow C'$ $(nó\ externo)$

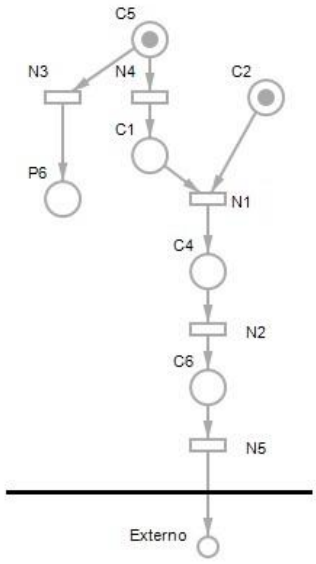


Figura 3-15 Rede de Petri baseada em regras

Supondo que as condições C2 e C5 sejam verdadeiras, na rede de Petri isto será indicado por uma ficha. Neste caso, as transições N3 e N4 disparam e as condições C1 e C3 se tornam verdadeiras.

Em seguida, a transição N1 fica habilitada por causa de C1 e C2 e dispara fazendo com que C4 se torne verdadeira. Como consequência, C6 também se torna verdadeira e provoca o envio de uma ficha para um sistema externo.

Para tornar esta rede de Petri nebulosa, é definido o conceito de ficha nebulosa (fuzzy token) onde esta ficha nebulosa pode assumir um valor real entre 0 e 1. Assim, um vetor de fichas nebulosas associado ao vetor de condições é definido, ou seja,
$\bar{F} = F(1), \cdots, F(k)$, onde k é a quantidade de condições do vetor $\bar{C} = C(1), \cdots, C(k)$.
O vetor $\bar{F}$ corresponde ao estado ou marcação da rede de Petri.

Cada arco desta rede de Petri pode ser associado a um valor nebuloso que será utilizado na avaliação de regras de implicação nebulosas. No caso de não se desejar que as transições disparem em níveis baixos, é definido um limiar de disparo para cada transição N(i), ou seja, um nível de decisão D(i). Note-se que as transições dessa rede de Petri se comportam como os neurônios das redes neurais que disparam quando a função dependente das entradas ultrapassa um certo limiar característico.

Admitindo valores nebulosos às fichas da Figura 3-15, supondo que F(1)=0,5 e F(2)=0,6 são as fichas associadas às condições C1 e C2, respectivamente, e que a transição N1 possua um limiar igual a D1 = 0,4, então o cálculo para determinar se N1 dispara ou não é o seguinte:

$$N1 = F(1) \wedge F(2) = MIN(0{,}5\ , 0{,}6) = 0{,}5 \tag{3.17}$$

$$N1 = 0{,}5 > D1 = 0{,}4 \ \Rightarrow N1\ dispara \tag{3.18}$$

e uma ficha nebulosa no valor de 0,5 será transmitida para C4.

O algoritmo para a tomada de decisão neste tipo de rede de Petri Nebulosa é o seguinte, considerando que $\bar{F}$ é o vetor de Fichas Nebulosas, $\bar{N}$ é o vetor das transições e $\bar{D}$ é o vetor com os limiares de disparo de cada transição.

1- Inicializar todas as fichas em $\bar{F}$ e todos os valores nebulosos das transições em $\bar{N}$ com o valor zero. Efetuar as leituras dos novos valores das fichas a partir dos dispositivos de entrada, atualizando a situação inicial do vetor $\bar{F}$.
2- Posicionar todos os componentes do cálculo de cada N(i) para 1 (preparação para o cálculo da função MIN). Propagar todas as fichas nebulosas do vetor $\bar{F}$ através da matriz de regras a fim de obter o novo valor do vetor de estado das transições $\bar{N}$.
3- Comparar cada componente do vetor de estado das transições N(i) com o valor de decisão correspondente D(i). Se $N(i) < D(i)$ então $N(i) = 0$, caso contrário, deixar $N(i)$ com seu valor atual.
4- Atualizar o vetor de estado de Fichas $\bar{F}$ a partir das condições. Transferir todos os valores nebulosos de $\bar{N}$ seguindo o caminho de seus arcos de saídas para os nós apropriados de condições através do cálculo de MAXimização (que é a generalização nebulosa da função OR) de cada N(i) de entrada com suas fichas nebulosas F(j) correspondendo a cada nó C(j).
5- Se a maior cadeia de implicação foi atravessada, então transferir o vetor de estado nebuloso $\bar{F}$ para outro subsistema, caso contrário, voltar ao passo 2.
6- Verificar se existe uma condição de término vinda de um subsistema externo. Se esta condição existir, então parar, caso contrário, vá para o passo 1.

Aplicando este algoritmo à rede de Petri da Figura 3-15 e supondo que a quantidade de fichas nebulosas sejam iguais a zero, exceto F(2)=0,8 e F(5)=0,5, obtém-se a seguinte sequência de passos.

Passo 1: Aquisição de valores iniciais

$$\bar{F} = [0 \quad 0{,}8 \quad 0 \quad 0 \quad 0{,}5 \quad 0] \tag{3.19}$$

Passo 2: Inicializar todos os N(i) que são 0 para 1 para cálculo da função MIN

$$N(1) = MIN\left(MIN\big(N(1), F(1)\big), F(2)\right) = MIN(MIN(1{,}0), 0{,}8) = 0 \tag{3.20}$$

$$N(2) = MIN\big(N(2), F(4)\big) = MIN(1, 0) = 0 \tag{3.21}$$

$$N(3) = MIN\big(N(3), F(5)\big) = MIN(1, 0{,}5) = 0{,}5 \tag{3.22}$$

$$N(4) = MIN\big(N(4), F(5)\big) = MIN(1, 0{,}5) = 0{,}5 \quad (3.23)$$

$$N(5) = MIN\big(N(5), F(6)\big) = MIN(1, 0) = 0 \quad (3.24)$$

Passo 3: Comparação com os limiares de disparo (não utilizado)

Passo 4: Atualização do estado das fichas nebulosas

$$F(1) = MAX\big(F(1), N(4)\big) = MAX(0, 0{,}5) = 0{,}5 \quad (3.25)$$

$$F(2) = 0{,}8 \quad (3.26)$$

$$F(3) = MAX\left(MAX\big(F(3), N(3)\big), N(5)\right) = MAX(MAX(0, 0{,}5), 0) = 0{,}5 \quad (3.27)$$

$$F(4) = MAX\big(F(4), N(1)\big) = MAX(0, 0) = 0 \quad (3.28)$$

$$F(5) = 0{,}5 \quad (3.29)$$

$$F(6) = MAX\big(F(6), N(2)\big) = MAX(0, 0) = 0 \quad (3.30)$$

Iteração do algoritmo: retorno ao Passo 2

Passo 2: Inicializar todos os N(i) que são 0 para 1 para cálculo da função MIN

$$N(1) = MIN\left(MIN\big(N(1), F(1)\big), F(2)\right) = MIN(MIN(1, 0{,}5), 0{,}8) = 0{,}5 \quad (3.31)$$

$$N(2) = MIN\big(N(2), F(4)\big) = MIN(1, 0) = 0 \quad (3.32)$$

$$N(3) = MIN\big(N(3), F(5)\big) = MIN(0{,}5, 0{,}5) = 0{,}5 \quad (3.33)$$

$$N(4) = MIN\big(N(4), F(5)\big) = MIN(0{,}5, 0{,}5) = 0{,}5 \quad (3.34)$$

$$N(5) = MIN\big(N(5), F(6)\big) = MIN(1, 0) = 0 \quad (3.35)$$

Passo 3: limiares de disparo não utilizados

Passo 4: Atualização do estado das fichas nebulosas

$$F(1) = MAX\big(F(1), N(4)\big) = MAX(0{,}5, 0{,}5) = 0{,}5 \quad (3.36)$$

$$F(2) = 0{,}8 \quad (3.37)$$

$$F(3) = MAX\left(MAX\big(F(3), N(3)\big), N(5)\right) = MAX(MAX(0{,}5, 0{,}5), 0) = 0{,}5 \quad (3.38)$$

$$F(4) = MAX\big(F(4), N(1)\big) = MAX(0, 0{,}5) = 0{,}5 \quad (3.39)$$

$$F(5) = 0{,}5 \quad (3.40)$$

$$F(6) = MAX\big(F(6), N(2)\big) = MAX(0, 0) = 0 \quad (3.41)$$

O vetor de estado nebuloso das Fichas $\bar{F}$ evoluiu da seguinte forma:

$$\bar{F}_0 = [0 \quad 0{,}8 \quad 0 \quad 0 \quad 0{,}5 \quad 0] \tag{3.42}$$

$$\bar{F}_1 = [0{,}5 \quad 0{,}8 \quad 0{,}5 \quad 0 \quad 0{,}5 \quad 0] \tag{3.43}$$

$$\bar{F}_2 = [0{,}5 \quad 0{,}8 \quad 0{,}5 \quad 0{,}5 \quad 0{,}5 \quad 0] \tag{3.44}$$

Iterando mais uma vez, o vetor $\bar{F}$ resultará em

$$\bar{F}_3 = [0{,}5 \quad 0{,}8 \quad 0{,}5 \quad 0{,}5 \quad 0{,}5 \quad 0{,}5] \tag{3.45}$$

Mais uma iteração e a transição $N(5)$ também resultará em 0,5 e, como consequência, ela será habilitada e transferirá a ficha nebulosa para um lugar externo com valor igual a 0,5.

3.6.2 Rede de Petri Nebulosa utilizada na Representação de Conhecimento

Para a utilização das Redes de Petri Nebulosas (RPN) na tomada de decisão, um importante trabalho com a formalização deste tipo de rede e especificação dos algoritmos para cálculo do raciocínio nebuloso foi o apresentado por Chen et al. (CHEN, KE e CHANG, 1990).

Na área de lógica nebulosa, a estrutura “Regra de Produção Nebulosa” é a que tem sido utilizada para a representação do conhecimento. Uma base de conhecimentos é constituída por um conjunto de Regras de Produção Nebulosas. Basicamente, uma Regra de Produção Nebulosa é uma regra que descreve uma relação nebulosa entre duas proposições.

Regra de Produção Nebulosa Genérica:

$$\boldsymbol{Se\ d_j\ Então\ d_k\ (Fator\ de\ Certeza\ FC = \mu)} \tag{3.46}$$

onde

- d_j e d_k são proposições que podem conter variáveis nebulosas do tipo “alto”, “muito pequeno”, “frio”, etc. A verdade de cada proposição é um número real no intervalo [0, 1].
- μ é um número real, também no intervalo [0, 1], que representa o grau de crença na validade da regra. Quanto maior esta crença, mais próximo de 1 será este fator de certeza.

A Regra de Produção Nebulosa funciona da seguinte maneira:

- definindo-se um valor limiar $\lambda \in [0,1]$ e chamando de y_j o grau de verdade da proposição d_j com $y_j \in [0,1]$, estes dois valores devem ser comparados.
- caso $y_j \geq \lambda$, então a regra pode ser disparada e o grau de verdade da proposição d_k é $y_j * \mu$.
- caso $y_j < \lambda$, então a regra não pode ser disparada.

Definição de RPN de Chen:

Uma Rede de Petri Nebulosa é uma 8-tupla

$$RPN = (P, T, D, I, O, f, \propto, \beta) \tag{3.47}$$

onde

$P = \{p_1, p_2, \cdots, p_n\}$ é $um\ conjunto\ finito\ de\ lugares$,

$T = \{t_1, t_2, \cdots, t_m\}$ é $um\ conjunto\ finito\ de\ transições$,

$D = \{d_1, d_2, \cdots, d_n\}$ é $um\ conjunto\ finito\ de\ proposições, P \cap T \cap D = \emptyset, |P| = |D|$,

$I: T \to P^{\infty}$ é $a\ função\ de\ entrada\ que\ mapeia\ transições\ a\ um\ conjunto\ de\ lugares$,

$O: T \to P^{\infty}$ é $a\ função\ de\ saída\ que\ mapeia\ transições\ a\ um\ conjunto\ de\ lugares$,

$f: T \to [0,1]$ é $uma\ função\ associativa\ que\ mapeia\ transições$

$a\ um\ números\ reais\ no\ intervalo\ 0\ e\ 1$,

$\propto: P \to [0,1]$ é $uma\ função\ associativa\ que\ mapeia\ lugares$

$a\ um\ números\ reais\ no\ intervalo\ 0\ e\ 1$,

$\beta: P \to D$ é $uma\ função\ associativa\ que\ mapeia\ lugares\ a\ proposições$.

Uma regra de produção nebulosa genérica pode ser representada por uma RPN da forma indicada na Figura 3-16.

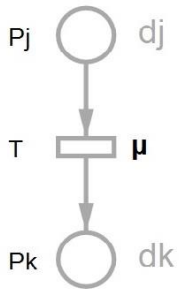
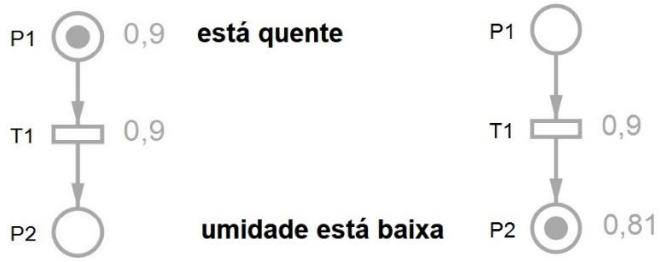


Figura 3-16 Rede de Petri Nebulosa (RPN)

Um caso particular de regra de produção nebulosa representada por uma RPN está esquematizado na Figura 3-17.



(antes da transição) (depois da transição)

Figura 3-17 Caso particular de uma RPN

A RPN da Figura 3-17 é definida por $RPN = (P, T, D, I, O, f, \propto, \beta)$, onde

$P = \{P_1, P_2\}$,

$T = \{T_1\}$,

$D = \{d_1 = \text{"está quente"}, d_2 = \text{"}umidade\ está\ baixa\text{"}\}$,

$I(T_1) = \{P_1\}$,

$O(T_1) = \{P_2\}$,

$f(T_1) = 0{,}9$

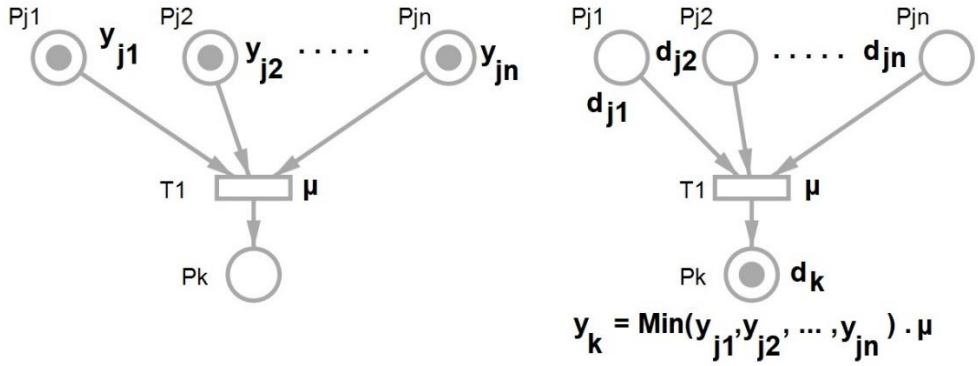
$\propto (P_1) = 0{,}9$; $\propto (P_2) = 0{,}0$

$\beta(P_1) = \text{"}está\ quente\text{"}$;$(P_2) = \text{"}umidade\ está\ baixa\text{"}$

Segundo Looney (LOONEY e ALFIZE, 1987), existem quatro tipos de regras compostas de produção nebulosas.

O tipo 1 possui o seguinte formato:

$$\boldsymbol{Se\ d_{j1} E\ d_{j2} E\ \cdots E\ d_{jn}\ Então\ d_k\ (Fator\ de\ Certeza\ FC = \mu)} \qquad (3.48)$$

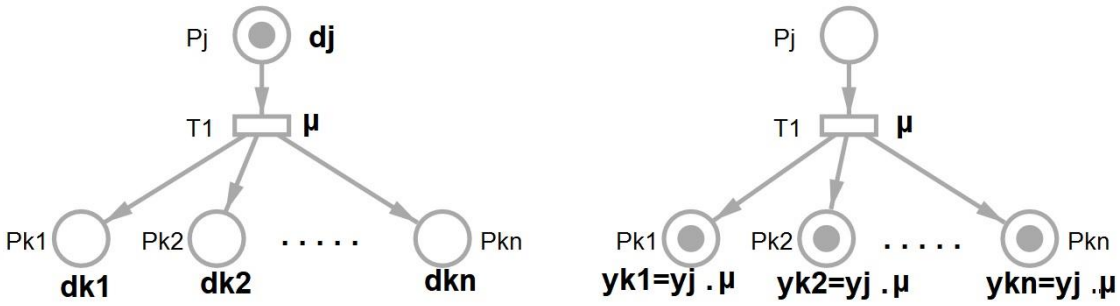


(antes do disparo de T1) (depois do disparo de T1)

Figura 3-18 RPN tipo 1

O tipo 2 possui o seguinte formato:

$$\boldsymbol{Se\ d_j\ Então\ d_{k1}E\ d_{k2}E \cdots E\ d_{kn}(Fator\ de\ Certeza\ FC = \mu)} \quad (3.49)$$

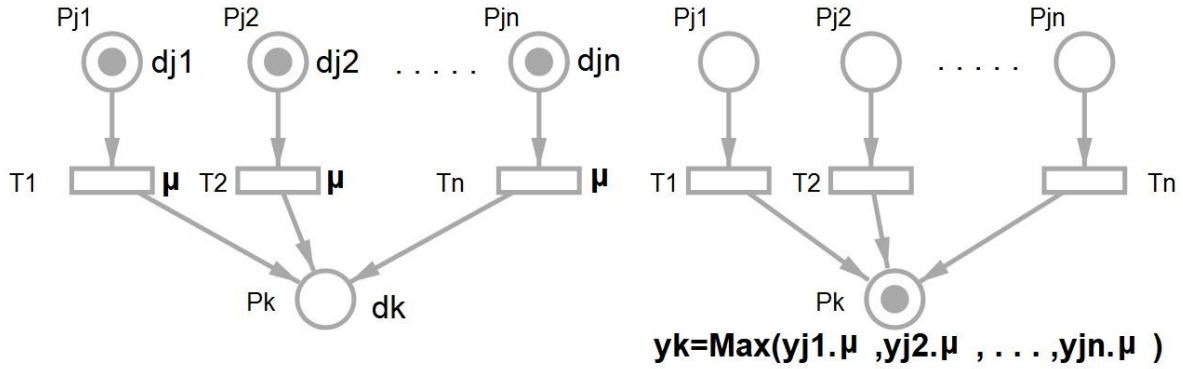


(antes do disparo de T1) (depois do disparo de T1)

Figura 3-19 RPN tipo 2

O tipo 3 possui o seguinte formato:

$$\boldsymbol{Se\ d_{j1}Ou\ d_{j2}Ou\ \cdots Ou\ d_{jn}\ Então\ d_k\ (Fator\ de\ Certeza\ FC = \mu)} \quad (3.50)$$



(antes dos disparos das n transições) (depois dos disparos das n transições)

Figura 3-20 RPN tipo3

O tipo 4 possui o seguinte formato:

$$\boldsymbol{Se\ d_j\ Então\ d_{k1}Ou\ d_{k2}Ou \cdots Ou\ d_{kn}(Fator\ de\ Certeza\ FC = \mu)} \quad (3.51)$$

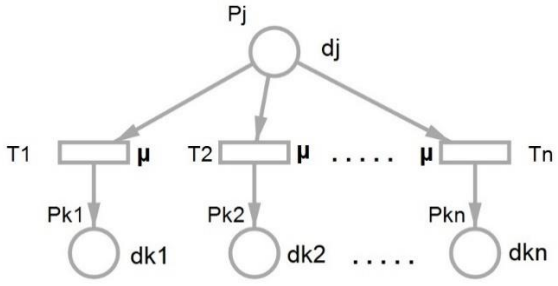


Figura 3-21 RPN tipo 4

4 MÉTODO DE FLEXIBILIZAÇÃO DE SISTEMAS VIA TECNOLOGIA ADAPTATIVA

Para a flexibilização de sistemas definimos a rede de Petri colorida adaptativa utilizando o esquema proposto por Camolesi.

4.1 Rede de Petri Adaptativa (RPA)

Uma rede de Petri adaptativa (RPA) foi definida por Camolesi (CAMOLESI, 2007) a partir do esquema dispositivo subjacente mais camada adaptativa da seguinte forma:

$$RPA = (C_0, AR_0, \Sigma, c_0, A, NA, BA, AA) \text{ na configuração inicial } c_0. \quad (4.1)$$

Estímulos de entrada movimentam a RPA para a próxima configuração se, e somente se, uma ação adaptativa não-vazia for executada.

No k-ésimo passo temos:

$$RPA_k = (C_k, AR_k, \Sigma, c_k, A, NA, BA, AA), \text{ onde} \quad (4.2)$$

$$RPA = (RP_0, AM) \quad (4.3)$$

é formado por um dispositivo inicial subjacente (RP_0) e um mecanismo adaptativo AM;

RP é o dispositivo Rede de Petri no passo k. RP_0 é o dispositivo subjacente inicial e o conjunto CR_0 representa o comportamento não-adaptativo inicial;

C_k é o conjunto de todos os possíveis comportamentos de RP no passo k e $c_k \epsilon C_k$ é o seu comportamento inicial no passo k;

ε (“cadeia vazia”) denota ausência de elemento válido;

Σ é o conjunto de todos os possíveis eventos de que se compõem a cadeia de entrada;

$A \subseteq C$ é o subconjunto de configurações de aceitação de RP;

$F = C - A$ é o conjunto das configurações de rejeição de RP;

BA e AA são conjuntos de ações adaptativas, que incluem a ação vazia;

$w = w_1 w_2 \ldots w_n$ é a cadeia de entrada;

NA é um conjunto finito de todos os símbolos que podem ser gerados como saídas por RPA em resposta à aplicação de regras adaptativas;

AR_k é o conjunto das regras adaptativas que definem o comportamento adaptativo de RPA no passo k e é dado por uma relação $AR_k \subseteq BA \times \Sigma \times C \times RP \times AA$.

AR_0 define o comportamento inicial da RPA e as ações adaptativas de inserção ou eliminação de lugares e transições vão transformando o conjunto de regras.

As regras $reg \in AR_k$ são da forma (<ba>,(P,T,I,O),<aa>) e operam da seguinte forma:

Um símbolo $\sigma \epsilon \Sigma$ faz *reg* executar a ação $ba\, \epsilon\, BA$. Se a ação de ba eliminar *reg* de AR_k, a execução de *reg* é abortada, caso contrário, aplica-se a regra subjacente de *reg* = (P, T, I, O). Finalmente, executa-se a ação adaptativa $aa\, \epsilon AA$.

4.2 Rede de Petri Colorida Adaptativa (RPCA)

Uma rede de Petri colorida (RPCA) é definida tendo como dispositivo subjacente uma rede de Petri colorida auto-modificável (RPSub) no qual o conjunto de comandos foi reduzido para o indicado na Tabela 4-1.

Ao contrário da RPCAM que permite a criação e eliminação de lugares, transições e arcos, a RPSub somente permitirá a criação e eliminação de regras, onde regras são conjuntos de lugares, arcos e transições que formam uma unidade, como uma sub-rede.

$$RPSub = \{P, T, A, U, C, M\} \text{ onde } P \cap T \neq \emptyset. \tag{4.4}$$

$P = \{p_1, p_2, p_3, \cdots, p_m\}$ é um conjunto finito de lugares representando as condições de uma regra ou as consequências da aplicação de uma regra, onde $m > 0$.

$T = \{t_1, t_2, t_3, \cdots, t_n\}$ é um conjunto finito de transições representando a execução de uma regra, onde $n > 0$.

$A: \{P \times T\} \cup \{T \times P\}$ é um conjunto finito de arcos representando a relação de fluxo.

$U = \{u_1, u_2, u_3, \cdots, u_l\}$ é o conjunto finito de cores; cada cor pode representar um tipo de comando da rede, onde $l > 0$.

$C = \{c_1, c_2, c_3, \cdots, c_k\}$ é o conjunto finito de comandos, onde $k > 0$ (por exemplo, “Criar Regra”, “Eliminar Regra”, ”Consultar Regra”).

$M : P \rightarrow \{I_r^+, I_c^+\}$ é um mapeamento do conjunto de lugares para um conjunto de números inteiros e representa uma marca de uma rede, onde (I_r – conta o número de fichas de recursos; I_c – conta o número de fichas coloridas); I_r^+= {0,1,2, . . .} e I_c^+ = {0,1,2, . . . }.

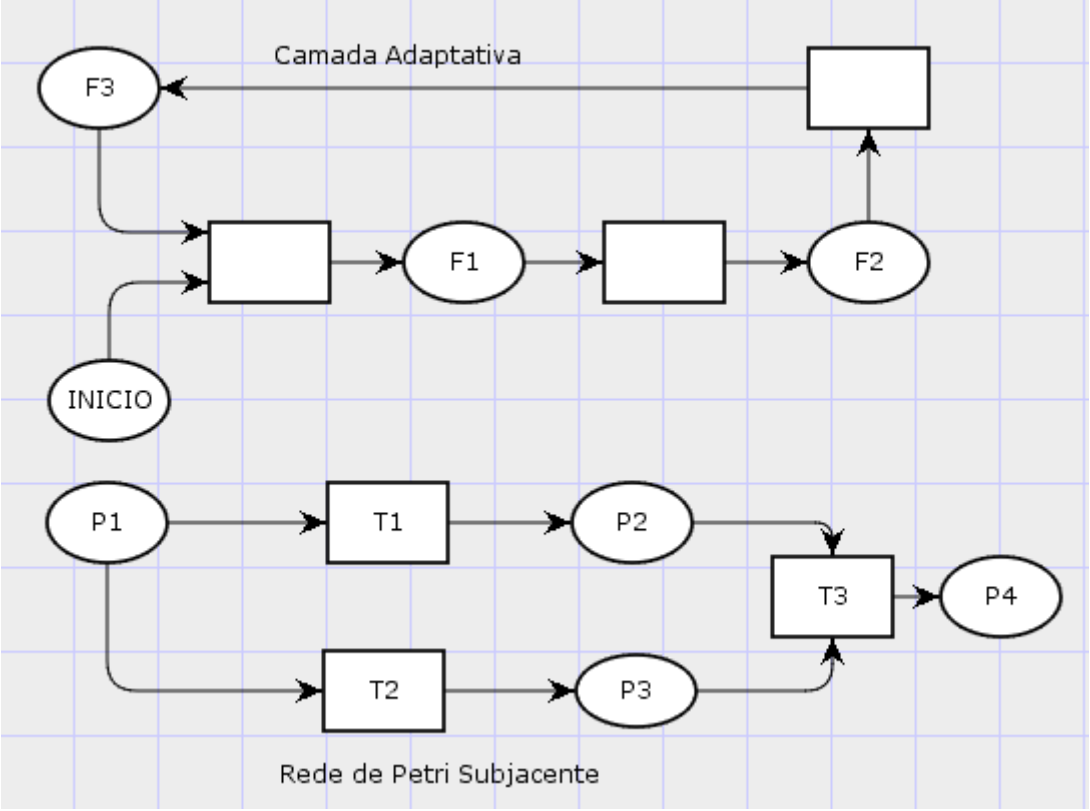


Figura 4-1 Rede de Petri Colorida Adaptativa

Tabela 4-1 – Relação de comandos das fichas de controle

1 – Criar regra R entre Pi e Pj	(Inserção de regra)
2 – Eliminar regra R entre Pi e Pj	(Eliminação de regra)
3 – Consultar regra R entre Pi e Pj	(Inspeção de regra)

Existem duas camadas interdependentes, a camada adaptativa, que é uma sub-rede de Petri de controle, e o dispositivo subjacente que é uma outra sub-rede de Petri. Os lugares da camada adaptativa são diferentes em sua natureza dos lugares da rede de Petri subjacente.

Cada lugar da camada adaptativa possui associado um conjunto de funções que são capazes de alterar a topologia da rede de Petri subjacente, ou seja, reconfiguram a rede de Petri subjacente.

$$RPCA = (RPSub, CA) \quad (4.5)$$

onde

RPSub é a rede de Petri colorida auto-modificável definida acima,

CA = (FA, RA) é a camada adaptativa.

Por sua vez, a camada adaptativa é composta pelo conjunto de funções adaptativas (FA) e pelo conjunto de regras (RA) que deverá ser inserido na ou excluído da rede de Petri Colorida Adaptativa através da execução das funções adaptativas.

As funções adaptativas básicas são de inspeção, inserção ou incorporação e exclusão de uma regra.

4.2.1 Flexibilização de Tabela de Decisão via RPCA

Na Tabela 4.2 temos esquematizado um exemplo de conjunto de regras a serem inseridas em uma RPCA.

Tabela 4-2 Regras para serem transformadas em sub-redes da RPCA

		Regras				
		R1	R2	R3	R4	R5
Condições	A	A = 1	A > 4	A=2	A=9	A=10
	B	B= TRUE	B= FALSE	B	B	B
	C	C >7	C ≠6	C=6	C=2	C<1
Ações/ Decisões	D	A+1	8	3	A-5	6
	E	TRUE	FALSE	TRUE	TRUE	FALSE

A regra R1 possui três condições (A=1, B=TRUE, C>7) e duas ações (D=A+1, E=TRUE). Para transformar esta regra em uma parte de uma rede de Petri, é utilizado um modelo (Lugar,Transição,Lugar) esboçado na Figura 4-3.

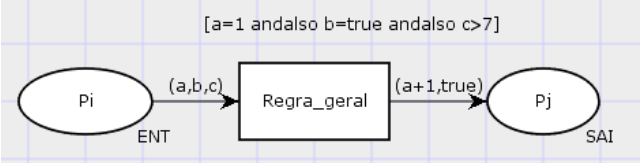


Figura 4-2 Modelo básico para transcrever uma regra da Tabela 4-2.

Conforme as regras vão sendo inseridas, elas vão sendo agrupadas hierarquicamente, ou seja, a execução das regras vai depender da sequência de decisões que forem ocorrendo na rede de Petri.

Na Figura 4-3 está esquematizado o resultado da inserção de três regras que compõem um certo subconjunto que compõem o total de regras.

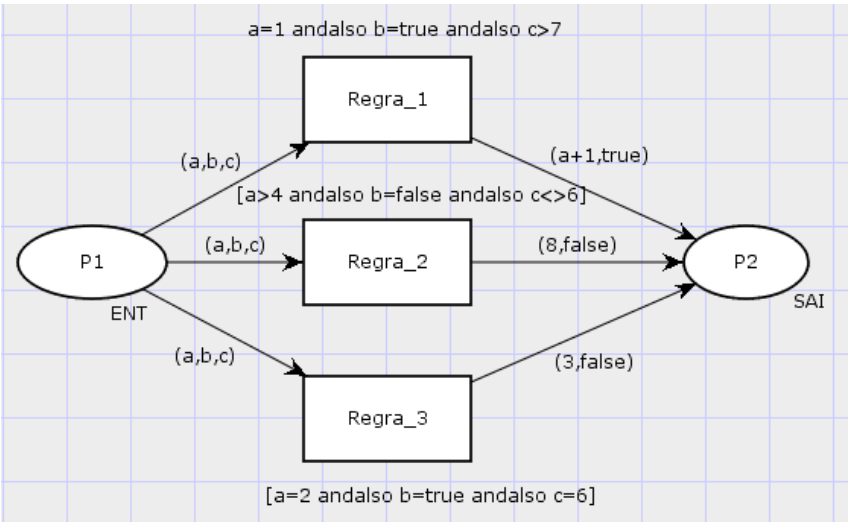


Figura 4-3 Subconjunto de regras em uma hierarquia

Ao inserirmos estas regras na rede de Petri, o conhecimento gerado pelo especialista passa a ser incorporado na rotina de acompanhamento do empreendimento, alertando o gestor em caso de alguma decisão implicar em não conformidade com as regras.

4.2.2 Flexibilização de Redes Neurais RBF via RPCA

O mesmo princípio de dividir um sistema em uma parte subjacente e uma camada adaptativa pode ser utilizado na flexibilização das redes neurais artificiais.

Na Figura 4-4, uma rede RBF genérica está esquematizada como uma rede de Petri subjacente e uma camada adaptativa constituída por uma sub-rede de Petri capaz de controlar a rede subjacente via fichas de controle aparece associada à rede RBF.

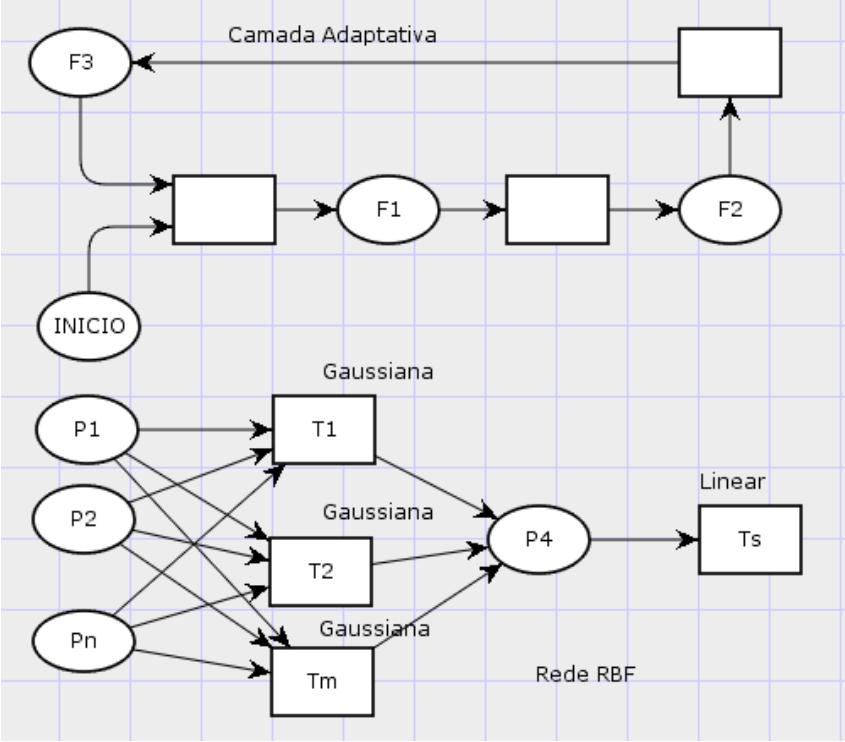


Figura 4-4 Rede RBF embutida em uma RPCA

4.2.3 Flexibilização das Redes de Petri Nebulosas via RPCA

O mesmo princípio utilizado para as redes RBF pode ser repetido para as redes de Petri Nebulosas (RPN) conforme apresentado na Figura 4-5.

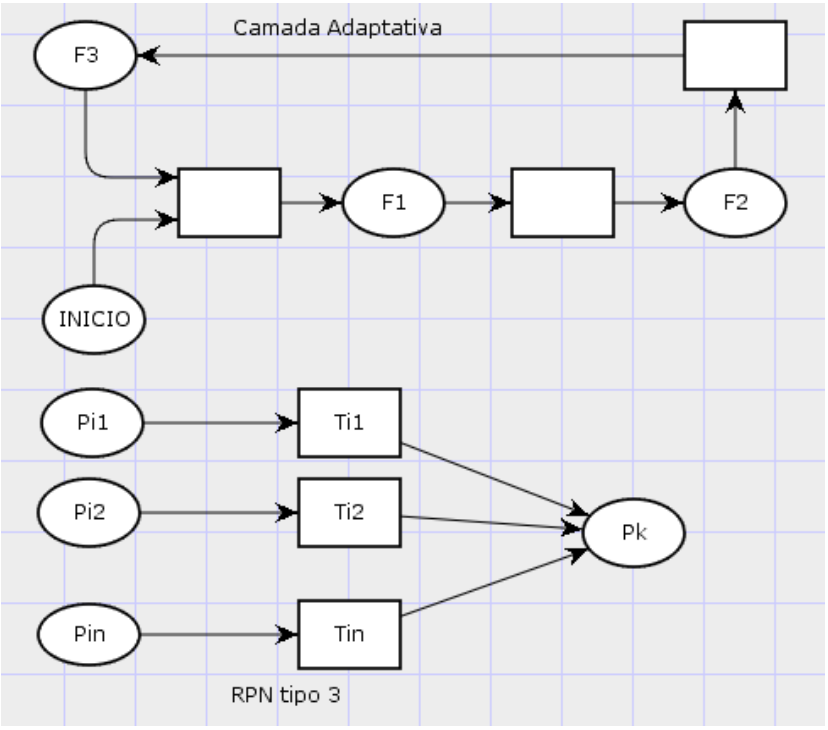


Figura 4-5 Rede de Petri Nebulosa inserida em uma RPCA

Nos dois casos de redes neurais inseridas em uma RPCA as fichas de controle da sub-rede de Petri da camada adaptativa podem fazer o dispositivo subjacente operar em modo de aprendizagem, modo de operação, ou mesmo em um modo de análise dos parâmetros da rede para maior compreensão do conhecimento armazenado na rede.

4.3 Aplicação da Tecnologia Adaptativa em Manufatura

Esta aplicação foi apresentada em (GUIBU e NETO, 2017).
Na Figura 4-6 está esquematizado o processo de fabricação de um produto a partir de 4 matérias primas, utilizando-se três máquinas.

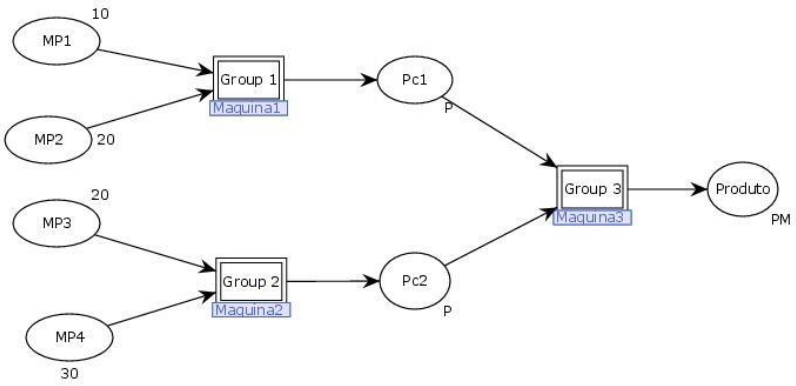


Figura 4-6 Rede de Petri de fabricação de um produto a partir de 4 matérias primas

A máquina 1 transforma as matérias primas MP1 e MP2 na peça 1, a máquina 2 transforma as matérias primas MP3 e MP4 na peça 2, e a máquina 3 utiliza as duas peças Pc1 e Pc2 para gerar o Produto.

Na Figura 4-7 está esquematizada a sub-rede correspondente ao funcionamento da máquina 1 e que é o mesmo para as máquinas 2 e 3.

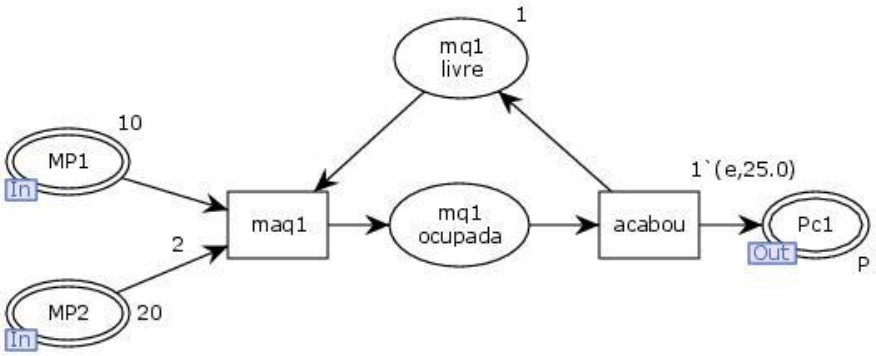


Figura 4-7 Rede de Petri de operação da Máquina 1

O funcionamento das máquinas especificado pela rede de Petri da Figura 4-4 foi definido pelo setor de fabricação.

Posteriormente, o setor de controle de qualidade detectou que o produto final não estava adequado, apresentando uma regra simples para melhorar a qualidade do produto.

Se x1 de Pc1 >= x2 de Pc2, então completar o produto, caso contrário, encaminhar Pc1 e Pc2 para reciclagem, onde x1 é um parâmetro da Peça 1 e x2 é um parâmetro da Peça 2.

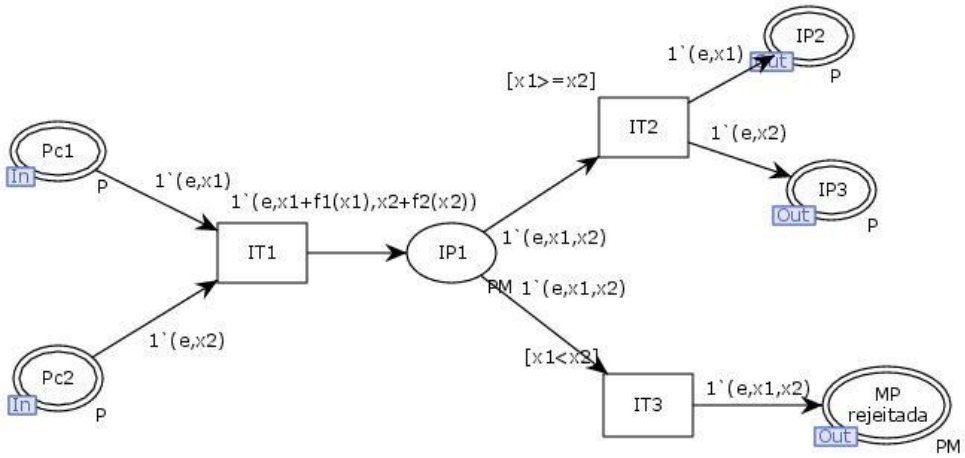


Figura 4-8 Sub-rede de Petri correspondente à regra de controle de qualidade inserida

A Figura 4-8 detalha a transformação da regra acrescentada na Rede de Petri de fabricação a partir de uma nova regra tipo IF- THEN, produzindo a nova Rede de Petri esquematizada na Figura 4-9.

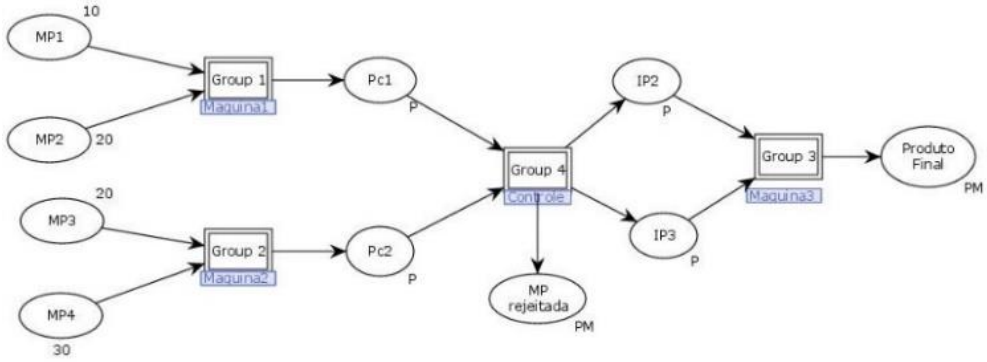


Figura 4-9 Rede de Petri modificada após a inserção de nova regra de controle

A transformação do formato de regras é útil quando o formato destino é mais adequado a uma aplicação que o formato original.

No exemplo fornecido, as Redes de Petri são mais utilizadas nos processos de fabricação que as Tabelas de Decisão. Além disso, diversas metodologias desenvolvidas para as Redes de Petri possibilitam detectar a existência de deadlocks e também a aplicação de esquemas que evitam seu aparecimento em redes de fabricação.

Este exemplo mostra como prover adaptatividade às redes de Petri coloridas, definindo as redes de Petri adaptativas coloridas (RPCA) que são capazes de incorporar tabelas de decisão em redes de Petri.
Esta característica tem implicação prática, pois possibilita a união de dois ou mais conjuntos de saberes que normalmente convivem em paralelo, interagem, mas muitas vezes sem um sincronismo adequado.
Como exemplo, as boas práticas de gestão muitas vezes são desconsideradas no dia-a-dia de um empreendimento em virtude do desconhecimento das consequências de certas decisões, nem sempre aparentes, ou seja, a ferramenta de acompanhamento de um empreendimento feita por um especialista não contempla os cuidados e recomendações de outro especialista.
Em outras áreas onde a utilização das redes de Petri está muito disseminada, também é possível obter um ganho como, por exemplo, nos sistemas de manufatura flexível.

4.4 Aplicação da Tecnologia Adaptativa em Tomada de Decisão

Esta aplicação apresenta uma mapeamento entre as Tabelas de Decisão Adaptativas Estendidas (TDAE) e as Redes de Petri Coloridas Adaptativas (RPCA) (GUIBU e NETO, 2017). Na área de sistemas de apoio à decisão, as ferramentas mais utilizadas pelos especialistas são as mesas de decisão, que deram origem a vários métodos para ajudar os gestores nas suas escolhas.

Entre os métodos desenvolvidos para o apoio à decisão estão os chamados métodos multicritérios, que envolvem a adoção de múltiplas tabelas de decisão hierarquicamente encadeadas. No processo de aperfeiçoamento das tabelas de decisão, são observadas novas características que, embora mais complexas, dão aos especialistas a capacidade de descrever seu modelo de trabalho de uma forma mais realista. Ao incorporar uma tabela de decisão em uma rede de Petri, podem ser utilizadas as ferramentas de simulação e análise disponíveis nos ambientes de desenvolvimento da rede de Petri, o que leva a um aumento da confiança nos critérios de decisão adotados.

4.4.1 Tabelas de decisão

A Tabela de Decisão é uma ferramenta auxiliar na descrição de procedimentos para resolver problemas complexos (TCHEMRA, 2009). Uma Tabela de Decisão Convencional, apresentada na Tabela 4.1, pode ser considerada como uma estrutura de dados composta por condições, ações e regras onde as condições são variáveis que devem ser avaliadas para tomada de decisão, ações são o conjunto de operações a serem realizadas dependendo das condições neste momento, e as regras são o conjunto de situações que são verificadas em resposta às condições.

Tabela 4-1 Tabela de decisão convencional

	Coluna de regras
Linhas de Condições	Valores das Condições
Linhas de Ações	Ações a serem tomadas

Uma regra é constituída pela associação de condições e ações em uma determinada coluna. O conjunto de colunas de regras deve cobrir todas as possibilidades que podem ocorrer dependendo das condições observadas e das ações a serem tomadas.

Dependendo das condições atuais de um problema, procuramos quais regras de tabela satisfazem essas condições:

• Se nenhuma regra satisfaz as condições impostas, nenhuma ação é tomada;

• Se apenas uma regra se aplica, então as ações correspondentes à regra são executadas;

• Se mais de uma regra satisfaz as condições, então as ações correspondentes às regras são aplicadas em paralelo.

• Uma vez aplicadas as regras, a tabela pode ser usada novamente.

• As regras de uma tabela de decisão são pré-definidas e novas regras só podem ser adicionadas ou excluídas revendo a tabela.

4.4.2 Tabelas de Decisão Adaptativas

Em 2001, Neto introduz a Tabela de Decisão Adaptativa (TDA) (NETO, 2001) a partir de um dispositivo adaptativo orientado por regras. Além da pesquisa de regras, uma TDA permite incluir ou excluir uma regra do conjunto de regras durante a operação do dispositivo. Como exemplo de seu potencial, Neto simula um autômato adaptativo para reconhecer frases de linguagens dependentes do contexto. Na TDA uma tabela de decisão convencional é o dispositivo subjacente ao qual um conjunto de linhas será adicionado para definir as funções adaptativas.

As funções adaptativas constituem a camada adaptativa do dispositivo adaptativo regido por regras. Modificar o conjunto de regras implica aumentar o número de colunas no caso de inserção de regra ou diminuir o número de colunas no caso de exclusão de regra. Em ambos os casos, a quantidade de linhas permanece fixa. A Tabela de Decisão Adaptativa (TDA) é capaz de mudar seu conjunto de regras como resposta a um estímulo externo através da ação de funções adaptativas (NETO, 2001). No entanto, a implementação de ações mais complexas não é uma tarefa simples devido à limitação das três operações elementares apoiadas pela TDA (TCHEMRA, 2009).

Quando um dispositivo de adaptação típico está em funcionamento e não encontra as regras aplicáveis, pára de executar, indicando que esta situação não foi prevista. Para dispositivos de operação contínua, que não têm aceitação ou rejeição de estados terminais, interromper sua execução reconheceria uma situação imprevista e constitui um erro.

4.4.3 Tabelas de Decisão Adaptativas Estendidas

Para superar este problema enfrentado por dispositivos de operação contínua, Tchemra (TCHEMRA, 2009) criou uma variante da TDA e chamou de Tabela de Decisão Adaptativa Estendida (TDAE), mostrado na Tabela 4.3.

Tabela 4-2 Tabela de Decisão Adaptativa Estendida

		Ações Adaptativas	**Regras**
Tabela de Decisão Convencional	Critérios	Conjunto de ações Adaptativas elementares	Valores dos critérios
	Alternativas		Ações a serem tomadas
Conjunto de funções auxiliares	Funções auxiliares		Funções Adaptativas a serem Chamadas
Camada adaptativa	Funções adaptativas		Ações adaptativas a serem tomadas

Na TDAE, a adaptabilidade não se aplica apenas durante a aplicação de uma regra, mas também na ausência de regras aplicáveis. Um dispositivo auxiliar modificador é consultado e a solução produzida pelo dispositivo modificador é incorporada na tabela sob a forma de uma nova regra, isto é, na repetição das condições chamadas de dispositivo modificador, a nova regra será executada e a Modificador não precisará ser chamado.

4.4.4 Redes de Petri Adaptativas na tomada de decisão

Esta aplicação utiliza o mesmo dispositivo adaptativo utilizado na aplicação anterior, a saber, a Rede de Petri Colorida Adaptativa (RPCA).

O exemplo a seguir foi adaptado de (TCHEMRA, 2009) para ilustrar a sequência das fases mencionadas acima, com base em um procedimento de decisão proposto em (SAATY, 1994). Este é um problema de decisão no qual o tomador de decisões precisa certificar e selecionar fornecedores de um determinado produto para sua empresa. Dois fornecedores A e B são analisados, de acordo com os julgamentos do tomador de decisão de acordo com selecionados para comparação:
C1 - Qualidade dos produtos;
C2 - Prazo para a entrega do produto;
C3 – Preço do produto e custo da transportadora.

Na fase I, critérios e alternativas ao problema são introduzidos em uma tabela de decisão convencional e o decisor pode criar um conjunto inicial de regras, como mostrado na Tabela 4.3.

Tabela 4-3 Tabela de Decisão Inicial

		Regra1	**Regra2**	**Regra3**	**Regra4**
Critérios	C1 - Qualidade	Y	Y	N	N
	C2 - Prazo	Y	Y	Y	Y
	C3 - Preço	N	Y	N	Y
Alternativas	A1 – Fornecedor A	X	X		X
	A2 – Fornecedor B			X	

Neste exemplo, as comparações entre os critérios são mostradas na Tabela 4.5, na qual o decisor julga os pares de critérios, obtendo a matriz de julgamentos recíprocos.

Tabela 4-4 Matriz de Julgamento

	C1	**C2**	**C3**
C1 - Qualidade	1	5	4
C2 - Prazo	1/5	1	1/3
C3 - Preço	1/4	3	1

Saaty estabeleceu uma relação entre critérios de comparação e valores numéricos que variam de 1 a 9. O valor 1 indica que os critérios possuem a mesma importância e esta é a razão da diagonal ser composta por 1's. O valor indica uma importância moderada de um critério sobre o outro. Na Tabela 4.5 o Preço é considerado moderadamente mais importante que o Prazo.

O valor 5 indica que um critério é fortemente mais importante que o outro, ou seja, na Tabela 4.5 a Qualidade é fortemente mais importante que o Prazo. O valor 4 é um valor intermediário entre 3 e 5. Nesta tabela os valores inversos representam a mesma relação entre os critérios, mas de forma inversa, ou seja, se o critério C_i é x vezes mais importante que o critério C_j, então o critério C_j é $^1/_x$ vezes mais importante que o critério C_i.

Tabela 4-5 Matriz de Julgamento Normalizada e Médias dos Critérios

	C1	**C2**	**C3**		**Médias**
C1 - Qualidade	0,69	0,56	0,75		0,67
C2 - Prazo	0,14	0,11	0,06		0,10
C3 - Preço	0,17	0,33	0,19		0,23

Depois de normalizar a matriz de julgamentos os pesos de cada critério são calculados a partir da média aritmética das linhas do critério, gerando o vetor de pesos:

$W = (0{,}67 \quad 0{,}10 \quad 0{,}23)^T$.

De acordo com o julgamento do tomador de decisão, o vetor com o peso de cada critério representa a importância relativa de cada um. Neste exemplo, os pesos resultantes indicam que o critério é mais importante em relação a outros:
Qualidade: 0,67 - Prazo: 0,10 - Preço: 0,23.

Neste ponto, é necessário verificar a coerência dos critérios julgamentos. A matriz de comparação entre os critérios da Tabela 4.5 é avaliada para a verificação do grau de consistência dos preceitos, que é dada pela razão de consistência (CR), como uma função da ordem da matriz:
A) autovalor λmax = 3,086
B) índice de consistência $CI = \frac{\lambda_{max}-3}{2} = 0{,}043$
C) razão de consistência $CR = \frac{CI}{ICR} = \frac{0{,}043}{0{,}58} = 0{,}074$ onde ICR é obtido da tabela de Saaty denominada Índice de Consistência Randômico (SAATY, 1994).

De acordo com (SAATY, 1994), o valor de CR = 0,074 indica que os julgamentos são consistentes e aceitáveis desde CR <0,01, caso contrário seria necessário rever a Tabela 4.5. A próxima operação é obter a matriz de desempenho. Para isso, as alternativas são comparadas aos pares, com cada um dos critérios. As comparações feitas pelo tomador de decisões no exemplo são mostradas na Tabela 4.6.

Tabela 4-6 Matriz de comparação de pares

	C1		**C2**		**C3**	
	A1	A2	A1	A2	A1	A2
A1 – Fornecedor A	1	7	1	5	1	1/3
A2 – Fornecedor B	1/7	1	1/5	1	3	1

Normalizando as matrizes, obtemos os seguintes valores:
$z_{1,1} = 0{,}875 \quad z_{1,2} = 0{,}83\, z_{1,3} = 0{,}25$
$z_{1,2} = 0{,}125 \quad z_{2,2} = 0{,}17\, z_{2,3} = 0{,}75$

que são as células da matriz de desempenho.

Tabela 4-7 Matriz de desempenho Z e Média por Fornecedor

	C1	C2	C3		Média
A1 – Fornecedor A	0,875	0,83	0,25		0,65
A2 – Fornecedor B	0,125	0,17	0,75		0,35

A partir da matriz de desempenho Z obtemos o vetor ax contendo as figuras indicando a importância relativa das alternativas.

$ax_1 = 0{,}65$ e $ax_2 = 0{,}35$

indicando que neste exemplo a alternativa A1 é muito melhor do que a alternativa A2 e o fornecedor A deve ser escolhido. A Figura 4.10 mostra a versão da rede Petri da tabela de decisão .

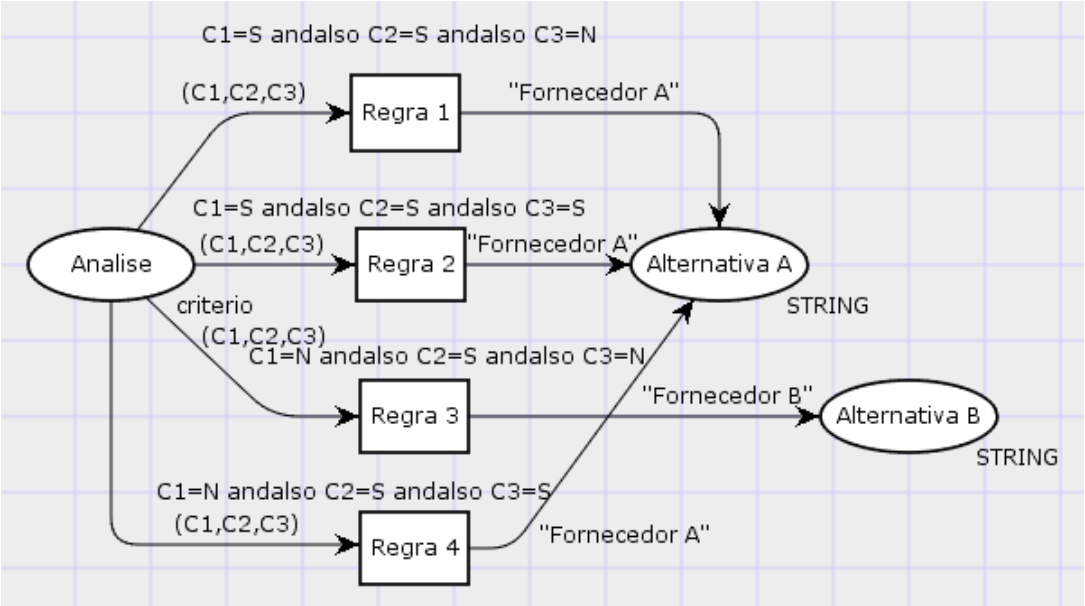


Figura 4-10 Rede de Petri equivalente à Tabela de Decisão Inicial

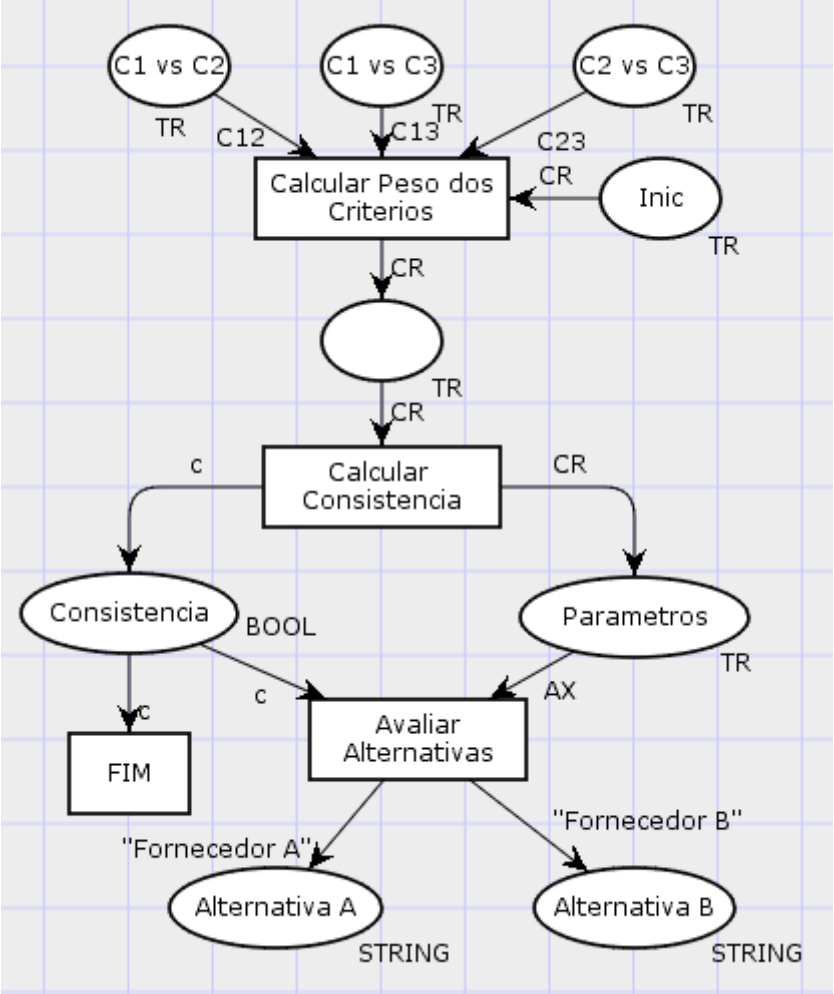


Figura 4-11 Rede de Petri do processo de tomada de decisão

5 CONCLUSÕES

Nesta tese apresentamos como a flexibilização de redes neurais e redes de Petri pode ser obtida através da definição de uma rede de Petri colorida adaptativa (RPCA) através de um esquema em que a camada adaptativa é ela própria uma rede de Petri na qual as fichas são comandos de inserção ou extração de regras do dispositivo subjacente.

Entre as diversas áreas de aplicação, duas áreas foram enfocadas: a de manufatura flexível e a de sistemas de apoio à decisão. No caso dos sistemas de apoio à decisão, procedimentos bem estabelecidos que utilizam tabelas de decisão podem ser usados em uma nova ferramenta que possui recursos de análise adicionais para detectar inconsistências nos procedimentos.

Nas atividades diárias, o especialista em negócios não compartilha seus conhecimentos com o gerente da fábrica, perdendo sinergia. Compartilhar dois conhecimentos que são geralmente isolados fornece uma sinergia. As boas práticas de gestão são muitas vezes ignoradas no dia-a-dia de uma empresa devido ao desconhecimento das consequências de certas decisões, que nem sempre são aparentes.

Na área de manufatura flexível, onde o uso de redes de Petri é generalizado, a utilização das tecnologias adaptativas é uma consequência natural da busca contínua de melhorias. As redes reconfiguráveis podem ser entendidas como um caso particular de redes adaptativas, onde a adaptação é conseguida através da reconfiguração da rede. A rede adaptativa é mais geral do que a rede reconfigurável porque pode alterar seu comportamento mantendo a mesma configuração modificando as regras de disparo das transições. Em uma rede adaptativa, as regras de operação em caso de falhas podem ser incorporadas na rede padrão, permitindo maior agilidade na operação sem a necessidade de parar o processo até que os especialistas nas falhas assumam o controle. As recomendações dos peritos já seriam incorporadas na rede Petri padrão utilizada na monitorização da operação. Sistemas reconfiguráveis durante a operação são uma tendência na concepção de

sistemas de controle e a capacidade de incorporar procedimentos de áreas relacionadas é uma característica que deve ser perseguida.

Em relação aos trabalhos futuros, a extração ou inserção de conhecimento em redes neurais de muitas camadas ocultas, as “Deep Neural Networks”, é uma área onde as técnicas adaptativas podem contribuir em seu desenvolvimento, principalmente na inserção de regras desenvolvidas por especialistas (HU, MA e LIU, 2016).

Uma área não abordada nesta tese e onde o emprego das redes de Petri coloridas adaptativas pode contribuir de forma significativa é a de controle de dispositivos robóticos. Normalmente, o maior desafio para um controlador autônomo é a capacidade de mapear um ambiente desconhecido como, por exemplo, um terreno nunca trilhado pelo dispositivo. A RPCA auxiliaria no mapeamento deste terreno e na adequação do conjunto de regras pré-estabelecidas para a movimentação deste dispositivo.

A mesma ideia pode ser aplicada a dispositivos mecatrônicos de apoio a pessoas com dificuldades motoras. Um dispositivo auxiliar poderia ser testado por pessoas sadias com o objetivo de fornecer uma base de dados de respostas esperadas e, posteriormente, esta base de dados seria enriquecida de modo incremental pelo uso do dispositivo pelas pessoas deficientes.

REFERÊNCIAS

AALST, W. V. D.; STAHL, C. **Modeling Business Processes -** A Petri Net-Oriented Approach. Cambridge, Massachusetts: The MIT Press, 2011.

AGERWALA, T. A Complete Model for Representing the Coordination for Asychronous Processes. **Hopkins Computer Research Report**, Baltimore, n. 32, July 1974.

AHSON, S. I. Petri Net Models of Fuzzy Neural Networks. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 25, n. 6, p. 926-932, June 1995.

ANDRADE, M. T. C. **Uma contribuição à pesquisa em inteligência computacional**. São Paulo: [s.n.], 2002.

ANDREWS, R.; DIEDERICH, J.; TICKLE, A. B. A survey and critique of techniques for extracting rules from trained artificial neural networks. **Knowledge-Based Systems**, v. 8, No.6, p. 373-389, 1995.

ASAR, A. U.; ZHOU, M. C.; CAUDILL, R. J. **Making Petri Nets Adaptive:** A Critical Review. Networking, Sensing and Control. [S.l.]: [s.n.]. 2005. p. 644-650.

AUGASTA, M. G.; KATHIRVALAVAKUMAR, T. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems. **Neural Processing Letters**, 2011.

BADOUEL, E.; OLIVER, J. Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes. **Workflow Systems**, 1998.

CAMOLESI, A. R. **Proposta de um Gerador de Ambientes para a Modelagem de Aplicações usando Tecnologia Adaptativa**. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, 2007.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Florianópolis: Editora da UFSC, 1997.

CASTRO, A. R. G. **Knowledge Extraction from Artificial Neural Networks:** Application to Transformer Incipient Fault Diagnosis. Thesis (Doctor) Porto, Portugal: Faculdade de Engenharia da Universidade do Porto, 2004.

CAVERSAN, F. L. **Exploração de relações entre técnicas simbólicas e conexionistas da inteligência computacional**. São Paulo: Dissertação (Mestrado) - Escola Politécnica da Universidade de Sâo Paulo, 2006.

CHEN, S. M. Weighted Fuzzy Reasoning Using Weighted Fuzzy Petri Nets. **IEEE Transactions on Knowledge and Data Engineering**, v. 14, n. 2, March/April 2002.

CHEN, S. M.; KE, J. S.; CHANG, J. F. Knowledge representation using fuzzy Petri nets. **IEEE Transactions on Knowledge and Data Engineering**, v. 2, n. 3, p. 311-9, Sep 1990.

CHOROWSKI, J.; ZURADA, J. M. Extracting rules from neural networks as decision diagrams. **IEEE Computational Inteligent Society**, p. 1-12, 2011.

COPPIN, B. **Inteligência artificial**. Rio de Janeiro: LTC, 2010.

CRAVEN, M. **Extracting comprehensible models from trained neural networks**. Madison, WI: Thesis (Ph.D.) - University of Wisconsin, 1996.

DOMINGOS, P. **O Algoritmo Mestre -** Como a busca pelo algoritmo de Machine Learning definitivo recriar a nosso mundo. [S.l.]: Novatec, 2017.

FRITZKE, B. Fast learning with incremental RBF networks. **Neural Processing Letters**, 1994. p. 2-5.

FRITZKE, B. **Automatic construction of radial basis function networks with the growing neural gas model and its relevance for fuzzy logic**. In: 1996 ACM SYMPOSIUM ON COMPUTING. **Proceedings …** [S.l.]: [s.n.]. 1996. p. 624-627.

GALLANT, S. I. Connectionist Expert Systems. **Communications of the ACM**, v. 31, n. 2, p. 152-169, 1988.

GAO, M.; ZHOU, M. C.; TANG, Y. Intelligent Decision Making in Disassembly Process Based on Fuzzy Reasoning Petri Nets. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 34, n. 5, p. 2029-2034, October 2004.

GROSSBERG, S. Nonlinear neural networks: Principles, mechanisms, and architectures. **Neural Networks**, v. 1, p. 17-61, 1988.

GUAN, S. U.; LIM, S. S. Modeling adaptable multimedia and self-modifying protocol execution. **Future Generation Computer Systems**, v. 20, n. 1, p. 123-143, 2004.

GUAN, S. U.; LIU, W. Self-Modifiable Color Petri Nets for Modeling. **IEEE Transaction on Computers**, v. 52, n. 7, p. 920-932, July 2003.

GUIBU, H. I.; NETO, J. J. **Incorporação de novas regras em uma Rede de Petri Colorida Adaptativa**. Memórias WTA2017. São Paulo: [s.n.]. 2017. p. 128-132.

GUIBU, H. I.; NETO, J. J. **Use of Adaptive Coloured Petri Network in Support of Decision-Making**. Computer Science & Information Technology. Geneva, Switzerland: Computer Science Conference Proceedings AIRCC. 2017. p. 17-27.

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

HORIKAWA, S.; FURUHASHI, T.; UCHIKAWA, Y. On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm. **IEEE Transactions on Neural Networks**, v. 3, n. 5, p. 801-806, 1992.

HUNG, C. Knowledge-based Rule Extraction from Self-Organising Maps, 2009.

HUNG, C. **Extracting Rules from Optimal Clusters of Self-Organizing Maps**. In: International Conference on Computer Modeling and Simulation, 2. **Proceedings…** [S.l.]: [s.n.]. 2010.

JANG, J. S. R. ANFIS: Adaptive Network based Fuzzy Inference System. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 23, n. 3, May/June 1993.

JANG, J. S. R.; SUN, C. T. Functional equivalence between radial basis function networks and fuzzy inference systems. **IEEE Transactions on Neural Networks**, v. 4, n. 1, p. 156-159, 1993.

JANG, J. S. R.; SUN, C. T.; MIZUTANI, E. **Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence**. New York: Prentice Hall, 1997.

JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri nets:** Modelling and validation of concurrent systems. Berlin: Springer-Verlag, 2009.

KANG, H. et al. Research on Translation of Index Pi-Net based on Index Pi-Calculus. **Journal of Computers**, v. 8, n. 3, p. 779-786, March 2013.

KASABOV, N. **Evoving fuzzy neural networks - algorithms, applications and biological motivation**. Methodologies for the Conception, Design and Application of Soft Computing. [S.l.]: World Scientific. 1998. p. 271-274.

KHELDOUN, A. et al. **A High Level Net for Modeling and Analysis Reconfigurable Discrete Event Control Systems**. In: IFIP INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND ITS APPLICATIONS. **Proceedings…** [S.l.]: Springer International Publishing. 2015. p. 551-562.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, v. 43, p. 59-69, 1982.

KOHONEN, T. **Self-organizing maps**. Berlin,Heidelberg: Springer, 1995.

LI, X.; LARA-ROSANO, F. Adaptive fuzzy petri nets for dynamic knowledge representation and inference. **Expert Systems with Applications**, v. 19, p. 235-241, 2000.

LITTLE, T. D. C.; GHAFOOR, A. A Synchronization and storage model for multimedia objects. **IEEE J. Selected Areas Communications**, p. 423-427, April 1990.

LIU, H. C. et al. Fuzzy Petri nets for knowledge representation and reasoning: A literature review. **Engineering Applications of Artificial Intelligence**, v. 60, p. 45-56, 2017.

LIU, H. C.; YOU, J. X.; TIAN, G. Determining Truth Degree of Input Places in Fuzzy Petri Nets. **IEEE Transactions on Systems, Man, and Cybernetics:** Systems, v. PP, n. 99, p. 1-7, 2016.

LLORENS, M.; OLIVER, J. Structural and dynamic changes in concurrent systems: Reconfigurable Petri Nets. **IEEE Transactions on Computers**, v. 53, n. 9, p. 1147-1158, 2004.

LOONEY, C. G. Fuzzy Petri Nets for Rule-Based Decisionmaking. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 18, n. 1, p. 178-183, January/February 1988.

MALONE, J. et al. Data mining using rule extraction from Kohonen self-organizing maps. **Neural Computing & Applications**, v. 15, n. 1, p. 9-17, 2006.

MCGARRY, K.; MACINTYRE, J.; ADDISON, D. **Integrating RBF Networks with Domain Knowledge**. International Joint Conference on Neural Networks. Washington, D.C.: [s.n.]. 2001.

MCGARRY, K.; WERMTER, S.; MACINTYRE, J. Knowledge extraction from local function networks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Proceedings …**[S.l.]: Laurence Erlbaum Associates. 2001. p. 765-770.

MINSKY, M. L.; PAPERT, S. A. **Perceptrons:** an introduction to computational geometry. Cambridge, Massachusetts: The MIT Press, 1969.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541-580, 1989.

NAUCK, D.; KURSE, R. Neuro-Fuzzy Systems for Function Approximation. In: International Workshop Fuzzy-Neuro Systems, 4. **Proceedings…**. [S.l.]: [s.n.]. 1997.

NETO, J. J. **Adaptive Rule-Driven Devices - General Formulation and Case Study**. Conference on Implementation and Application of Automata CIAA 2001. Pretoria - South Africa: Springer-Verlag. 2001. p. 234-250.

NEWELL, A.; SIMON, H. A. **Human problem solving**. Englewood Cliffs: Prentice-Hall, 1972.

OKADA, R. S. **Tecnologia Adaptativa Aplicada a Sistemas Híbridos de Apoio à Decisão.** Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, 2013.

PEDRAZZI, T. C. **Um Ambiente de Desenvolvimento Baseado em Tabelas de Decisão Adaptativas**. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, 2007.

PETERSON, J. L. **Petri Net Theory and the Modeling of Systems**. [S.l.]: Prentice Hall, 1981.

POP, E.; HAYWARD, R.; DIEDERICH, J. RULENEG: extracting rules from a trained ANN by stepwise negation. **Technical Report QUT NRC**. [S.l.]. 1994.

RIASCOS, L. A. M.; MIYAGI, P. E. **Fault tolerance in manufacturing systems applying Petri nets**. [S.l.]: VDM Verlag, 2010.

SAATY, T. L. How to make a decision: the Analytic Hierarchy Process. **Interfaces**, v. 24, n. 6, p. 19-43, 1994.

SASAKI, E. M. **Aplicação de Redes de Petri Auto-Modificáveis em Sistemas de Manufatura**. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, 1996.

SETIONO, R. Extracting M-of-N rules from trained neural networks. **IEEE Transactions on Neural Networks**, v. 11, n. 2, p. 512-9, 2000.

SETIONO, R.; LEOW, W. K.; ZURADA, J. M. Extraction of rules from artificial neural networks for nonlinear regression. **IEEE Transactions on Neural Networks**, v. 13, n. 3, p. 564-77, 2002.

SILVA, A. C. M. D. **Extração do Conhecimento em Forma de Regras Difusas a partir de Mapas Auto-Organizáveis de Kohonen -** Aplicação em Diagnóstico de Faltas Incipientes em Transformadores. Tese (Doutorado) - Universidade Federal do Pará, 2013.

SILVA, I. N. D.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para engenharia e ciências aplicadas**. São Paulo: Artliber Editora, 2010.

STANGE, R. I. **Adaptatividade em Aprendizagem de Máquinas:** Conceitos e Estudo de Caso. Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo, 2011.

STANGE, R. L.; NETO, J. J. Learning decision rules using adaptive technologies: a hybrid approach based on sequential covering. In: INTERNATIONAL WORKSHOP ON ADAPTIVE TECHNOLOGY. [S.l.]: **Procedia Computer Science**. 2017. p. 1188-1193.

TABAK, D.; LEVIS, A. H. Petri Net Representation of Decision Models. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 15, n. 6, p. 812-818, November/December 1985.

TAJ, S. M.; KUMARAVEL, A. Survey on Fuzzy Petri Nets for Classification. **Indian Journal of Science and Technology**, v. 8, n. 14, p. 1-8, July 2015.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE Transactions on Systems, Man and Cybernetics**, v. 5, n. 1, p. 116-32, 1985.

TCHEMRA, A. H. **Tabela de decisão adaptativa na tomada de decisão multicritério**. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, 2009.

THRUN, S. B. **Extracting provably correct rules from artificial neural networks**. Institut für Informatik III - University of Bonn. [S.l.]. 1993. (IAI-TR-93-5).

TOWELL, G.; SHAVLIK, J. The extraction of refine rules from knowledge based neural networks. **Machine Learning**, v. 131, p. 71-101, 1993.

ULTSCH, A. **Clustering with SOM:** U*C. In: WORKSHOP ON SELF-ORGANIZING MAPS. **Proceedings…** Paris, France: [s.n.]. 2005. p. 75-82.

ULTSCH, A.; KORUS, D. **Automatic acquisition of symbolic knowledge from subsymbolic neural networks**. In: EUROPEAN CONGRESS ON INTELLIGENT TECHNIQUES AND SOFT COMPUTING, 3. **Proceedings…** Aachen, Germany: [s.n.]. 1995. p. 326-331.

ULTSCH, A.; SIEMON, H. P. **Kohonen's self organizing feature maps for exploratory data analysis**. In: INTERNATIONAL NEURAL NETWORKS CONFERENCE. **Proceedings…** [S.l.]: [s.n.]. 1990.

VALK, R. Self-modifying nets, a natural extension of petri nets. **Lecture Notes in Computation**, v. 62, p. 464-476, 1978.

VINCENT, L.; SOILLE, P. Watersheds in digital space: An efficient algorithm based on immersion simulations. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 6, p. 583-598, 1991.

WONG, K. W. et al. Fuzzy rules extraction using self-organizing neural network and association rules. In: IEEE region International Conference on Electrical and Electronic Technology,10. **Proceedings…** Singapore: [s.n.]. 2001. p. 19-22.

YEUNG, D. S.; TSANG, E. C. C. A Comparative Study on Similarity-based Fuzzy Reasoning Methods. **IEEE Transaction on Systems, Man, and Cybernetics, Part B(Cybernetics)**, v. 27, n. 2, Apr 1997.

ZADEH, L. A. Fuzzy Sets as a Basis for a Theory of Possibility. **Fuzzy Sets Systems**, v. 1, n. 1, p. 3-28, 1978.

ZHANG, J. **Modeling and verification of reconfigurable discrete event control systems**. Thesis (Doctoral Dissertation) [S.l.]: - Xidian University, 2015.

ZHANG, J. et al. R-TNCES: A novel formalism for reconfigurable discrete event control systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 43, n. 4, p. 757-772, 2013.

ZHANG, J. et al. Reconfigurable coordination of distributed discrete event control systems. **IEEE Transactions on Control Systems Technology**, v. 23, n. 1, p. 323-330, 2015.

ANEXO A – LINGUAGEM CPN ML ASSOCIADA AO SIMULADOR CPN TOOLS

O objetivo deste anexo é destacar as características mais importantes desta linguagem para facilitar o entendimento dos diagramas apresentados nesta tese.
Os detalhes desta linguagem podem ser obtidos diretamente online em http://cpntools.org/documentation/start.
Informações igualmente importantes e de forma mais didática, mas com uma sintaxe ligeiramente diferente da atualmente adotada, podem ser encontradas no livro de Jensen (JENSEN, 1996). Neste anexo, apresentamos alguns exemplos de Jensen com a sintaxe atualizada para a nova versão de CPN Tools.

A característica mais importante desta linguagem é o conceito de ficha colorida. Na rede de Petri original e nas primeiras versões que se seguiram, só existia um tipo de ficha e todo o controle de marcação de estados era realizado apenas com a quantidade de fichas disponíveis em cada lugar da rede.

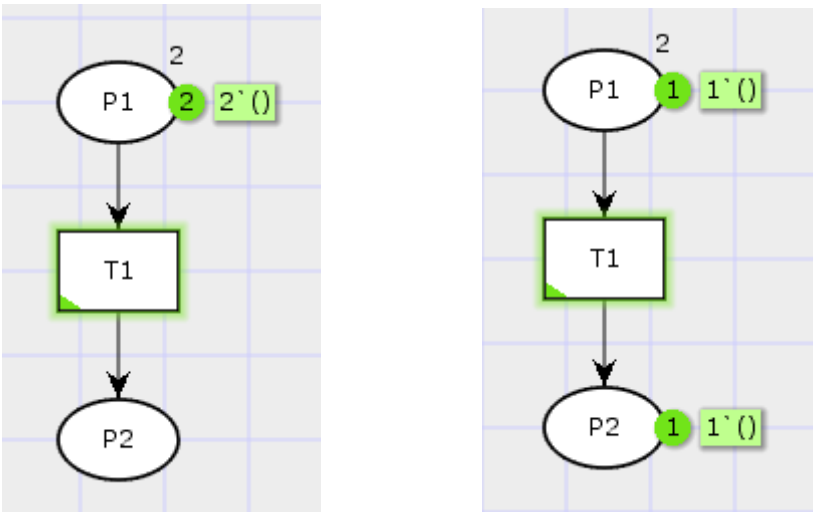
A primeira evolução no conceito de ficha foi o aparecimento das fichas coloridas, ou seja, um lugar passa a ser marcada não apenas pela quantidade das fichas, mas também pelo tipo de ficha como, por exemplo, fichas pretas, fichas vermelhas e fichas verdes.

O grande salto na capacidade de representação de problemas mais complexos ocorreu quando o conceito de ficha colorida foi generalizado, passando a representar tipos de variáveis, aproximando a rede de Petri das linguagens de programação convencionais, mas sem eliminar sua força que é a representação gráfica e a capacidade de análise de situações críticas..

A.1 Conjuntos coloridos (Colour sets - colset)

Na atual versão do CPN Tools, existem os seguintes conjuntos coloridos pré-definidos:

colset UNIT	somente uma cor, representado por (); correponde à ficha original



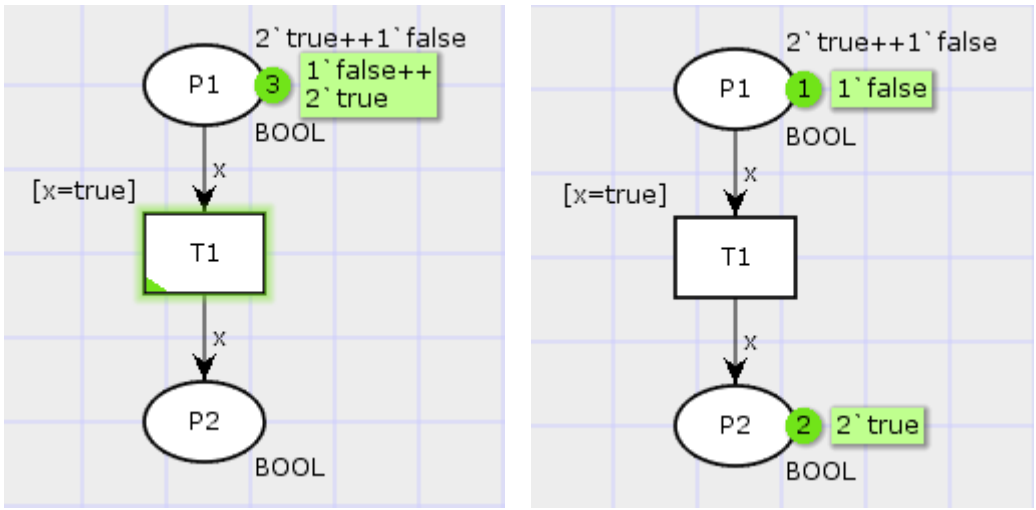
(a) Antes do disparo de T1 (b) Depois do disparo de T1

Figura A-1 Exemplo de conjunto colorido unitário (UNIT)

No exemplo da Figura A-1, cada arco está associado a 1 ficha UNIT, ou seja, cada disparo de T1 vai retirar 1 ficha de P1 e vai depositar 1 ficha em P2.

colset BOOL somente duas cores, verdadeiro e falso

var x: BOOL;



(a) Antes do disparo de T1 (b) Depois de dois disparos de T1

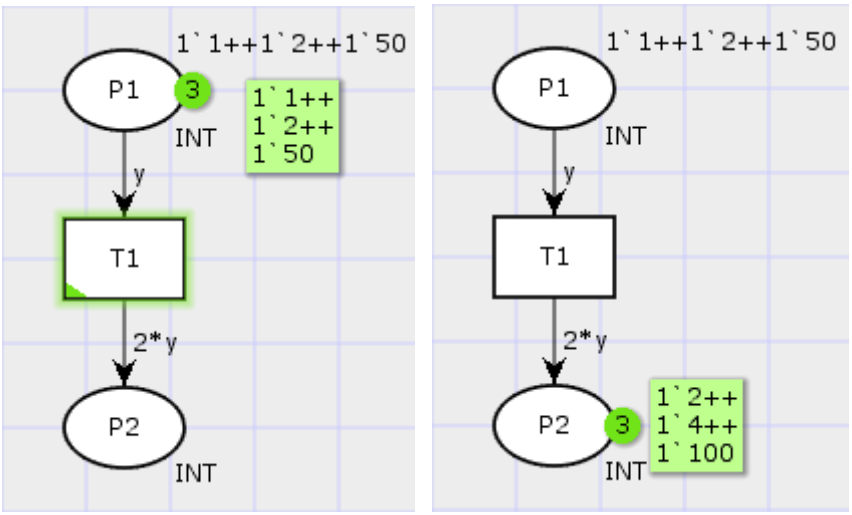
Figura A-2 Exemplo de conjunto colorido booleano (BOOL)

No exemplo da Figura A-2, cada arco está associado à variável x que foi definida como BOOL. Na transição T1 existe uma restrição para seu disparo que é x=true, ou seja, em (a), existem duas fichas true e uma ficha false e por isso T1 está habilitada

a disparar; na situação (b), após dois disparos de T1, no lugar P1 existe apenas uma ficha false, ou seja, T1 não está habilitada a disparar.

colset INT número inteiros

var y:INT;

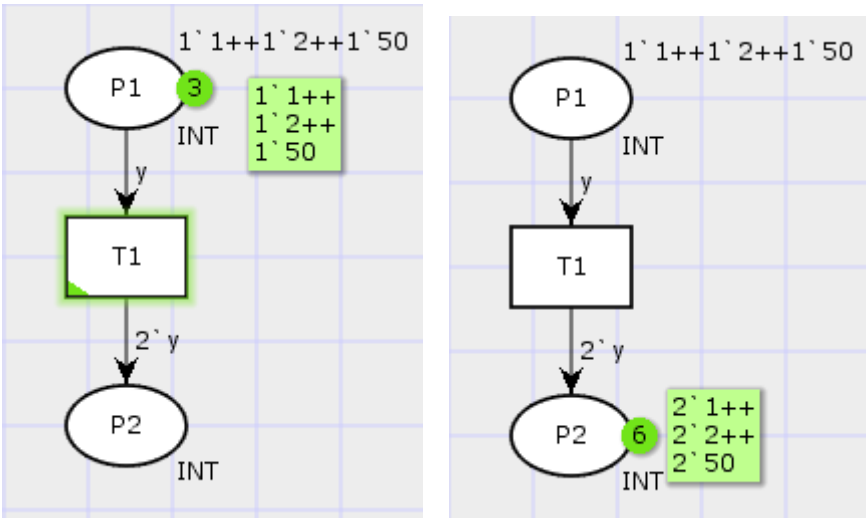


(a) Antes do disparo de T1 (b) Depois de três disparos de T1

Figura A-3 Exemplo de conjunto colorido inteiro (INT)

No exemplo da Figura A-3, antes do disparo de T1 o lugar P1 possui 3 fichas do tipo INT, uma ficha com valor 1, uma ficha com valor 2 e uma ficha com valor 50. O símbolo ++ indica que fichas diferentes serão depositadas neste lugar. O arco de entrada de T1 está associado à variável INT y, ou seja, para T1 ser habilitada, é necessário que P1 contenha fichas do tipo INT. O arco de saída de T1 está associado a 2*y, indicando que o valor da ficha será multiplicado por 2. Após três disparos de T1, três fichas são depositadas no lugar P2, mas cada ficha vale o dobro da ficha original.

No segundo exemplo do conjunto colorido INT ilustrado na Figura A-4, o arco de saída de T1 insere duas fichas no lugar P2, mas não altera o valor das fichas. Após três disparos de T1, o lugar P2 contém seis fichas, duas com valor 1, duas com valor 2 e duas com valor 50, ou seja, dobrou a quantidade de fichas mas não alterou seus valores.

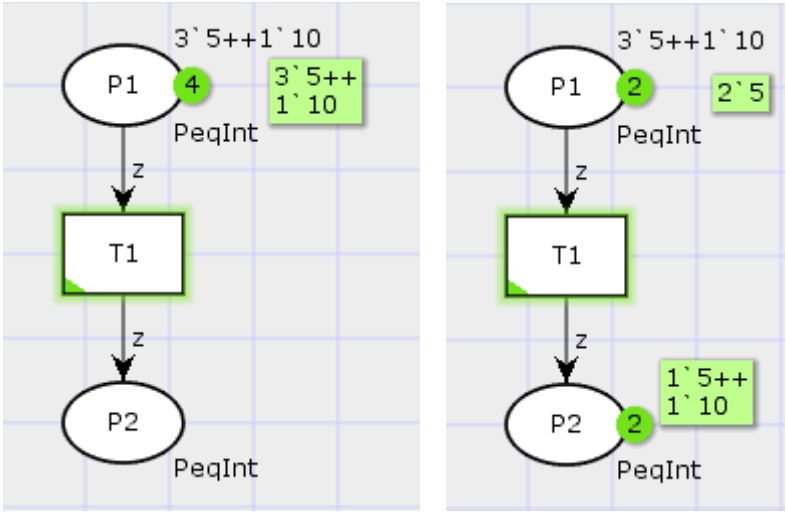


(a) Antes do disparo de T1 (b) Depois de três disparos de T1

Figura A-4 Segundo exemplo de conjunto colorido inteiro (INT)

Na Figura A-5 um novo colset foi definido a partir do colset básico INT e da cláusula with, possibilitando a delimitação da faixa de definição das fichas.

colset int with	números inteiros com limites, especificados pela cláusula “with”
colset PeqInt = int with 5..14;	este novo closet é do tipo INT mas restrito à faixa de 5 a 14 graças à cláusula with
var z:PeqInt;	variável tipo PeqInt



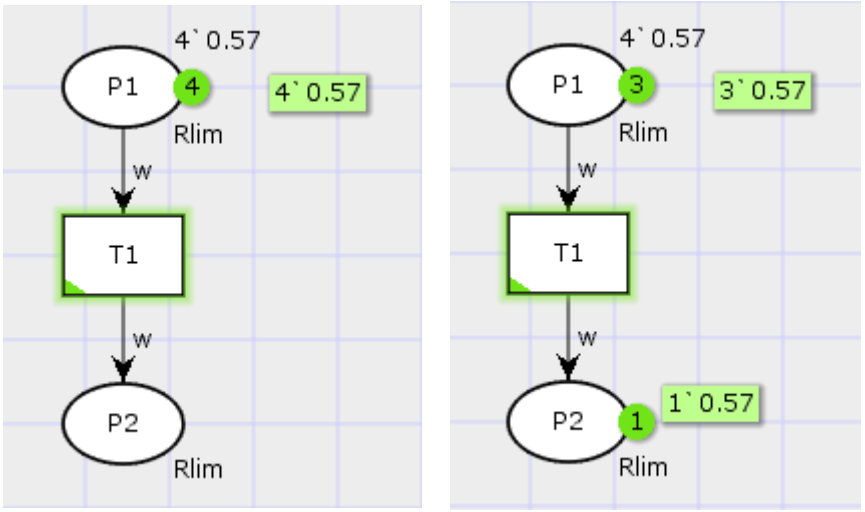
(a) Antes do disparo de T1 (b) Depois de dois disparos de T1

Figura A-5 Definição de um novo conjunto colorido com restrição

colset REAL	define fichas do tipo números reais
colset real with	analogamente ao apresentado para o tipo INT, a cláusula with restrige o campo de variação das fichas e das variáveis;

colset Rlim = real with 0.0 .. 1.0;

var w = Rlim;



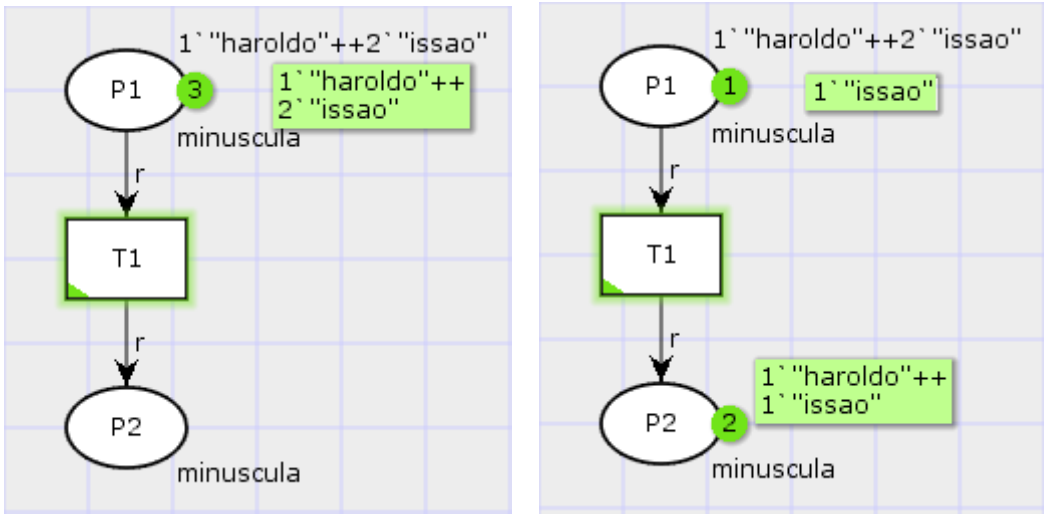
(a) Antes do disparo de T1 (b) Depois de um disparo de T1

Figura A-6 Exemplo de conjunto colorido real com restrição

Na Figura A-6, os dois lugares P1 e P2 e a variável w foram definidos como tipo Rlim, onde Rlim é um tipo REAL limitado ao campo de variação de 0.0 a 1.0. Analogamente ao tipo INT, o que é transmitido é a ficha que, neste exemplo, possui o valor de 0.57.

colset STRING	define fichas do tipo alfanumérico
colset minuscula = string with “a”..”z”;	define fichas limitadas ao intervalo das letras minúsculas

var r: minúscula;



(a) Antes do disparo de T1 (b) Depois de dois disparos de T1

Figura A-7 Exemplo de conjunto colorido alfanumérico com restrição (STRING with)

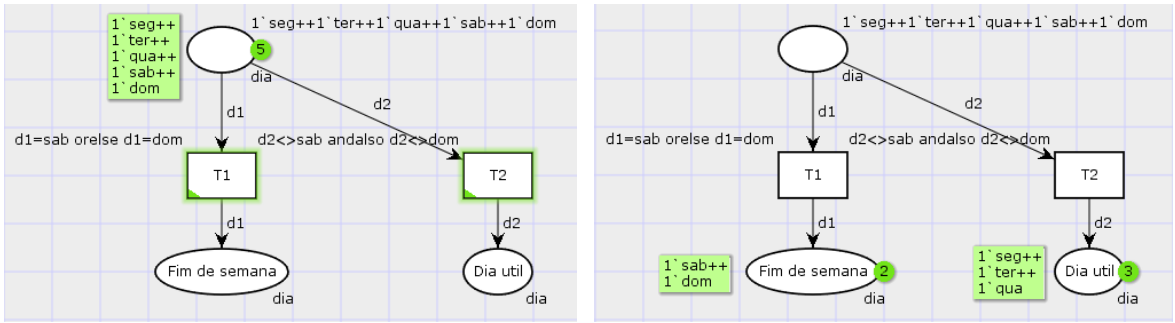
O exemplo da Figura A-7 mostra a definição de fichas do tipo alfanuméricas limitadas pela utilização da cláusula with ao intervalo das letras minúsculas. Cada ficha é constituída de um pacote alfanumérico.

Conjuntos Coloridos Enumerados

Este tipo de conjunto estabelece todos os seus componentes na própria definição.

colset dia = with seg | ter | qua | qui | sex | sab | dom;

var d1,d2 : dia;



(a) Antes dos disparos de T1 ou T2 (b) Depois de dois disparos de T1 e três de T2

Figura A-8 Exemplo de conjunto colorido enumerado

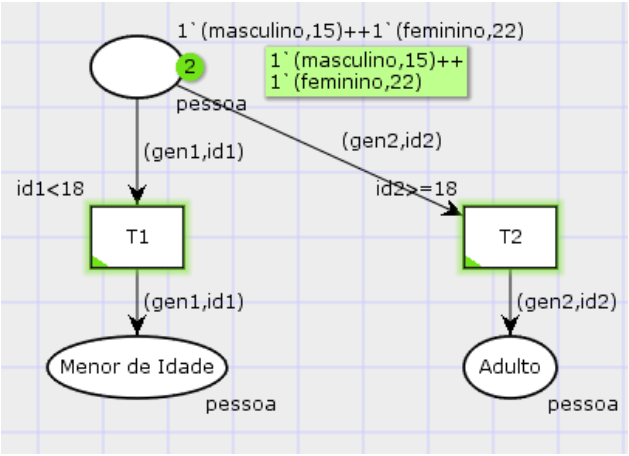
A.1 Conjunto Coloridos Compostos (Compound Colour Sets)

Os conjuntos coloridos compostos são associações de conjuntos coloridos simples e definidos através dos construtores de tipos “product”, “record” e “list”.

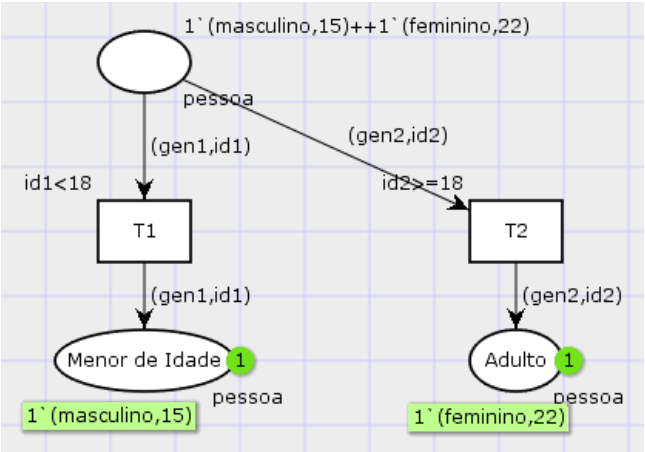
A.1.1 Construtor product

Através deste construtor obtém-se o produto cartesiano de dois ou mais conjuntos coloridos.

colset idade = int with 0..130;	define idade=[0, 130]
colset genero = with masculino | feminino;	define genero
colset pessoa = product genero * idade;	define conjunto colorido composto
var id1, id2 : idade;	variáveis tipo idade
var gen1, gen2 : genero;	variáveis tipo gênero



(a) Antes da transição T1 ou T2



(b) Depois da Transição de T1 e T2

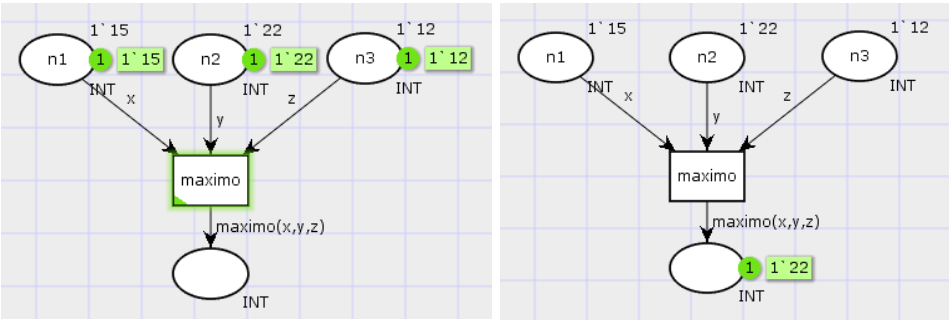
Figura A-9 Exemplo de conjunto colorido composto construído por produto cartesiano

A.2 Funções

Exemplo de definição e utilização de funções.

```
var x,y,z : INT;
fun maximo(a:INT, b:INT, c:INT) =
    if (b>c) andalso (b>a) then b
    else if a>c then a else c;
```

função maximo retorna o maior valor entre a,b,c



(a) Antes do disparo de maximo (b) Depois do disparo de maximo

Figura A-10 Exemplo de função definida pelo usuário

Gerador de números aleatórios com distribuição normal e uniforme

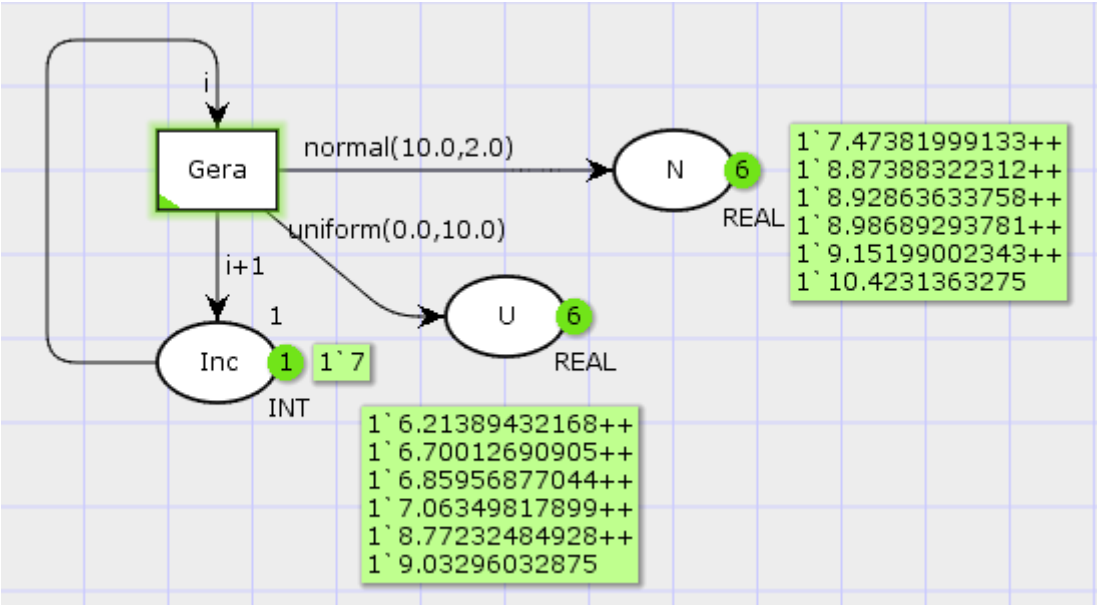


Figura A-11 Exemplo de funções da biblioteca do CPN Tools

A função normal(n:real, v:real) produz um número real aleatório com distribuição normal, com média n e desvio padrão v.

A função uniform(a:real, b:real) produz um número real aleatório com distribuição uniforme e média igual a $(a + b)/2$.

A.3 Estruturas de Controle

A.3.1 Escolha entre várias ações

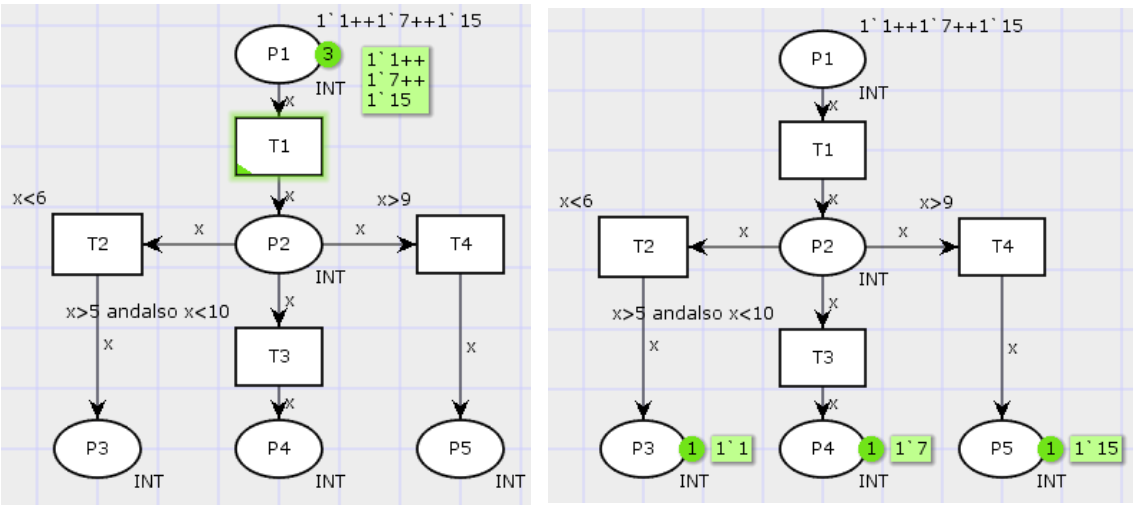


Figura A-12 Escolha entre opções

Na Figura A-12 está esquematizado um esquema de escolha entre três opções. Estas opções dependem do disparo das transições T2, T3 ou T4. Cada uma destas transições possui um parâmetro denominado guarda ou sentinela que só permite a habilitação da transição caso este parâmetro seja verdadeiro. No exemplo da figura, T2 só é habilitada se x<6, T3 só é habilitada se x>5 e x<10 e T4 só é habilitada se x>9. Como estas três condições são mutuamente exclusivas, a rede de Petri

A.3.2 IF THEN ELSE

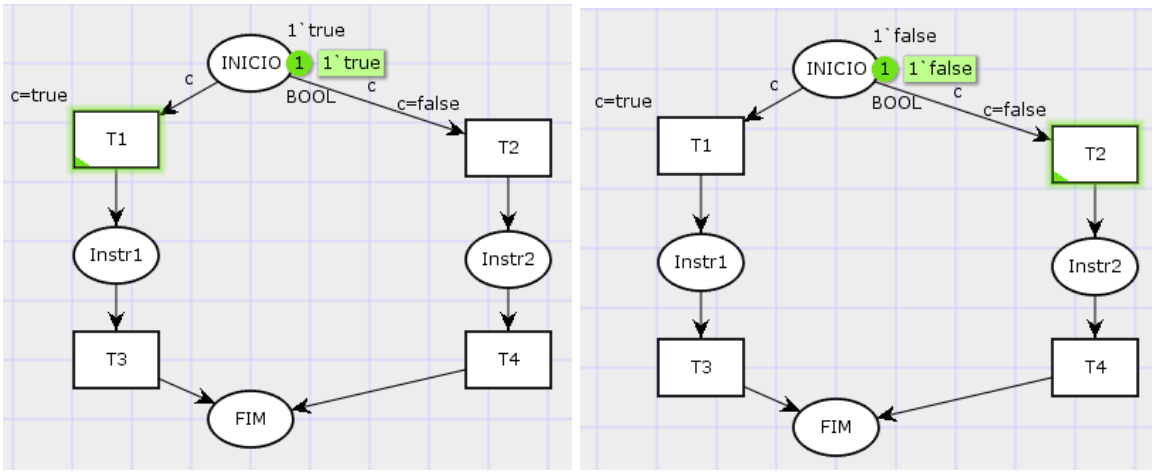


Figura A-13 Exemplo de If Then Else

A.3.3 WHILE DO

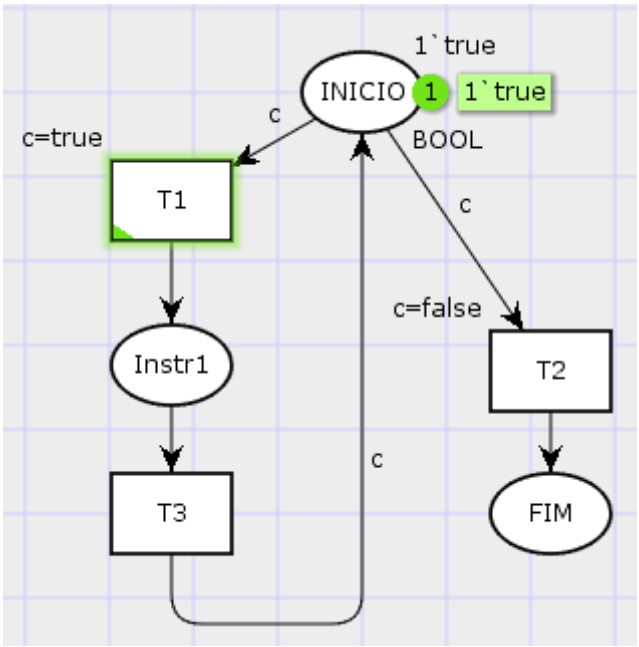


Figura A-14 Exemplo genérico de While Do

O controle de laço por “While Do” está esquematizado na Figura A-14. Enquanto a variável de controle for verdadeira, o conjunto de instruções representado por “Instr1” é executado, voltado em seguida para o teste do laço. Quando este teste resultar “falso”, então o laço é terminado e o programa continua a execução do programa a partir do disparo da transição T2.

A.3.4 REPEAT UNTIL

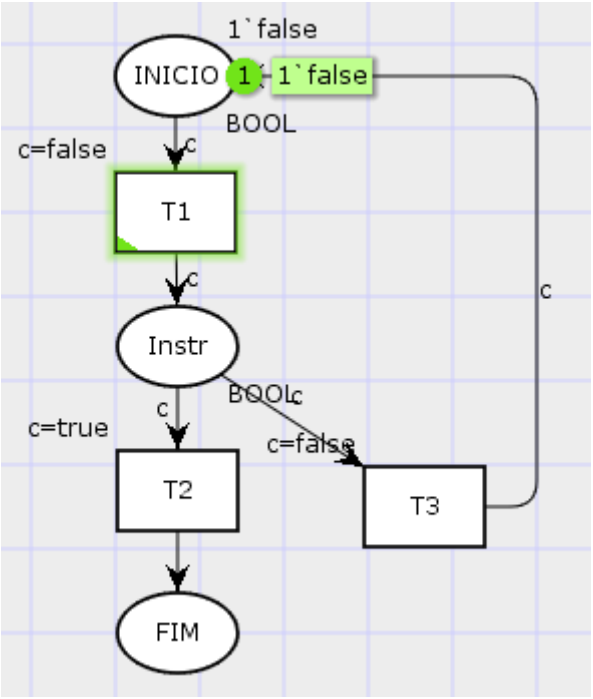
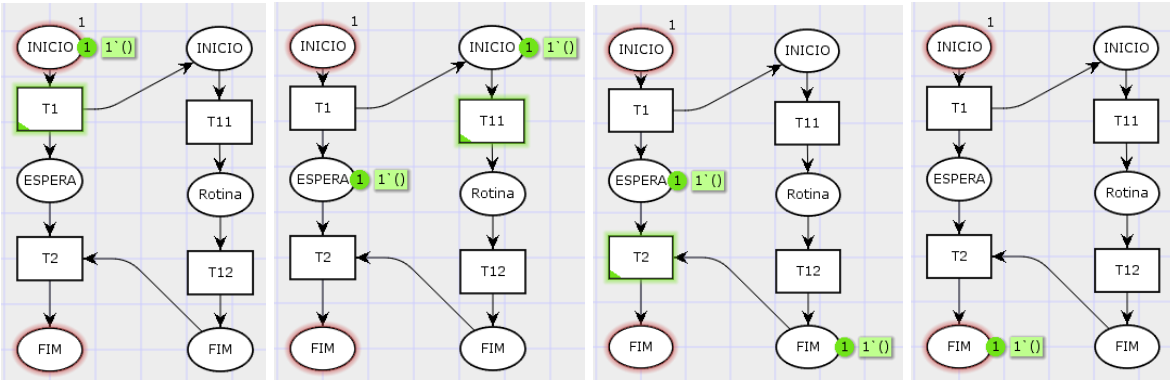


Figura A-15 Exemplo genérico de Repeat Until

No laço “Repeat until”, o conjunto de instruções genericamente representado na Figura A-15 por “Instr” executa uma vez e, caso a variável de controle seja verdadeira, o programa sai do laço e prossegue via disparo da transição T2. Em caso contrário, a transição T3 dispara, obrigando o conjunto “Instr” ser novamente executado, repetindo a execução de “instr” até que a variável de controle se torne verdadeira.

A.3.5 CHAMADA DE ROTINA



(a) (b) (c) (d)

Figura A-16 Exemplo de chamada de rotina

O controle de chamada de rotina é realizado através das transições T1 e T2 da Figura A-16. Em (a), T1 está habilitada e,quando T1 dispara, a rotina é chamada e o programa principal entra em um estado de espera (b) implementado pela transição T2. Quando a rotina termina sua execução (c), a transição T2 fica habilitada e, ao disparar, continua a executar o processo principal (d).