

Apostila  
Redes de Petri, Controle Supervisório e  
Controladores Lógicos Programáveis

Eduard Montgomery Meira Costa, DSc  
UFBA  
Gilberto Bispo da Silva Góes, MSc  
CEFET-BA

©Eduard Montgomery Meira Costa, DSc  
UFBA  
Gilberto Bispo da Silva Góes, MSc  
CEFET-BA, 2003

# Introdução

A preocupação com a produtividade tem sido um dos fatores de propulsão do desenvolvimento tecnológico e da crescente automatização de processos industriais, que tem como objetivo a melhoria qualitativa e quantitativa do que se produz. As inovações técnico-científicas têm possibilitado a criação de equipamentos que não só substituem a força muscular do homem, mas também têm a condição de decidir diante de opções de corrigir seus próprios erros. Assim, a automação invade a rotina industrial e nos coloca a necessidade de conhecer e conviver com ela, retirando dividendos e explorando as vantagens que ela nos oferece [1].

A velocidade das inovações tecnológicas é tão grande, que um sistema instalado hoje, pode se tornar obsoleto amanhã. Por exemplo, a situação atual vivida pela agência espacial americana (NASA), que vem participando de leilões de componentes eletrônicos usados, a fim de manter os sistemas em operação, devido à necessidade de renovação total. Estas inovações são provocadas basicamente pela busca do trinômio quantidade, custo e qualidade, ou seja, produzir em larga escala, a baixo custo e com qualidade. Desta maneira, os sistemas automatizados têm que ser bastante flexíveis, a fim de se adaptarem, por exemplo, a um novo programa (software) de implementação para a mesma célula (hardware), tendo em vista que na maioria das vezes não se pode simplesmente abandonar a planta já instalada, pois significa novos investimentos, o que nem sempre é possível. A flexibilização destes sistemas permite a adaptação rápida destes às exigências de alteração dos produtos, podendo assim atender rapidamente às flutuações do mercado. Como aponta Coriat [2]: *“a flexibilidade favorece a superestimação das taxas de utilização das capacidades instaladas e a aceleração da amortização dos equipamentos, pois permite a fabricação simultânea e de maneira automática de uma gama de peças diferenciadas elaboradas a partir de um produto elementar ou produto de base”*.

Dentre os sistemas automatizados, os sistemas de manufatura vêm sendo atualmente intensamente pesquisados. O principal objetivo consiste na busca de modelos matemáticos adequados para representá-los. Estes sistemas têm em comum a forma pela qual percebem as ocorrências do ambiente. Essas ocorrências são denominadas eventos, que são instantâneas e provocam mudanças de estados no sistema [3]. Após a ocorrência de um evento, o sistema se acomoda em uma nova configuração ou estado,

onde permanecerá até que um novo evento ocorra. São exemplos de eventos o início e o término de uma tarefa e a chegada de um sinal de pressão em um atuador. Sistemas com estas características são chamados de Sistemas a Eventos Discretos (SEDs).

A natureza discreta dos SEDs, em oposição aos sistemas de variáveis contínuas, tratados pela Teoria de Controle Clássica, faz com que os modelos matemáticos convencionais, baseados em equações diferenciais, não sejam adequados para tratá-los. Ilustrativo é comparar a trajetória de um SED, como a apresentada na Figura 0.1, com a trajetória de um Sistema Dinâmico a Variáveis Contínuas (SDVC), apresentada na Figura 0.2.

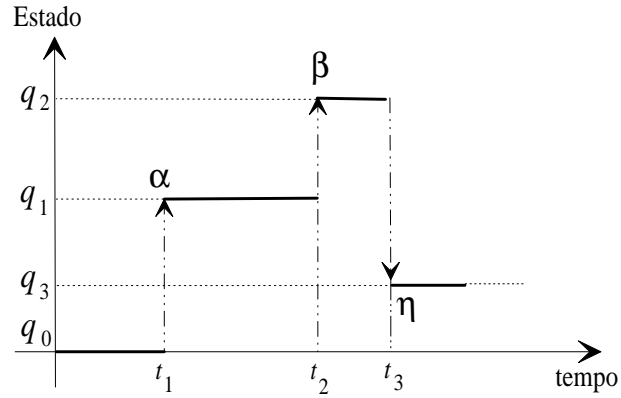


Figura 0.1: Evolução dinâmica de um SED: ocorrências assíncronas de eventos e mudanças instantâneas de estado.

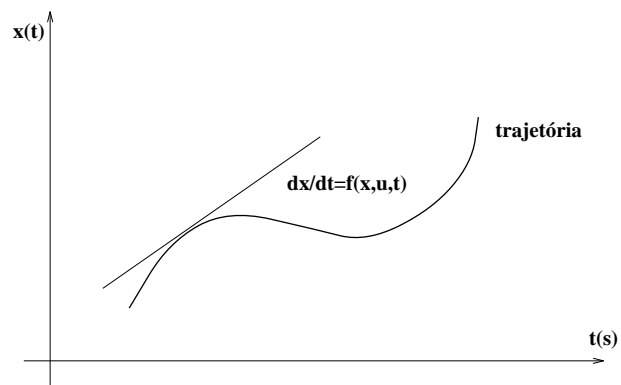


Figura 0.2: Exemplo de trajetória de um SDVC.

Da análise destas figuras, nota-se que o espaço de estados de um SED é limita-

do, enquanto que em um SDVC é infinito. Os SEDs podem permanecer um tempo arbitrariamente longo em um mesmo estado, sendo a sua trajetória descrita por uma seqüência de eventos, enquanto que em um SDVC, em geral, muda de estado a cada instante, e seu comportamento é descrito por uma função que relaciona o estado (variável dependente) ao tempo (variável independente).

Diversos são os exemplos de sistemas de manufatura automatizados que são tratados como SEDs. Um caso particular será apresentado adiante, em que a trajetória apresentada na Figura 0.1 pode caracterizar, por exemplo, os seguintes estados: sistema inicialmente em repouso ( $q_0$ ); atuador linear de dupla ação se movimenta para pegar uma peça na entrada ( $q_1$ ); motor de corrente contínua transportando a peça para o lugar de teste  $L_1(q_2)$ ; peça é testada no lugar  $L_1(q_3)$ ; peça sendo rejeitada no lugar de teste  $L_1(q_4)$  e por fim o último estado representa o retorno do sistema ao estado inicial ( $q_0$ ); por outro lado, como pode ser visto na Figura 0.1, a mudança destes estados se dá pela ocorrência de eventos. Especificamente, no caso do estado “ $q_1$ ”, sua ocorrência está condicionada a habilitação de dois eventos: o braço está posicionado na posição que pode pegar uma peça e a existência de uma peça na entrada. Do ponto de vista dos SDVCs, a trajetória apresentada na Figura 0.2, poderia dizer respeito, por exemplo, a um dos movimentos dos atuadores.

Em automação industrial, a TCS vem sendo utilizada largamente. Diversos paradigmas de modelagem de SEDs são utilizados, como as Redes de Petri (RP) com e sem temporização [4, 5, 6, 7, 8], Redes de Petri Controladas [6], Cadeias de Markov, Teoria das Filas, Processos Semi-Markovianos Generalizados (GSMP) [9], Álgebra de Processos [10], Álgebra Max-Plus [11], Lógica Temporal [12] e Teoria de Linguagens Formais e Autômatos [13], embora nenhum tenha se tornado universalmente aceito para o tratamento geral dos SEDs, em que cada um apresenta diferentes objetivos na análise dos sistemas em estudo.

Dentre estes paradigmas de modelagem, as RP vêm sendo atualmente bastante utilizadas, pois em relação aos outros paradigmas apresentam as seguintes vantagens:

1. Facilidade de modelagem relacionadas com as características de um SED (concorrência, sincronismo e assincronismo, conflito, exclusão mútua, relações de precedência, não determinismo e bloqueio);
2. Ótima visualização de dependência de sistemas;

3. Possibilidade de geração de códigos de controle supervisorio diretamente na representação gráfica;
4. Transmissão visual de informações locais;
5. Testes de propriedades indesejáveis (bloqueio e reinicialização);
6. A análise de desempenho sem simulação é sempre possível para muitos sistemas;
7. Simulação dos eventos discretos a partir do modelo;
8. Visualização do estado atual do sistema para monitoração em tempo real;
9. Abordagem de modelagem do tipo refinamento e do tipo composição modular.

Estas redes, cada vez mais estão vinculadas à Teoria de Controle Supervisorio (TCS), pois facilita o trabalho de pesquisa desejado na TCS, que é a modelagem, controle, simulação e análise de desempenho.

## Formalismos de Controle de SEDs

O estudo dos SEDs utilizando as linguagens formais e os autômatos permitiu a formalização da TCS, onde dado o modelo do sistema e uma especificação funcional, constrói-se um supervisor [14, 15]. O supervisor é o agente que avalia os eventos gerados pelo sistema e define a ação de controle a ser executada para o SED seguir a especificação requerida. O paradigma utilizado no formalismo da TCS não define restrições quanto ao uso de outros. Inclusive, por que as linguagens formais e os autômatos na TCS se aplica ao caso em que a linguagem do sistema possa ser descrita como uma linguagem regular (que pode ser expressa compactamente por uma expressão regular [13]). Além do mais, o tratamento dos SEDs com a TCS se dá ao nível de sistemas não temporizados.

Vários outros paradigmas são utilizados para o tratamento dos SEDs quando não se considera a temporização. Exemplo disto são as redes de Petri [16, 17, 18, 19, 20]. Especificamente, em se tratando de um formalismo de grande interesse para o controle de SEDs não temporizados, encontra-se a aplicação das redes de Petri com função de habilitação de transições (RPFHT) [21] como alternativa à formalização do supervisor,

como visto em Barroso [22]. Nesta abordagem, Barroso determina a construção de uma rede supervisora que se apresenta com a mesma estrutura do modelo do SED. A implementação do modelo é feita utilizando o procedimento de Zhou e DiCesare [23], e através da execução dos algoritmos: AMArA (Algoritmo Modificado da Árvore de Alcançabilidade) e do ACGS (Algoritmo para a Construção da Suprema Sublinguagem Controlável), constróem-se o espaço de estados do SED (árvore de alcançabilidade) e as funções de habilitação das transições para a RPFHT supervisora, respectivamente.

Em se tratando da formalização de controle de SEDs temporizados, o problema de controle necessita de uma abstração a mais, que é definida na inclusão da representação temporal. Uma das alternativas utilizadas para este caso é apresentado por Brandin e Wonham [24], que utiliza a inclusão do evento denominado ‘*tick*’ que é sincronizado ao relógio global do sistema. Esta abordagem utiliza o mesmo formalismo da TCS, embora apresente o problema do aumento de estados e transições do modelo para o procedimento de síntese do supervisor. Este formalismo não se aplica a sistemas que necessitam de sincronização.

Quando o sistema necessita de sincronização, a álgebra de dióides [25] se apresenta como ferramenta de grande aplicabilidade, tanto para o caso dos autômatos temporizados [26, 27] ou em suas representações por meio das matrizes de incidência [28, 29], como para as redes de Petri temporizadas [30, 31]. Entretanto, a formalização do controle de SEDs temporizados utilizando como paradigma de modelagem as redes de Petri temporizadas sem esta álgebra é encontrada em Silva [32], onde é aplicado um formalismo similar ao de Barroso a SEDs temporizados. Nesta abordagem, os SEDs são modelados por redes de Petri *seguras* e o supervisor é formalizado por uma rede de Petri temporal interpretada com função de habilitação de transições (RPTIFHT) que possui a mesma estrutura do modelo. Neste caso, o espaço de estados é construído pelo Algoritmo da Árvore de Alcançabilidade de Classes (AAAC), e o ACGS de Barroso é aplicado sem distinção. Além do mais, uma formalização para transformar a estrutura de funções do supervisor em um programa de CLP é introduzida, em que a linguagem utilizada é a IL (Instruction List), apresentando-se este trabalho como de grande importância na utilização das redes de Petri para o controle de SEDs.

No entanto, em se tratando de sistemas reais, em que as ocorrências do ambiente (ou eventos) são definidas através de ocorrências de eventos externos (sensibilizações

de sensores localizados em lugares estratégicos do sistema) é necessário relacionar as transições da rede que modela o sistema com esses sensores. Neste caso, um outro problema se encontram no controle do sistema, que é o de paralisação, ociosidade parcial ou atrasos na execução geral, tornando-o lento devido à falha de um dos sensores. Em nenhum caso citado anteriormente, o supervisor pode tomar uma decisão com relação a este problema. Este é um problema de necessidade real, que exige uma formulação específica para seu tratamento.

## Redes de Petri

Rede de Petri é uma ferramenta matemática e gráfica, que permite modelar e analisar sistemas que apresentam concorrências em suas evoluções dinâmicas. Nos últimos anos, as redes de Petri se mostraram de grande eficácia no tratamento do problema de controle supervisorio de sistemas a eventos discretos, desde a modelagem, até a síntese do supervisor, apresentando várias vantagens sobre os autômatos. Dentre as abordagens utilizadas para a solução deste problema, encontram-se extensões de redes de Petri variadas, como: redes de petri controladas (RPCtl) [33, 16, 20, 34], redes de Petri coloridas (RPC) [35, 36], redes de Petri com função de habilitação de transições (RPFHT) [21], bem como suas próprias propriedades características como os invariantes de lugar [37, 38, 39, 40].

No contexto desta apostila, são estudadas as redes de Petri básicas, também conhecidas por redes de Petri lugar/transição e a RPFHT utilizada para fundamentar o supervisor para SEDs através da abordagem de Barroso [22].

### Redes de Petri Lugar/Transição

São grafos direcionados, bi-partidos (têm associados dois tipos de nós denominados lugares e transições) e ponderados (cada arco apresenta um peso  $w$  que representa  $w$  arcos em paralelo) com uma marcação inicial (para cada lugar, há uma quantidade de marcas, ou fichas, que definem sua marcação). Nas redes de Petri, os arcos são direcionados sempre de um lugar para uma transição e de uma transição para um lugar. A representação gráfica dos lugares são círculos, e as transições são barras ou retângulos. Uma marcação de um lugar é um número inteiro  $m$  representada por  $m$

círculos pretos internos a este lugar. A marcação completa da rede de Petri é definida por um vetor  $M = [m_1, m_2, m_3, \dots, m_n]^T$  onde  $n$  é o número de lugares da rede, e  $m_i$  o número de fichas no lugar  $i$ .

As redes de Petri podem ser de capacidade infinita, onde cada lugar pode conter um número ilimitado de fichas, ou de capacidade finita, em que há um número de fichas limitado para cada lugar.

**Definição 1** *Uma rede de Petri é uma sextupla  $RP = (P, T, A_r, K, W, M_0)$ , onde:*

- $P = \{p_1, p_2, \dots, p_m\}$  é um conjunto finito de lugares;
- $T = \{t_1, t_2, \dots, t_n\}$  é um conjunto finito de transições;
- $A_r \subseteq (P \times T) \cup (T \times P)$  é um conjunto de arcos;
- $K : P \rightarrow \mathbb{N} \cup \{\infty\}$  é a função de capacidade;
- $W : A_r \rightarrow \mathbb{N}^+$  é a função de ponderação;
- $M_0 : P \rightarrow \mathbb{N}$  é a função de marcação inicial, que satisfaz  $\forall p \in P : M_0(p) \leq K(p)$ .

Quando não é considerada a marcação inicial da rede de Petri, denomina-se esta por estrutura de rede de Petri, a qual é denotada por  $E = (P, T, A_r, W)$ . Dessa forma, convencionou-se abreviar a notação de uma rede de Petri por  $RP = \{E, K, M_0\}$ . Se a capacidade da rede é infinita, então denota-se a rede de Petri por  $RP = \{E, M_0\}$ .

O conjunto de arcos das redes de Petri, isto é,  $A_r$ , pode ser dividido em dois subconjuntos definidos como:

- $I \subseteq P \times T$  e
- $O \subseteq T \times P$ .

Dessa forma, tem-se que  $A_r \subseteq I \cup O$ .

O funcionamento da rede de Petri gera uma seqüência de mudanças nas marcações, o que define a evolução dinâmica. Este funcionamento é determinado através de habilitações de transições em marcações alcançadas, e seus respectivos disparos que seguem algumas regras básicas, apresentadas na definição a seguir.



**Definição 2** A mudança da marcação da rede de Petri segue as seguintes regras de disparo das transições:

1. Uma transição  $t$  é dita habilitada (pronta para disparar) em uma marcação  $M$  se e somente se

$$\begin{aligned} \forall p \in P \text{ que entrada de } t : W(p, t) &\leq M(p) \\ \forall p \in P \text{ que saída de } t : M(p) &\leq K(p) - W(t, p); \end{aligned}$$

2. Uma transição habilitada pode ou não disparar;
3. O disparo de uma transição  $t \in T$ , habilitada na marcação  $M$ , é instantânea e resulta em uma nova marcação  $M'$  da rede de Petri dada pela equação:

$$M'(p) = M(p) - W(p, t) + W(t, p), \forall p \in P; \quad (0.1)$$

4. A ocorrência do disparo de  $t$ , que modifica a marcação  $M$  da rede para uma nova marcação  $M'$ , é denotada por  $M[t\rangle M'$ .

**Exemplo 1** Na Figura 0.3(a) é mostrado um exemplo de rede de Petri cuja marcação inicial é  $M_0 = [2 \ 0 \ 0]^T$ . Na Figura 0.3(b) é apresentada a mesma rede de Petri com sua nova marcação, a qual é alcançada após o disparo da transição  $t$ , de acordo com a Definição 2.

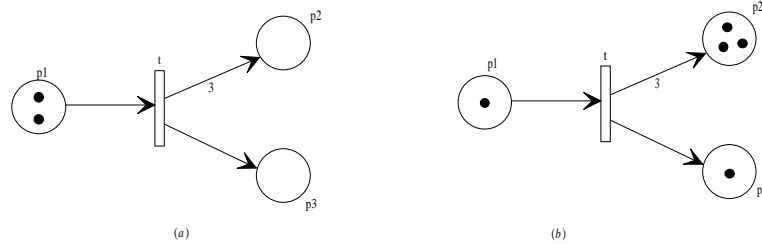


Figura 0.3: Exemplo de uma rede de Petri antes e após o disparo de sua transição.

Denota-se uma transição  $t$  habilitada em uma marcação  $M$ , como  $M[t\rangle$ .

Numa rede de Petri, enquanto há transições habilitadas para cada marcação alcançada, os disparos das transições podem continuar indefinidamente. Isto gera um conjunto de marcações alcançadas pelo funcionamento da rede, o qual é definido como a seguir.

**Definição 3**  $MA(E, M_0)$  é o conjunto de marcações alcançáveis de uma rede de Petri a partir de  $M_0$ , tal que  $M \in MA(E, M_0)$  e, se  $M_1 \in MA(E, M_0)$  é habilitada para uma dada transição  $t \in T$ , e  $M_1 [t] M_2$ , então  $M_2 \in MA(E, M_0)$ .

Este conjunto de marcações alcançáveis é encontrado por meio de seqüências de disparos de transições habilitadas a partir de  $M_0$ . Sendo assim, pode-se mapear todas essas marcações ligadas pelas respectivas transições que mudam de uma marcação para outra, em uma estrutura em árvore. Esta estrutura direcionada de marcações alcançadas é denominada de *Árvore de Alcançabilidade* e permite avaliar o comportamento da rede. A árvore de alcançabilidade de uma rede de Petri é construída pelo algoritmo a seguir:

**Algoritmo 1** *Algoritmo da Árvore de Alcançabilidade*

- *Início*

**Passo 1** *Rotule a marcação inicial  $M_0$  de raiz e etiquete-a com nova;*

**Passo 2** *Enquanto houver marcações nova, faça o seguinte:*

1. *Escolha uma nova marcação  $M$ ;*
2. *Se  $M$  for idêntica a outra marcação no caminho da raiz até  $M$ , etiquete-a como antiga e vá para uma outra marcação;*
3. *Se nenhuma transição estiver habilitada em  $M$ , etiquete-a como bloqueada;*
4. *Enquanto existirem transições habilitadas em  $M$ , para cada transição habilitada, faça o seguinte:*
  - i) *Obtenha a marcação  $M'$  que resulta do disparo da transição  $t$  em  $M$ ;*
  - ii) *Se no caminho da raiz até  $M'$  existir uma marcação  $M''$  tal que  $M'(p) \geq M''(p)$  para cada lugar  $p$  e  $M' \neq M''$ , então substitua  $M'(p)$  por  $\omega$  para cada lugar  $p$  tal que  $M'(p) > M''(p)$ ;*
  - iii) *Introduza  $M'$  como um nó, desenhe um arco com rótulo  $t$ , de  $M$  para  $M'$  e etiquete  $M'$  com nova;*

- *Fim*

**Exemplo 2** Considere a rede de Petri da Figura 0.4. Seguindo o algoritmo da árvore de alcançabilidade, constrói-se a árvore apresentada na Figura 0.5.

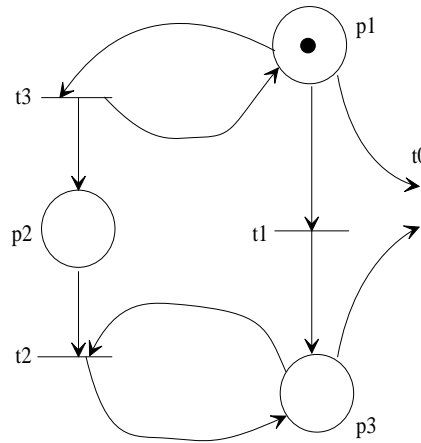


Figura 0.4: Rede de Petri utilizada para a construção da árvore de alcançabilidade.

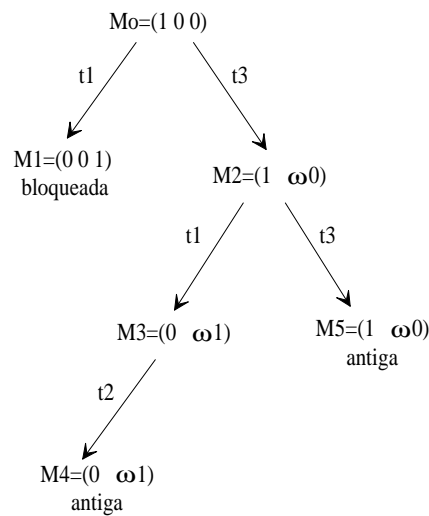


Figura 0.5: Árvore de alcançabilidade da rede de Petri.

A árvore de alcançabilidade de uma rede de Petri pode ser formalizada em termos de um grafo direcionado, denominado de grafo de alcançabilidade. Este difere da árvore de alcançabilidade apenas pela eliminação das marcações *antiga* e o seu consequente direcionamento para estas marcações (etiquetada por *antiga*), bem como a eliminação das etiquetas *bloqueadas*.

**Exemplo 3** Considere a rede de Petri do Exemplo 2. Seu grafo de alcançabilidade está apresentado na Figura 0.6.

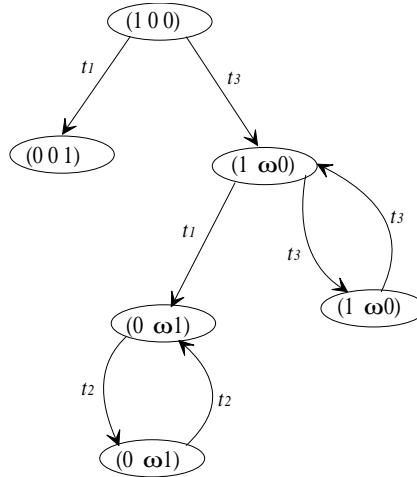


Figura 0.6: Grafo de alcançabilidade da rede de Petri.

Desde que a árvore de alcançabilidade pode ser representada por um grafo, pode-se ver que a evolução dinâmica da rede de Petri que é definida por seqüências de disparos pode ser representada por uma linguagem formal. Isto é feito quando se etiquetam as transições por símbolos de um alfabeto  $\Sigma$ . Esta linguagem é denominada de linguagem de rede de Petri.

### Linguagens das Redes de Petri

A linguagem gerada por uma rede de Petri é definida através dos disparos de transições que são etiquetadas com símbolos de um alfabeto. As transições da rede são etiquetadas com os símbolos de um alfabeto, através da função de etiquetagem  $h$  definida como  $h : T \rightarrow \Sigma$ .

**Exemplo 4** Sejam para a rede de Petri da Figura 0.7 as transições etiquetadas por  $h(t_1) = \alpha$  e  $h(t_2) = \beta$ . Logo, vê-se que a seqüência de disparos das transições desta rede é  $\tau = t_1 t_2 t_1 t_2 \dots$ , a qual através da função de etiquetagem  $h(\tau)$ , transforma-a na linguagem  $h(\tau) = \alpha \beta \alpha \beta \dots = (\alpha \beta)^*$ , que é sua linguagem gerada.

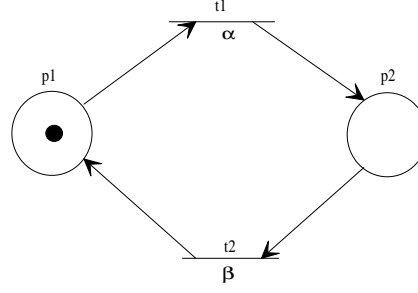


Figura 0.7: Exemplo de uma rede de Petri que gera a linguagem  $(\alpha\beta)^*$ .

## Redes de Petri com Função de Habilitação de Transição

As redes de Petri com função de habilitação de transições são redes de Petri que comportam em suas transições funções que determinam se a transição pode ser habilitada quando satisfaz as condições normais de habilitação (Definição 2). Estas funções são dependentes das marcações de alguns lugares da rede. Com estas funções, é possível determinar condições específicas para controlar a evolução dinâmica da rede de Petri, impedindo que certos ramos da árvore de alcançabilidade sejam alcançados. A definição desta rede é apresentada a seguir:

**Definição 4** *Uma rede de Petri com função de habilitação de transição, é uma quíntupla*

$$RPFHT = (E, l, K, M_0, \Phi),$$

em que:

- $E = (P, T, A_r, W)$  é uma estrutura de rede de Petri;
- $l : T \rightarrow \Sigma$ , é a função que etiqueta as transições, onde  $\Sigma$  é um alfabeto;
- $K : P \rightarrow \mathbb{N} \cup \{\infty\}$  é a função de capacidade;
- $M_0$  é a marcação inicial, como definido para as redes de Petri;
- $\Phi = \{\phi_1, \dots, \phi_m\} : MA(E, M_0) \rightarrow \{0, 1\}$  é a função de habilitação das transições, que mapeia o conjunto de marcações alcançáveis em 0 ou 1.

Com a introdução das funções de habilitação, uma transição  $t_j$  com função de habilitação  $\phi_j$  associada estará desabilitada na marcação  $M_i \in MA(E, M_0)$  quando

$\phi_j(M_i) = 0$ . Caso contrário, se  $\phi_j(M_i) = 1$ , então  $t_j$  estará habilitada e sujeitas às condições definidas nas regras de disparo das transições.

**Definição 5** *O estado, ou marcação, de uma RPFHT varia de acordo com a seguinte regra de disparo das transições:*

1. Uma transição  $t_j$  é dita habilitada (para disparar) em  $M$  se e somente se

$$\begin{aligned} &\forall p \in P \text{ que entrada de } t_j : W(p, t_j) \leq M(p); \\ &\forall p \in P \text{ que saída de } t_j : M(p) \leq K(p) - W(p, t_j); \\ &\phi_j = 1; \end{aligned}$$

2. Uma transição habilitada pode ou não disparar;
3. O disparo de uma transição  $t_j \in T$ , habilitada na marcação  $M$ , é instantânea e resulta em uma nova marcação  $M'$  dada pela equação:

$$M'(p) = M(p) - W(p, t_j) + W(t_j, p), \forall p \in P; \quad (0.2)$$

4. A ocorrência do disparo de  $t_j$ , que modifica a marcação  $M$  da rede para uma nova marcação  $M'$ , é denotado por  $M[t > M']$ .

Os mesmos métodos de análise das redes de Petri L/Tr podem ser usados para as RPFHTs, embora tenham de ser consideradas as condições específicas para a evolução.

**Exemplo 5** *Na Figura 0.8 é mostrado um exemplo de RPFHT, onde a transição  $t$  tem a função de habilitação associada definida por*

$$\phi = [M(p_1) \geq 2 \wedge M(p_3) = 0].$$

*De acordo com a função de habilitação  $\phi$ , a transição  $t$  está habilitada podendo disparar (ver Figura 0.8(a)), desde que as condições necessárias a sua habilitação sejam verdadeiras, tornando  $\phi = 1$ . Com o seu disparo (Figura 0.8(b)), vê-se que a condição para o lugar  $p_3$  ( $M(p_3) \neq 0$ ), o que torna a função  $\phi = 0$  e desabilita a transição  $t$ .*

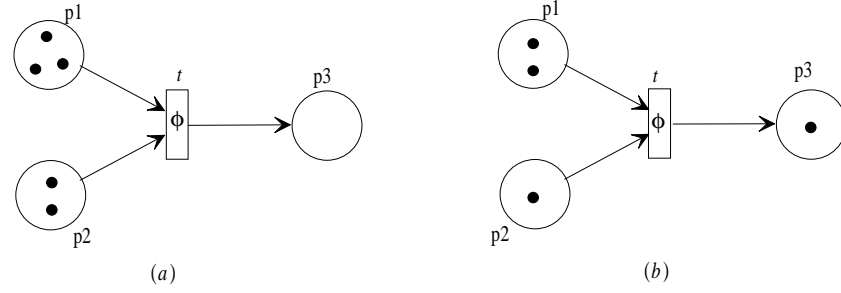


Figura 0.8: Exemplo de uma RPFHT.

## Redes de Petri e TCS

Um dos formalismos que utiliza as redes de Petri para a síntese de supervisores para SEDs é a abordagem via RPFHT descrita por Barroso [22]. Nesta formalização, as redes de Petri são utilizadas para modelar o gerador  $G$  e o supervisor  $S$ . O formalismo de utilização das redes de Petri nesse contexto é visto no diagrama em blocos da Figura 0.9, onde se pode ver que dado o modelo do sistema, sintetiza-se o supervisor baseado em especificações de comportamento a serem realizadas pelo sistema supervisionado. Esta abordagem se torna mais flexível, pois para o SED modelado por uma rede de Petri, qualquer tarefa especificada mantém a estrutura do supervisor. O formalismo de síntese do supervisor está apresentado na Figura 0.10.

Para esta abordagem, utiliza-se da árvore de alcançabilidade da rede que modela o SED como o espaço de estados do sistema. Também, o comportamento desejado pode ser definido através de uma linguagem, desde que as redes de Petri têm uma linguagem associada, da mesma forma que os autômatos. Com estes dados, constrói-se a  $SupC(L)$ , fundamentando funções de habilitação para as transições que controlam a evolução da rede satisfazendo as condições da TCS.

Com a construção da RPFHT supervisora, pode-se gerar a execução síncrona da rede de Petri que modela o sistema juntamente com o supervisor, de acordo com as restrições impostas ao seu comportamento dinâmico, que são determinadas pelas funções de habilitação de transições. Dessa forma, a rede supervisora garante que seja seguida uma sequência de transições requerida, impedindo bloqueios ou comportamentos não permitidos ao sistema, fazendo com que o mesmo realize a tarefa especificada.

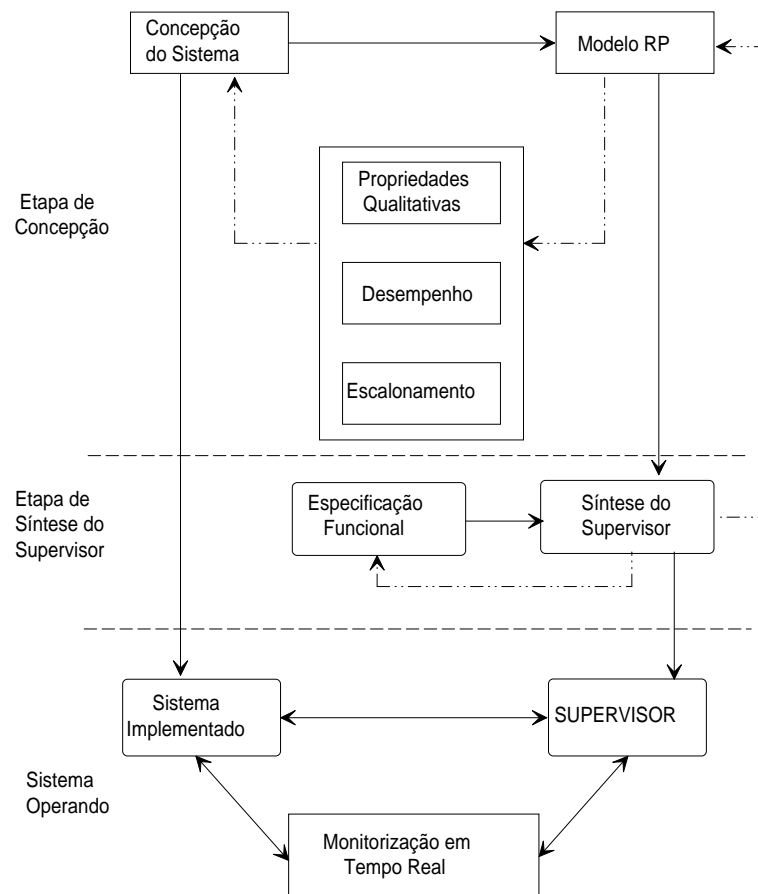


Figura 0.9: Utilização de redes de Petri e TCS para a concepção, análise e controle de um SED.

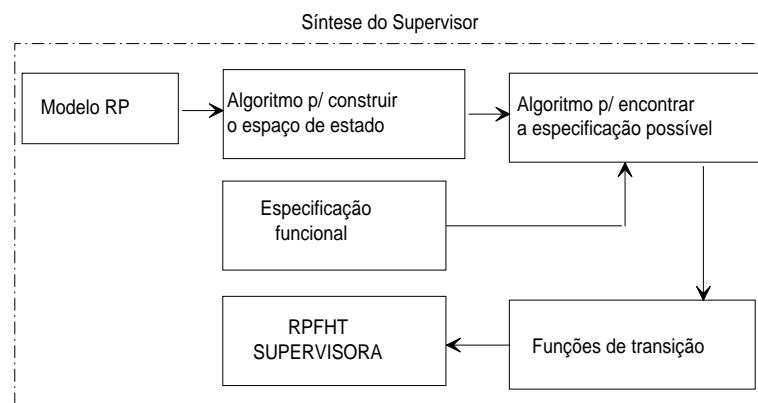


Figura 0.10: Diagrama em blocos do procedimento de síntese do supervisor.



## Definição do Problema de Controle Supervisório

Da TCS, o problema de controle supervisório é dividido em três etapas para sua resolução, que são: a modelagem, a definição da especificação de comportamento e a síntese do supervisor.

### Modelagem de SEDs por Redes de Petri

Para modelar um SED é necessário descrever detalhadamente o seu funcionamento; definir um conjunto finito de estados que o sistema pode alcançar, o qual deve ser suficiente para descrever esse comportamento; as variáveis que se deseja estudar e um conjunto de eventos, que descrevam todas as transições entre os seus estados. Assim, quando utilizando redes de Petri na modelagem de SEDs se considera que cada evento é associado com uma transição da rede, os estados do sistema são definidos por suas marcações alcançáveis e os lugares são suas partes componentes, como visto em [18, 23]. Dessa maneira, tem-se que:

1. Um lugar pode ser interpretado como o estado de um recurso ou de uma atividade. Quando o lugar é interpretado como o estado de um recurso, o número inicial de fichas pode ser constante para representar que há uma quantidade de recursos no sistema, ou variável para representar a quantidade de tarefas realizadas no sistema;
2. Se um lugar é interpretado como o estado de um recurso, a presença de uma ou mais fichas nesse lugar indica que o recurso está disponível e a ausência de fichas indica que o recurso não está disponível. Por outro lado, se o lugar é interpretado como o estado de uma atividade, a presença da ficha indica que essa atividade está sendo realizada e a ausência da ficha indica que a atividade não está sendo realizada;
3. Uma transição pode ser interpretada tanto como o início quanto como o término de um processo ou de uma atividade;

Com essa formalização, constrói-se um modelo de um SED em rede de Petri de acordo com os seguintes passos:

1. Identificar os recursos e atividades necessários ao funcionamento do SED;
2. Criar uma lista ordenada de atividades de acordo com as relações de precedência definidas da descrição textual do funcionamento do SED;
3. Para cada atividade da lista:
  - (a) Criar e etiquetar um lugar para representar a condição da atividade;
  - (b) Criar uma transição para representar o início da atividade com arcos direcionados para os lugares de saída;
  - (c) Criar uma transição para representar o término da atividade com arcos direcionados para os lugares de entrada. De modo geral, a transição de término de uma atividade será a mesma transição de início da próxima atividade na lista ordenada. Quando a rede for executada, uma ficha num lugar significa que a atividade está sendo executada e várias fichas indicarão sua execução na multiplicidade do número de fichas. O disparo de uma transição de inicialização representa o início do processo e o disparo de uma transição de finalização representa a complementação da atividade e pode também representar o início da próxima atividade;
4. Para cada atividade ordenada: se um determinado lugar não tiver ainda sido criado, criar e etiquetar o lugar para cada recurso que deve estar disponível para iniciar a atividade. Conectar todos os lugares de disponibilidade de recursos apropriados com arcos a cada transição de entrada para a inicialização da atividade. Criar arcos de saída para conectar às transições de finalização seguintes à atividade para algum lugar de recurso representando recursos que se tornem disponíveis (estão livres) na complementação da próxima atividade;
5. Especificar a marcação inicial para o sistema.

**Exemplo 6** Na Figura 0.11 está apresentado um simples sistema de manufatura consistindo de duas estações de processamento com máquinas,  $M_1$  e  $M_2$ , um robô compartilhado,  $R$ , para descarga e um buffer,  $B$ , para armazenamento temporário de peças intermediárias. Cada peça é processada primeiro em  $M_1$  e depois em  $M_2$ . As peças entram no sistema e na estação de processamento são automaticamente fixadas a uma

bandeja e carregadas na máquina  $M_1$ . Após o processamento, o robô  $R$  descarrega de  $M_1$  a peça intermediária e a coloca no buffer  $B$ . Logo a seguir, as peças intermediárias são automaticamente carregadas em  $M_2$  e processadas. Quando  $M_2$  encerra o processamento de uma peça,  $R$  descarrega o produto final e libera a bandeja para a primeira estação de trabalho. Assume-se que peças de entrada estão sempre disponíveis para serem processadas e que o produto final é sempre removido. Seguindo a metodologia de construção do modelo do sistema, tem-se no primeiro passo que as atividades requeridas são: estações de processamento (fixação à bandeja, carga e processamento de peças), armazenamento e descarga. Os recursos são  $M_1$ ,  $M_2$ ,  $R$ ,  $B$ , fixadores e peças. No segundo passo, identifica-se a ordem das atividades como:

- $M_1P$  :  $M_1$  carrega, fixa e processa a peça;
- $RU_1$  :  $R$  descarrega uma peça intermediária no buffer;
- $BS$  :  $B$  armazena uma peça intermediária;
- $M_2P$  :  $M_2$  carrega e processa uma peça intermediária;
- $RU_2$  :  $R$  descarrega o produto final de  $M_2$ , libera a bandeja e retorna primeira estação de trabalho.

Seguindo o terceiro passo, é criada a rede mostrada na Figura 0.12(a). No quarto passo, tem-se que a atividade  $M_1P$  requer uma bandeja (representada pelo lugar  $PA$ ) e a máquina  $M_1$  livre (representada pelo lugar  $M_1l$ ). Continuando no quarto passo, criam-se os lugares  $BA$  (buffer livre),  $RA$  (representando o robô livre) e  $M_2l$  (representando a máquina  $M_2$  livre). Os arcos são ligados às transições, como mostrado na Figura 0.12(b), satisfazendo os requerimentos do sistema. Por fim, coloca-se a marcação inicial: inicialmente, ambas as máquinas estão livres (uma ficha em  $M_1l$  e uma ficha em  $M_2l$ ), há quatro bandejas disponíveis (quatro fichas no lugar  $PA$ ) o robô está livre (uma ficha em  $RA$ ) e há espaço livre no buffer para duas peças intermediárias (duas fichas no lugar  $BA$ ). Assim, constrói-se a rede de Petri que modela o sistema de manufatura definido, como visto na Figura 0.13.

## Especificação de Comportamento

A especificação de comportamento pode ser formalizada utilizando as linguagens formais, porque a árvore de alcançabilidade de uma rede de Petri formaliza um grafo que

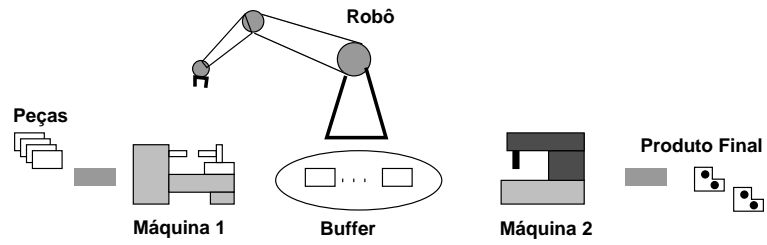


Figura 0.11: Simple sistema de manufatura com recursos compartilhados.

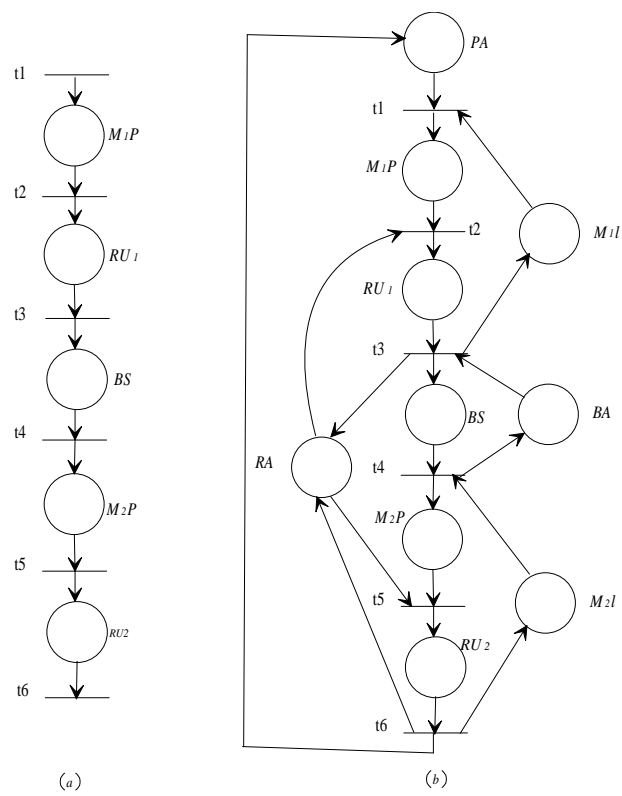


Figura 0.12: Criação da estrutura da rede de Petri que modela o SED.

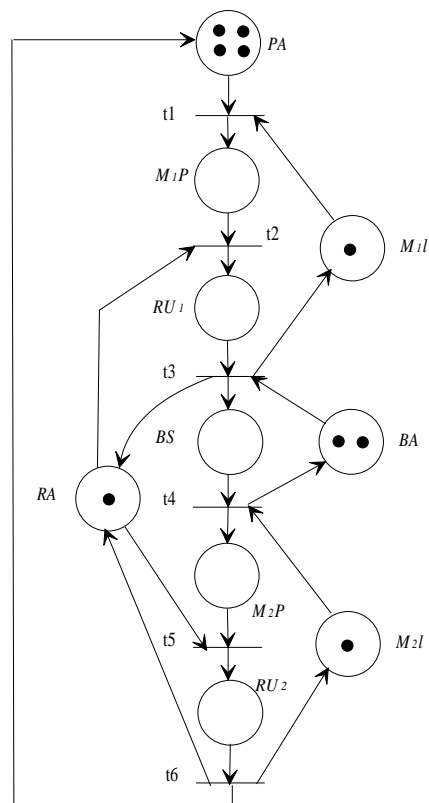


Figura 0.13: Rede de Petri que modela o sistema de manufatura com recursos compartilhados.

é uma estrutura de um autômato. Neste caso, os estados marcados são representados por marcações alcançáveis da rede que são desejadas e que representam tarefas completadas pelo SED.

**Exemplo 7** Considere o sistema de transmissão e recepção de dados, que está apresentado na Figura 0.14, modelado por uma rede de Petri, que tem sua árvore de alcançabilidade apresentada na Figura 0.15. Neste sistema, o transmissor está sempre pronto para enviar uma mensagem que deve passar por um buffer antes de ser transmitida pelo canal, e o receptor deve enviar uma mensagem de retorno, confirmando o recebimento da mensagem. Neste sistema, uma nova mensagem só pode ser enviada, se houver o aviso de recepção. Formula-se, então, a especificação de comportamento definindo-a em termos de uma linguagem marcada. Assim, deseja-se que este sistema esteja sempre preparando novas mensagens para serem enviadas ao receptor, indistintamente de sua transmissão. Isto implica na linguagem  $L_m = (t_1^* + t_1^*t_2t_3t_4)^*$ . Se é desejado que esta mensagem só seja preparada se o sistema receber uma mensagem de retorno, determina-se que após  $t_1$  disparar, ela só deve disparar novamente se  $t_4$  disparar, isto é,  $L_m = (t_1t_2t_3t_4)^*$ .

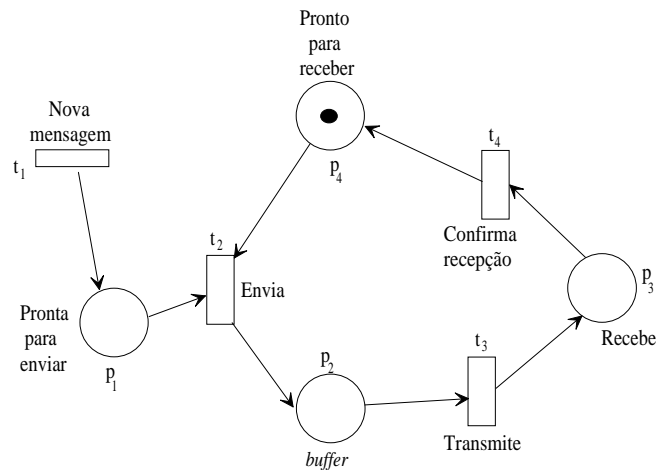


Figura 0.14: Sistema de Transmissão/Recepção modelado por rede de Petri.

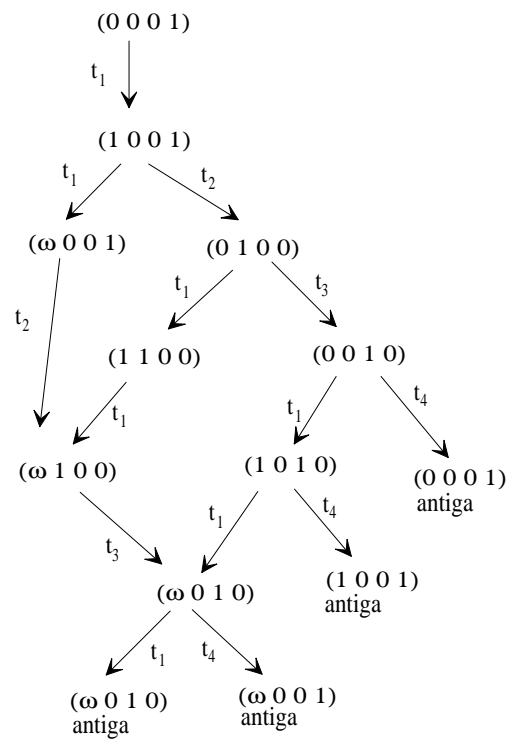


Figura 0.15: Árvore de alcançabilidade da rede de Petri, para o sistema de Transmissão/Recepção.

## Síntese do Supervisor por RPFHT

A abordagem da síntese de supervisores de SEDs usando as redes de Petri necessita dos algoritmos fundamentados em Barroso [22]:

1. AMArA que cria a árvore de alcançabilidade da rede de Petri que apresenta capacidade finita, incluindo um passo a mais na estrutura para caminhos que apresentam o símbolo  $\omega$  que é utilizada pelo ACGS para construir as funções de habilitação de transições e
2. ACGS que, a partir da árvore construída pelo AMArA juntamente com a especificação dada, cria as funções das transições do supervisor.

A seguir são vistos estes algoritmos.

### Algoritmo Modificado da Árvore de Alcançabilidade

#### Algoritmo 2 AMArA

- *Início*

1. Rotule a marcação inicial  $M_0$  como raiz e etiquete-a como *nova*;
2. Enquanto existirem marcações *nova* faça:
  - a) Selecione uma marcação *nova*  $M$ ;
  - b) Se  $M$  for idêntica a uma outra marcação já existente no caminho da raiz até  $M$ , etiquete-a como *antiga*;
  - c) Se nenhuma transição está habilitada em  $M$ , etiquete  $M$  como *bloqueada*;
  - d) Enquanto existirem transições habilitadas em  $M$ , faça o seguinte para cada transição habilitada:
    - i. Obtenha a marcação  $M'$  que resulta do disparo de  $t$  em  $M$ ;
    - ii. Se a capacidade de algum lugar  $p$  é excedida na marcação  $M'$ , então substitua  $M'(p)$  por  $\omega$ ;



- iii. Introduza  $M'$  como um nó da árvore, ligue um arco, com rótulo  $t$ , de  $M$  para  $M'$  e etiqüete  $M'$  como *não-permitida* se a capacidade de algum lugar foi excedida, de outra forma, etiqüete-a como *nova*.

• *Fim.*

**Exemplo 8** O uso deste algoritmo está ilustrado para o caso do sistema apresentado na Figura 0.16, cujo modelo em rede de Petri é de um sistema produtor/consumidor. Neste caso, a utilização deste algoritmo, resulta na árvore de alcançabilidade apresentada na Figura 0.17, onde os estados mostrados nos vetores, são relativos ao vetor de marcação

$$M = [p_1 \ p_2 \ p_3 \ p_4 \ p_5]^T.$$

É observado que, devido à seqüência  $(\alpha_1\beta_1)^*$ , o sistema atinge uma marcação não-permitida, pois o lugar  $p_5$  irá ter um aumento descontrolado no número de fichas. Esta seqüência deve ser evitada pela ação de controle externo ao sistema.

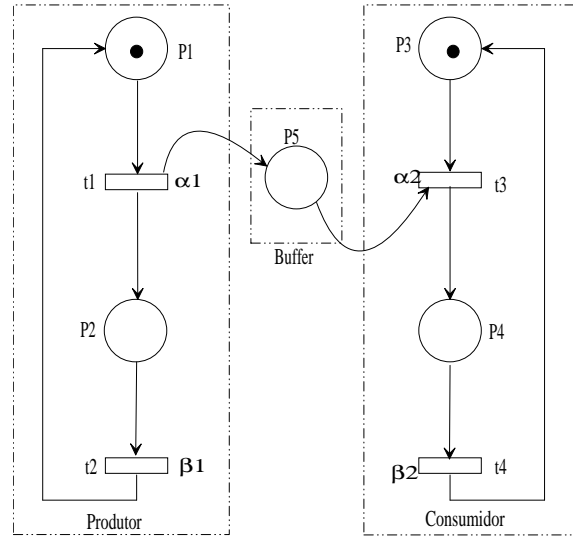


Figura 0.16: Modelo do sistema produtor/consumidor via rede de Petri.

Algoritmo para a Construção do Gerador da  $SupC(L)$

**Algoritmo 3** ACGS

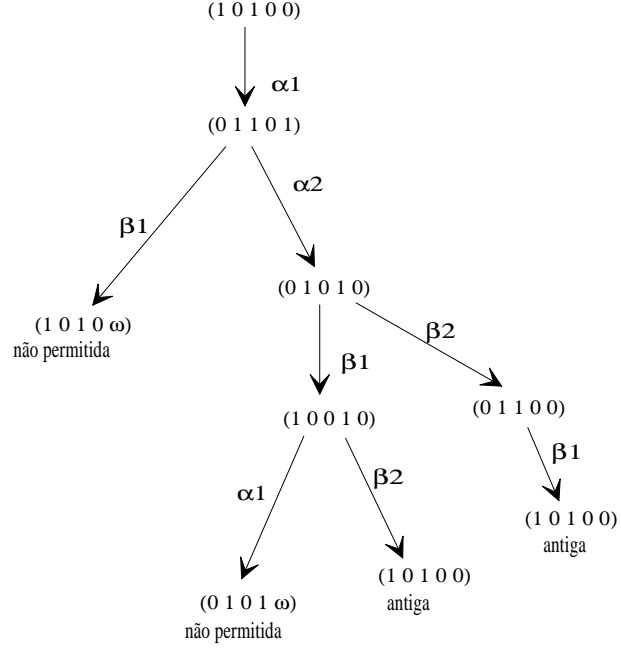


Figura 0.17: Árvore de alcançabilidade do sistema produtor/consumidor modelado por rede de Petri, com utilização do AMArA.

- *Início*

1. Criar uma lista dinâmica, *lista-bloc*, e incluir na mesma os estados ou marcações bloqueadas, incluindo as marcações do tipo *não-permitida*, onde:

$$M : M[t_j > 0, \text{ uma marca o bloqueada e}$$

$$M : M(p_i) = w \text{ uma marca o n o-permitida;}$$

2. Adicionar à *lista-bloc* os estados, não marcados, cuja única transição habilitada, se disparada, leva o sistema a um estado bloqueado, ou seja:

$$M : M[t_j > M', t_j \text{ nica transi o habilitada em } M,$$

$$M \notin Q_m \text{ e } M' \in \textit{lista-bloc};$$

3. Adicionar à lista os estados nos quais exista pelo menos uma transição habilitada, etiquetada por um evento não controlável, cujo disparo da transição leve o sistema para uma marcação na *lista-bloc*, ou seja

$$\exists \alpha \in \Sigma_u, l(t_j) = \alpha \text{ e } M[t_j > M', M' \in \textit{lista-bloc};$$

4. Criar uma lista, *lista-perigo*, com os estados antecessores dos elementos (estados) da *lista-bloc*, juntamente com o evento que os liga, desde que o antecessor não esteja na *lista-bloc*. Estes eventos deverão estar sempre desabilitados quando o sistema se encontrar nesses estados, ou seja:

$$\exists \beta \in \Sigma_c | l(t_j) = \beta \text{ e } M[t_j > M', \text{ e } M \notin \textit{lista-bloc} \text{ e } M' \in \textit{lista-bloc};$$

5. Dada a especificação desejada para o sistema, encontre a suprema linguagem controlável -  $\text{SupC}(L)$ ;
6. Adicionar à *lista-perigo* os estados e seus respectivos eventos de saída a serem desabilitados para que a linguagem executada, desde que estes estados não estejam ainda na *lista-perigo*, ou seja:

$$\exists \beta \in \Sigma_c | l(t_j) = \beta \text{ e } M[t_j > M', \text{ e } M \notin \textit{lista-perigo} \text{ e } M' \notin G(\text{Sup } C(L)).$$

- *Fim.*

O passo 5 deste algoritmo se processa da seguinte forma:

**Algoritmo 4** Passo 5 do ACGS

- Dados o gerador  $G$  trim e o gerador  $H$  (especificação), faça:

1. Adicione à *lista-bloc* os estados que não satisfazem

$$\Sigma(g(x)) \cap \Sigma_{uc} \subseteq \Sigma(x)$$

em que  $g(x)$  representa os eventos habilitados no estado  $x$  do gerador  $G$ ;

2. Para cada estado  $x_i$ , na *lista-bloc*, adicione à *lista-bloc*:

- (a) os estados

$$x_j : (\exists \sigma_u \in \Sigma) x_i = \xi(\sigma_u, x_j);$$

- (b) os estados

$$x_k : (\exists \sigma_c \in \Sigma_c) x_i = \xi(\sigma_c, x_k) \wedge x_k \notin X_m \wedge |\Sigma(x_k)| = 1;$$

3. *Encontre a componente acessível do gerador resultante.*

Deve-se observar que o **passo 1** adiciona à *lista-bloc* todos os estados em que um evento não controlável que é fisicamente possível de ocorrer nele, não esteja definido na especificação. O **passo 2** incrementa a lista e preserva a coacessibilidade. O **passo 3** remove os estados inacessíveis deixados pelos passos anteriores.

**Exemplo 9** Na rede da Figura 0.16, o ACGS determina as funções de habilitação de transições que impedem que o sistema alcance os estados não permitidos ou de bloqueio. De acordo com uma especificação do tipo  $\alpha_1\beta_1\alpha_2\beta_2$ , estas funções seriam  $\phi_1 = \overline{p_5}$ ,  $\phi_2 = \phi_3 = \phi_4 = 1$ . Nessa condição, a função de habilitação da transição  $t_1$  determina que ela só pode disparar se não houver fichas no lugar  $p_5$ , isto é,  $\phi_1 = 1$  se  $M(p_5) = 0$ .

## Exemplo da Síntese do Supervisor de SEDs Modelados por Redes de Petri

Considere o sistema de manufatura com recursos compartilhados modelado por rede de Petri, apresentado na Figura 0.18, cuja marcação inicial seja  $M_0 = [6 \ 0 \ 0 \ 0 \ 0 \ 0 \ 8]^T$ . Este sistema representa a produção de itens utilizando recursos, os quais são representados pela marcação no lugar  $p_7$ . Deseja-se nele determinar inicialmente sua árvore de alcançabilidade, para em seguida ser definida uma especificação de comportamento e determinar as funções de habilitação das transições que executem a mesma, impedindo o sistema de cair num estado indesejável, como é o caso da seqüência de disparos das transições dada por

$$t_1 t_2 t_1 t_2 t_1 t_1 t_3 t_3 t_2 t_3 t_1,$$

que leva a rede a atingir a marcação  $M' = [1 \ 2 \ 0 \ 3 \ 0 \ 0 \ 0]^T$  em que nenhuma transição está habilitada.

A utilização do AMArA gera a árvore de alcançabilidade deste sistema, a qual guarda todas as seqüências de transições possíveis e todos os estados alcançáveis pelo mesmo, determinando para cada seqüência, qual delas leva a um estado bloqueado ou não. Esta árvore contém 162 estados e não é apresentada graficamente devido ao seu tamanho.

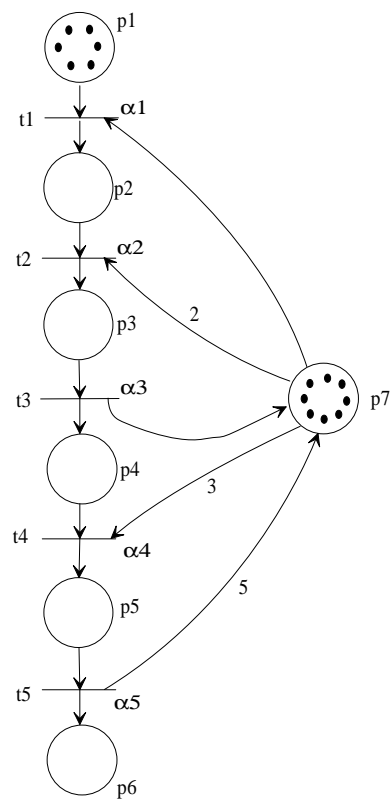


Figura 0.18: Modelo em rede de Petri para um sistema de manufatura com recursos compartilhados.

Definindo uma especificação de comportamento, de forma a ser feito o processamento paralelo de mais de uma peça por vez, tem-se

$$L_m = (t_1^2 t_2 (t_3 + t_2 t_3) (t_4 t_5)^2)^*$$

Com esta especificação, a execução do ACGS determina a seguinte função de habilitação de transição

$$\phi_1 = \{M(p_7) > 6\},$$

que controlará o sistema para impedir seus possíveis bloqueios e seguir a especificação dada. A RPFHT supervisora para esta especificação é vista na Figura 0.19. Observe que a especificação é uma sublinguagem da linguagem da rede de Petri, pois após o disparo de  $t_4$  pela primeira vez, não há mais recursos suficientes para repetir seu disparo, só sendo possível esta sequência.

## Transformação de Redes de Petri em Diagrama Ladder

Na concepção de controladores para SEDs, tanto os computadores como os CLPs têm sido utilizados. No caso dos computadores, a visualização do modelo (seja em autômato, seja em rede de Petri) e sua dinâmica são viáveis, inclusive na visualização de supervisão em tempo real. No caso dos CLPs, apenas a estrutura do supervisor é incluída em forma de código de programa, não havendo visualização do modelo.

A utilização de CLPs na execução de controle de sistemas é muito comum nas indústrias, e com os estudos e formalismos de controle de SEDs, muitos trabalhos têm sido desenvolvidos para a geração de código de CLP que estruture o supervisor destes. Exemplo disto, encontra-se na transformação de redes de Petri em código de programa de CLP utilizando a *IL* (instruction list), como em Silva [32], utilizando o diagrama Ladder como em Chirn e McFarlane [41], Uzam *et al* [42] e Uzam [43], Frey e Litz [44, 45, 46] e Uzam *et al* [47] para SEDs temporizados.

Entretanto, para a maioria dos casos tratados na literatura, a aplicação dos CLPs no controle de SEDs se baseia na estrutura de um modelo formalizado como uma rede *segura*. Para estes casos, a formalização do código de programa em diagrama Ladder para um lugar qualquer da rede se baseia na utilização de sinais binários (0 ou 1),

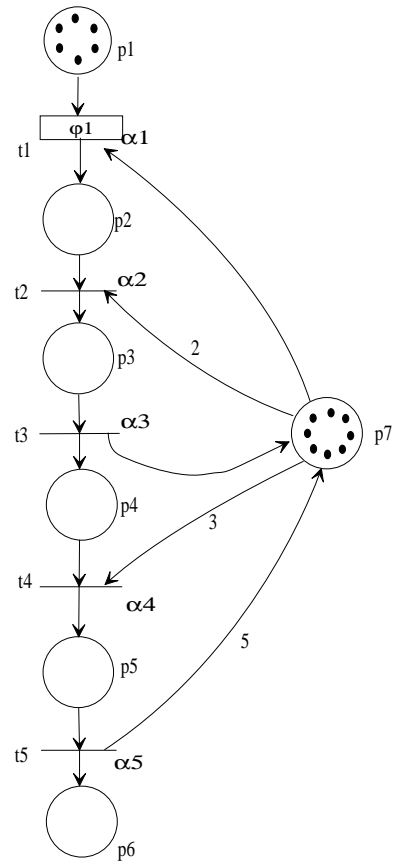


Figura 0.19: Modelo em RPFHT para um sistema de manufatura com recursos compartilhados, para realizar o processamento de duas peças em paralelo.

referenciando que o lugar está sem fichas (0) ou com uma ficha (1). Em outros termos, um lugar pode ser representado por um relé que memoriza o número de fichas (aberto - sem ficha; fechado - com ficha).

Quando são determinadas as condições de uma rede de Petri não segura (em que cada lugar pode ter mais de uma ficha, podendo reduzir ou aumentar este número de acordo com as seqüências de disparos das transições), podem-se utilizar contadores para representar estes lugares. É importante, inicialmente, definir o tipo de contador associado a cada lugar, se decrementativo ou incrementativo. Há ainda casos de lugares que necessitam ser modelados por contadores que são ao mesmo tempo incrementativos e decrementativos, para este caso temos que definir o tipo baseado no seu estado inicial. Um caso similar desta formalização pode ser visto em Uzam *et al* [42].

Como em determinados projetos de controle a rede de Petri *segura* apresenta um grande número de lugares e transições, isto pode ser considerado como um problema devido à limitação dos componentes de um CLP, necessitando a utilização de vários CLPs para a implementação de um supervisor. Neste caso, a necessidade da utilização de redes de Petri não seguras é de grande importância para tornar funcional projetos de supervisão de SEDs, pois várias abordagens de controle destes sistemas se baseiam nesta formalização, como é o caso da abordagem de Barroso. Para tanto, as redes de Petri síncronas são necessárias e definidas a seguir.

Uma rede de Petri Síncrona (RPS) [48] pode ser definida como em Silva [32]:

**Definição 6** *Uma Rede de Petri Síncrona é uma tripla*

$$RPS = (RP, E, FE),$$

*em que*

*RP* uma rede de Petri marcada;

*E* um conjunto de eventos externos;

*FE* uma função do conjunto de transições *T* da *RP* para  $E \cup \{e\}$ ,

*com e sendo o evento que possui ocorrência permanente, isto é, ele é o elemento neutro do monóide  $(E^1 + \dots + E^p)^*$ , onde  $E = \{E^1, \dots, E^p\}$  é o conjunto de eventos externos.*

Essa definição é uma formalização mais simplificada que em [48], em que se consideram todos os eventos externos com as mesmas prioridades, sendo a utilizada nesta



apostila. Nessa definição, o evento  $e$  é um evento que corresponde à sequência de eventos externos cujo período é zero. Uma transição sincronizada por este evento é disparada assim que ela se torna habilitada. A notação  $E^i$ , refere-se ao *nome* de um evento externo e a notação  $E_j$  será utilizada para o evento associado à transição  $t_j$ .

Para a habilitação de uma transição em uma RPS é necessário, além da habilitação formal definida para as redes de Petri (todos os lugares de entrada da transição devem ter o número de fichas que garantam seu disparo e os lugares de saída não ultrapassem sua capacidade), que um evento externo  $E$ , associado à transição, ocorra para que ela possa disparar. Esta situação define que a transição está receptiva ao evento  $E$ , e com sua ocorrência, ela dispara, gerando uma nova marcação conforme as regras de disparo das redes de Petri clássicas.

As condições citadas sobre a receptividade de uma transição  $t$  se apresentam como uma importante formalização para a estruturação de controle de SEDs, quando é desejado avaliar a dinâmica do sistema físico sobre a visão do modelo, desde que os eventos externos são as informações obtidas do sistema através dos sensores. Assim, o estado do modelo em RPS do sistema, seja este formalizado em um computador, ou em um CLP, é atualizado quando ocorre a chegada de uma informação de um determinado sensor do sistema.

**Exemplo 10** *Considere a RPS apresentada na Figura 0.20. Nesta rede, tem-se o conjunto de eventos externos que estão associados às transições dado por  $E = \{E^1, E^2, E^3, E^4\}$ . Os eventos  $E^1$  e  $E^2$  estão associados à transição  $t_1$ , tal que quando esta transição está habilitada, ela é receptiva a estes dois eventos. Para a transição  $t_2$ , há apenas um evento associado, que é o evento externo  $E^4$  e para a transição  $t_3$ , tem-se associado o evento  $E^3$ . Na marcação  $M_0 = [2 \ 0 \ 0]^T$ , a única transição habilitada é a transição  $t_1$ , a qual é receptiva aos eventos externos  $E^1$  e  $E^2$ . Assim, a ocorrência de qualquer um destes eventos faz a transição  $t_1$  disparar, levando à marcação  $M_1 = [1 \ 1 \ 0]^T$ , e uma nova ocorrência de qualquer um destes eventos leva a rede à marcação  $M_2 = [0 \ 2 \ 0]^T$ . Da mesma forma, na marcação  $M_1$ , se ocorre o evento  $E^3$ , a transição  $t_3$  que também está receptiva, dispara levando à marcação  $M_3 = [1 \ 0 \ 1]^T$ , e assim por diante. Entretanto, para uma marcação, como  $M_0$ , se ocorrer o evento  $E^4$ , que não está associado à transição  $t_1$ , não há mudança de marcação na rede, desde que ela é a única habilitada, porém não é receptiva a este evento.*

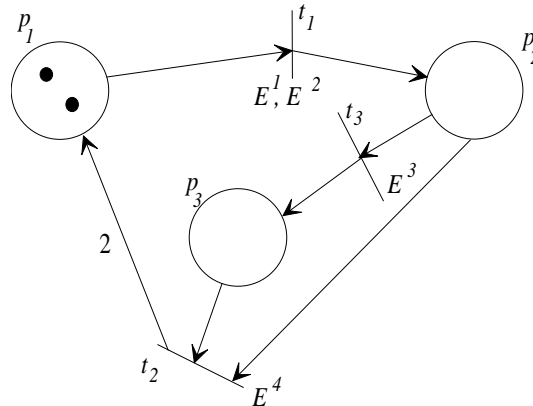


Figura 0.20: Exemplo de uma RPS.

Deve-se observar no Exemplo 10 que a diferença básica da regra de disparos das transições, refere-se unicamente à sua receptividade em relação a um evento externo que esteja associado à esta transição. Dessa forma, a evolução dinâmica da RPS é similar à rede de Petri clássica.

## Árvore de Alcançabilidade das RPS

Como visto no Exemplo 10, uma transição habilitada numa marcação  $M$  dispara se houver a ocorrência de um evento associado a esta transição. Uma sequência de ocorrências de eventos externos que estejam associados às transições receptivas leva aos disparos das mesmas. Dessa forma, a evolução dinâmica da RPS tem uma forma similar às redes de Petri clássicas, podendo ter a árvore de alcançabilidade descrita da mesma maneira, apenas associando aos arcos os eventos que tornam as respectivas transições receptivas, desde que os eventos externos se apresentem com iguais prioridades de ocorrência.

De acordo com as sequências de eventos externos que ocorrem no sistema, as transições irão disparar formando sequências de marcações, como visto no Exemplo 10. Assim, pode-se construir toda a árvore de alcançabilidade da RPS, desconsiderando a ocorrência de eventos externos que não estejam associados a nenhuma transição habilitada numa determinada marcação.

**Exemplo 11** A RPS do Exemplo 10 tem sua árvore de alcançabilidade mostrada na Figura 0.21.

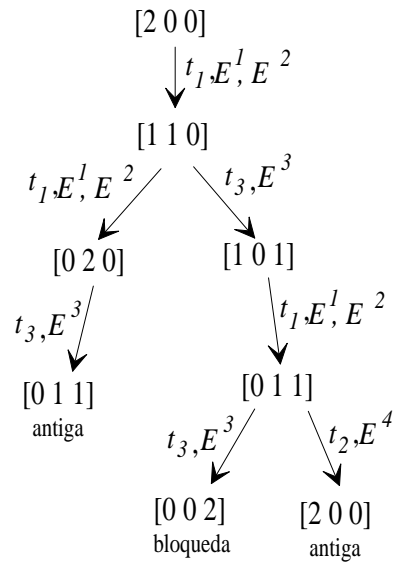


Figura 0.21: Árvore de alcançabilidade de uma RPS.

No Exemplo 11, como citado anteriormente, observa-se que apenas são consideradas as ocorrências dos eventos externos associados a transições que estejam habilitadas, tornando a árvore de alcançabilidade similar à árvore de alcançabilidade de uma rede de Petri clássica com a mesma estrutura (não considerando a inclusão dos eventos externos). Com este formalismo, pode-se utilizar a abordagem de Barroso [22] como base para construir supervisores utilizando RPS.

## Controladores Lógicos Programáveis (CLPs)

É um aparelho digital que usa memória programável para armazenar instruções que implementem funções como: lógica, seqüenciamento, temporização, contagem e operações aritméticas, para controlar através de módulos de entrada e saída diversos tipos de máquinas ou processos [49]. São compostos de CPU, memória, módulos de entrada e saída, linguagens de programação, dispositivo de programação, módulos de comunicação e módulos especiais (opcionais). Na Figura 0.22 é visto o diagrama de blocos de um CLP, como apresentado em [8], onde se podem ver os componentes internos que o compõe e a comunicação entre eles.

Entre outros, alguns exemplos de controladores são:

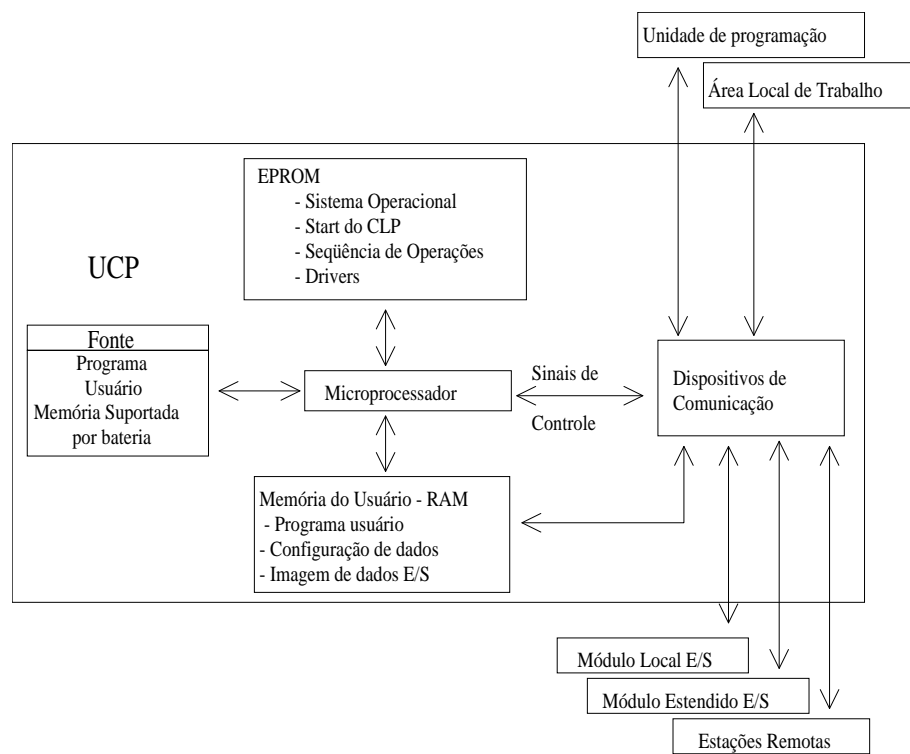


Figura 0.22: Diagrama de blocos de um CLP.

FESTO - FPC 101: É um CLP compacto, especialmente adequado para controlar pequenas instalações industriais de produção e processos. O controle consiste essencialmente de uma unidade de sinal de entrada, o circuito integrado de sinal de processamento e a unidade do sinal de saída. Pode ser programado com 3 linguagens compatíveis com a norma IEC 1131: Ladder, Lista de Instruções e Matrix.

TSX Premium - Fabricante Schneider Eletric: É um CLP de médio/grande porte destinado à automação de processos. Pode ser programado com 4 linguagens compatíveis com a norma IEC 1131: Ladder, Lista de Instruções, Grafcet (SFC) e Texto estruturado.

TBX MICRO - Fabricante Schneider Eletric: É destinado a fabricantes de máquinas e de pequenos sistemas; tem como recursos diversidade de dispositivos, processamento analógico e funções avançadas. Pode ser programado com 2 linguagens compatíveis com a norma IEC 1131: Ladder e Grafcet (SFC).

Controllogix - Fabricante Rockwell Automation 5500: É de grande porte e utilizado em automações de configurações complexas. Pode ser programado com 3 linguagens compatíveis com a norma IEC 1131: Ladder, Blocos de Funções e Grafcet (SFC).

PLC 5-80 - Fabricante Rockwell Automation 5500: É um controlador lógico programável de médio porte, de grande versatilidade e capacidade de comunicação em rede. Pode ser programado com 4 linguagens compatíveis com a norma IEC 1131: Ladder, Lista de Instruções, Grafcet (SFC) e Texto estruturado.

## **Principais Características do CLP Utilizado para Implementação**

O CLP utilizado nesta apostila é o da série FPC 101 [49] da FESTO.

O FPC é ligado através de suas entradas e saídas ao processo a ser controlado. No estado de operação, as entradas recebem informações através dos sensores (com ou sem contato físico) do sistema controlado, e as portas lógicas dessas informações, de acordo com as instruções do programa, determinam o estado de saída desejado. Os

sinais de saídas são emitidos para os atuadores, como motores, válvulas e embreagens magnéticas

O controlador possui operandos; internamente: 256 *flags*, 16 contadores, 32 temporizadores e 64 registradores e, externamente: canais de sinal digital sendo 20 entradas e 14 saídas.

Já o programa utilizado é o FST (Festo Software Tools) da série 100 da Festo [50] o qual permite programar nas linguagens Statement List (STL), Ladder Diagram (LDR) e Matrix (MAT). A linguagem de programação a ser utilizada será o diagrama Ladder [50], também conhecida como diagrama escada.

## Estrutura de um Programa em Diagrama Ladder

Uma das linguagens mais utilizadas para programação de CLPs é o diagrama Ladder. Esta linguagem foi especialmente desenvolvida para que a programação de controladores programáveis pudesse ser representada da mesma forma que diagramas elétricos [49]. Enquanto um circuito elétrico é representado entre linhas de alimentação horizontais, no diagrama Ladder um programa para um CLP é representado entre linhas verticais em forma de escada (Ladder). Devido a essa facilidade de representação, a linguagem se tornou simples e universal, podendo ser utilizada por qualquer pessoa que conheça circuitos elétricos.

No diagrama Ladder, um programa para CLP é escrito como um circuito na forma de uma escada seqüenciado em duas barras verticais, em que se tem no lado esquerdo toda a parte condicional (para entradas) e do lado direito a parte executável (para saídas). Basicamente, todo programa Ladder é fundamentado sobre condições e execuções (*if...then*). Ou seja, para cada condição satisfeita, executa-se uma ação ou resposta.

Ao editar um programa em diagrama Ladder, trabalha-se com operandos que podem ser por exemplo, as entradas/saídas do FPC 101, *flags*, temporizadores, etc. Esses operandos podem ser ainda absolutos ou simbólicos. Operandos absolutos são operandos padrões para os controladores FESTO. Por exemplo, para o FST 100 o operando o0.1, especifica um sinal de saída associado ao endereço 0.1 do controlador, enquanto os simbólicos são operandos representados por nomes de até nove caracteres alfanuméricos, onde o primeiro caractere deve ser, obrigatoriamente, uma letra. Estes nomes podem

refletir suas funções no programa, como por exemplo, *START*, que representa um sinal de entrada para a partida do processo controlado.

**Exemplo 12** Um exemplo de diagrama Ladder é apresentado na Figura 0.23, onde é vista a estrutura condição/execução. Nesta estrutura, a condição da entrada i0.0 (um contato normal aberto o qual é fechado ao ser pressionada a chave de comando) define a condição da saída o0.1 (que pode ser qualquer circuito). Se a entrada i0.0 se mantém aberta, a saída o0.1 nada tem como resposta (mantém-se no seu estado). Se a entrada i0.0 for fechada, a saída é “setada”, ou seja, seu nível lógico é tornado 1, ligando o equipamento conectado à esta saída. Da mesma forma, na segunda linha, tem-se que a condição na entrada i1.0 (contato normal fechado o qual é aberto ao ser pressionada a chave de comando) define a condição da saída o0.2 (que pode ser qualquer circuito). Se a chave da entrada i1.0 for acionada, a saída se mantém “resetada”, ou seja, seu nível lógico é mantido em 0, mantendo assim as condições iniciais, onde o equipamento conectado a esta saída estava desligado.

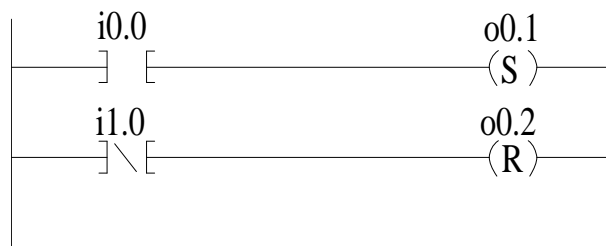


Figura 0.23: Exemplo de um diagrama Ladder.

Nos CLPs, esta estrutura de linguagem é muito utilizada, onde várias condições podem ser colocadas para teste das entradas, de maneira a definir as condições de várias saídas. Situações de “ou” lógico entre várias condições de entrada são descritas por condições em paralelo, e situações de “e” lógico são descritas por condições em série. Para as condições de saída, deve-se sempre considerar ligações em paralelo (condições “ou” lógico).

**Exemplo 13** Na Figura 0.24 é apresentada uma linha de código de um diagrama Ladder, onde se pode ler: se a chave i0.0 for acionada e i0.1 não for acionada, ou a chave i1.2 for fechada, então ligue e mantenha ligadas as saídas o0.1 e o1.1 e desligue e mantenha desligada a saída o0.0.

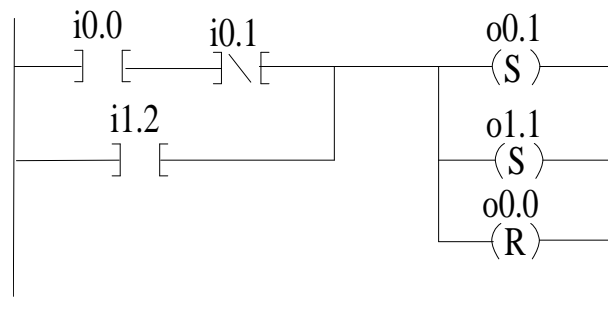


Figura 0.24: Exemplo de um diagrama Ladder com condições lógicas “e” e “ou” na entrada e “ou” na saída.

Naturalmente, para realizar um programa de CLP em diagrama Ladder é necessário definir as condições iniciais de todos os componentes do sistema, bem como estruturar as entradas/saídas a sensores/componentes do sistema a ser controlado. Desta forma, os CLPs contêm várias formalizações de dados, como registradores, temporizadores, relés, entre outros, em que o programador pode desenvolver a seqüência de controle a ser realizada no sistema, de acordo com os requerimentos desejados.

## Programação em Diagrama Ladder

Conforme citado na Seção anterior, os CLPs apresentam internamente e externamente operandos, que podem ser utilizados para desenvolver um programa de controle de um dado sistema em uma de suas linguagens (*STL*, *IL*, *Ladder*, etc.). A formalização destes componentes em uma estrutura lógica de um programa permite descrever a dinâmica de um sistema ligado ao CLP. Abaixo, serão apresentados os operandos necessários para transformação da RPFHT e sua modelagem em diagrama Ladder. Nesta apostila, não são tratados os casos temporizados.

### Bobinas:

Podem assumir as funções de atribuição não retentiva

—( )—

atribuição negada não retentiva

—( / )—



ativa e retém

—( *S* )—

desativa

—( *R* )—

incrementa contador

—(*INC*)—

e decrementa contador

—(*DEC*)—

Sua representação, conforme descrita anteriormente, é feita na parte executiva da linha de comando do diagrama. Nesta posição podem ser especificadas saídas, *flags*, temporizadores e contadores. Nos exemplos das Figuras 0.23 e 0.24 as saídas são representação de bobinas em Ladder, onde a representação *ox.x* está relacionada ao endereço absoluto do CLP.

### Entradas:

A representação das entradas é definida na parte condicional da linha de comando do diagrama. Estão associadas em geral a sinais oriundos do sistema. Estas informações são retiradas do sistema através de sensores e chaves de partida. Nos exemplos das Figuras 0.23 e 0.24 as entradas são modeladas por contatos normal aberto (*NA*) e normal fechado (*NF*), onde a representação *ix.x* está relacionada ao endereço absoluto do CLP.

### Contadores:

Para os contadores, a representação é definida na parte executável do diagrama. Sua modelagem envolve inicialmente a escolha do tipo de contador: se incrementativo ou decrementativo.

**Exemplo 14** *A Figura 0.25 apresenta um exemplo de um contador incrementativo. Quando da chegada de um sinal de partida, representado pelo operando simbólico START o contador é inicializado, e desta forma a saída representada pelo simbólico LAMP será ativada. Existindo uma lâmpada ligada a esta saída a mesma acenderá.*

O operando absoluto C0 representa um flag para determinar o estado da contagem. Através dele, pode-se saber se a contagem programada já decorreu ou não. O número 2 representa o valor final de contagens progressivas (crescentes). Decorridos dois pulsos através do operando absoluto I1.3, o flag C0 será desativado e a lâmpada apagará. Na hipótese do operando absoluto I0.1 enviar 2 pulsos após o contador ser inicializado, o flag C0 só será desativado após 4 pulsos no operando I1.3.

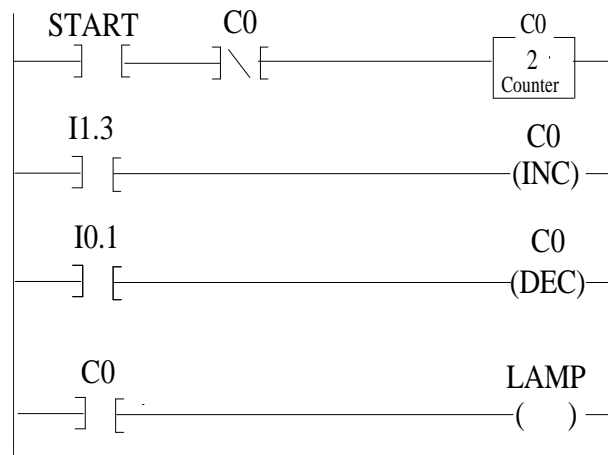


Figura 0.25: Exemplo de um Contador Incrementador

**Exemplo 15** Um exemplo de contador decrementador é mostrado na Figura 0.26. Quando da chegada de um sinal de partida, representado pelo operando simbólico *START* o contador é inicializado, e desta forma a saída representada pelo simbólico *LAMP* será ativada, existindo uma lâmpada ligada a esta saída a mesma acenderá. O operando absoluto C0 tem as mesmas funções que as do contador incrementador. O número 2 representa o valor final de contagens regressivas (decrementantes). Decorrido dois pulsos através do operando absoluto I0.1, o flag C0 será desativado e a lâmpada apagará. Na hipótese do operando absoluto I1.3 enviar 2 pulsos após o contador ser inicializado, o flag C0 só será desativado após 4 pulsos no operando I0.1.

### Flags:

Sua representação, enquanto bobina, é definida na parte executiva da linha de comando do diagrama. No caso de contatos associados à bobina a representação é feita na parte

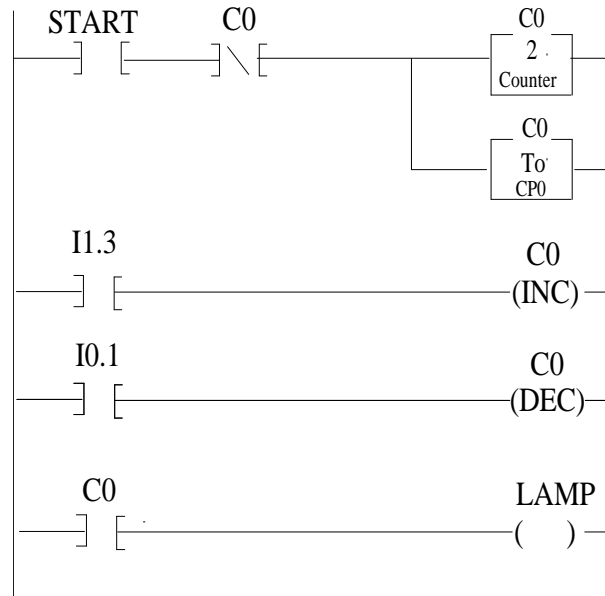


Figura 0.26: Exemplo de um Contador Decrementador

condicional da linha de comando. Os *flags* podem assumir dois estados: ativos ou inativos.

**Exemplo 16** A Figura 0.27 apresenta um exemplo de um flag. Quando da chegada de um sinal de partida, representado pelo operando simbólico *START*, o flag *F0.0* é ativado. Desta forma, a saída representada pelo simbólico *LAMP* será ativada e, existindo uma lâmpada ligada a esta saída, a mesma acenderá imediatamente.

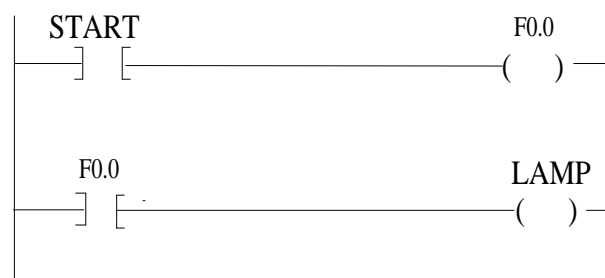


Figura 0.27: Exemplo de um Flag

Estes operandos, só ou em combinação permitem modelar sistemas de controle para SEDs, sejam eles por meios de autômatos [13, 28] ou redes de Petri [41, 42], o que será visto a seguir.

## Transformação de uma RPFHT em Diagrama Ladder

Utilizando os formalismos de [42, 41, 44, 45, 46], pode-se construir o código em diagrama Ladder para um CLP de uma rede de Petri segura, onde os lugares são representados por relés. No entanto, em se tratando de uma rede de Petri não segura, são necessários outros formalismos para descrever os lugares, como é o caso da utilização dos contadores cujo valor atualizado define o número de fichas no lugar, como pode ser visto em [47] para as redes de Petri T-Temporizadas e em [42] para as redes de Petri não temporizadas.

Para transformar uma rede de Petri em um diagrama Ladder, utiliza-se especificamente neste caso, a rede de Petri síncrona, definida a seguir:

A descrição de uma RPFHT, que é uma classe de rede de Petri com funções de habilitação de transições  $\phi$  incluídas em sua estrutura de maneira a gerar o controle de sua evolução dinâmica. A transformação de uma RPFHT em diagrama Ladder será feita utilizando o Algoritmo para Conversão da RPFHT para Ladder (ACRPFHTL), o qual é uma adaptação do ACRL [51] e é apresentado a seguir. Também, deve-se considerar que as RPFHT supervisoras incluem os eventos externos associados às transições para sua melhor visualização e formalização da transformação para Ladder.

### **Algoritmo 5** *Algoritmo para Conversão da RPFHT para Ladder - ACRPFHTL*

- *Início*

1. *Criar variáveis internas no CLP para representar cada um dos lugares ( $p_i$ ) da rede supervisora:  $P_i$  para lugares binários ou  $C_i$  para lugares  $k$ -limitados ( $k > 1$ ). O valor inicial de cada uma das variáveis que representam os lugares, corresponde a quantidade de fichas em cada lugar na marcação inicial.*
  - i. *Cada lugar 1-limitado deve ser modelado com um flag do CLP na posição de execução.*
  - ii. *Cada lugar  $k$ -limitado ( $k > 1$ ) deve ser modelado por um contador na posição de execução.*
  - iii. *As funções de habilitação de transição ( $\phi$ ) devem ser modeladas com blocos aritméticos comparadores na posição de teste (condição).*

2. Para construir a parte de atualização de estados e controle, deve-se construir uma lógica para cada transição da rede supervisora da seguinte maneira:
    - i. Os lugares de entrada da transição são associados em lógica AND.
    - ii. Se a transição for uma transição controlável agregar com sobreposição a lógica dos lugares de entrada com a lógica da função de habilitação desta transição.
    - iii. Se à transição estiver associado um canal de entrada do CLP, agregar o canal de entrada em lógica AND com os lugares de entrada.
    - iv. Os lugares de saída da transição são associados em lógica OR. Eles são representados pelas variáveis internas.
    - v. Se à transição estiver associado um canal de saída do CLP, agregar o canal de saída em lógica OR com os lugares de saídas.
  3. Ao final do disparo de cada transição, fazer uma atualização dos estados (marcação da rede).
- Fim

Neste algoritmo, o **passo 2.i.** é visto na Figura 0.28, a qual sua parte condicional apresenta a modelagem das entradas da transição  $t_2$  da rede da Figura 0.36 (lógica AND). O **passo 2.ii.** é também visto na Figura 0.28, a qual modela as entradas da transição  $t_2$  (função  $\phi_2$ ).

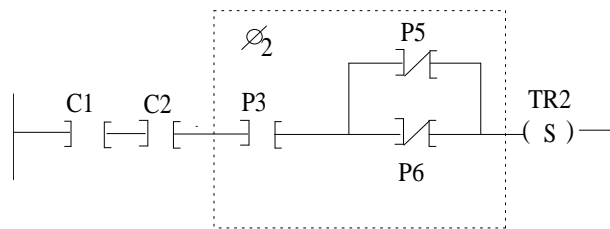


Figura 0.28: Agregação da lógica dos lugares de entrada à variável interna que representa a função de habilitação de uma transição.

O **passo 2.iii.** é visto na Figura 0.29.

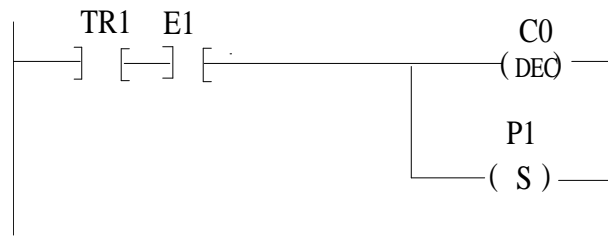


Figura 0.29: Agregação de canal de entrada à lógica dos lugares de entrada de uma transição.



Figura 0.30: Atualização dos estados.

O **passo 3.** é visto na Figura 0.30, a qual modela a desabilitação da transição  $t_1$ , considerando o bloco comparador utilizado como teste do valor de contagem de um contador (CP0) com o valor 1 gravado no registrador  $R$  (R1).

Os demais passos deste algoritmo são vistos no Exemplo final desta apostila (Figuras 0.37 a 0.41).

Desta forma, este algoritmo realiza a transformação da rede supervisora (RPFHT) para o diagrama Ladder, o qual gera a execução do sistema supervisionado pelo CLP.

A seguir é apresentado um exemplo de um sistema modelado em rede de Petri síncrona apresentado em [51] e considerando apenas a estrutura de construção de supervisores de Barroso [22] incluindo os eventos externos, e o diagrama Ladder do supervisor do sistema, gerado utilizando os algoritmos apresentados.

## Exemplos Demonstrativos

Nesta Seção são apresentados alguns exemplos de transformação de redes de Petri em diagrama Ladder. Deve-se observar que nos 3 primeiros exemplos, não são utilizados eventos externos na rede (não são utilizadas RPS), sendo uma representação direta da rede das Figuras citadas. O problema desta formalização é que sem os eventos externos, a velocidade da execução do sistema não acompanha a evolução do CLP.

Quando se utilizam as respostas dos eventos externos (sensibilizações dos sensores), estes sincronizam a evolução do CLP ao sistema. No terceiro exemplo, é mostrada tanto a rede de Petri em diagrama Ladder, como a modificação da linha do diagrama Ladder para a inclusão do controle (RPFHT).

### Exemplo 1

A Figura 0.4 tem sua representação em diagrama Ladder como apresentado na Figura 0.31. Nesta Figura, vê-se que o lugar  $p_2$  é representado por um contador, desde que é o único que pode ter uma marcação maior que 1. Inclusive, ele é inicializado como um contador incrementador, pois sua marcação inicial é zero, podendo ser aumentada com o disparo da transição  $t_3$ . Depois de ter alguma ficha neste lugar, se a transição  $t_1$  disparar, a transição  $t_2$  se torna habilitada, podendo disparar, o que passa a decrementar o número de fichas no lugar  $p_2$ . Os lugares  $p_1$  e  $p_3$  são representados por *flags*, desde que suas marcações sempre são binárias. Observe que a transição  $t_0$  não está representada no diagrama Ladder, desde que ela é uma transição morta (nunca pode disparar).

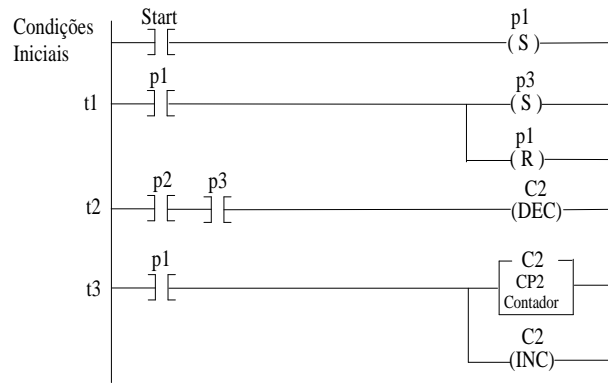


Figura 0.31: Diagrama Ladder para a rede de Petri da Figura 0.4.

### Exemplo 2

A Figura 0.13 tem sua representação em diagrama Ladder como apresentado na Figura 0.32. Nesta Figura, vê-se que o lugar  $PA$  é representado por um contador decrementador, bem como o lugar  $BA$ , enquanto que o lugar  $BS$  é um contador incrementador.

Os demais lugares são *flags*, desde que suas marcações sempre são binárias. A linha inicial representa as condições iniciais da rede, e cada uma das demais linhas representa uma transição.

### Exemplo 3

A Figura 0.18 tem sua representação em diagrama Ladder como apresentado na Figura 0.31. Nesta Figura, vê-se que todos os lugares são representados por contadores, pois todos alcançam marcações maiores que 1. Os blocos comparadores nas regiões de condição do diagrama Ladder (linhas 3 e 5) são utilizados para garantirem as condições de disparo da rede (exigência do número mínimo de fichas nos devidos lugares). Na atualização dos estados, encontram-se os blocos aritméticos de subtração (linhas 3 e 5) e de soma (linha 6) que representam a retirada/colocação do número de fichas dos respectivos lugares (pesos dos arcos). Quando o peso do arco é unitário, é necessário apenas o operador de incremento/decremento.

Quando é definido um controle para a evolução de uma rede, isto é, é formalizado um supervisor através da inclusão de funções de habilitação de transições, a mudança na estrutura da rede é feita apenas nas regiões condicionais das respectivas transições, como é o caso da função  $\phi_1$  introduzida nesta rede (ver Figura 0.19). Assim, a única diferença do diagrama Ladder da Figura 0.31, para a inclusão desta função é feita na linha 2, que representa a transição  $t_1$ , como visto na Figura 0.34, que é um bloco comparador que avalia se o número de fichas satisfaz o que a função pede.

### Exemplo 4

Na Figura 0.35 é apresentado um simples sistema composto por um braço robótico - *br* (motor de corrente contínua acionando um parafuso sem fim tendo um atuador de dupla ação montado sobre o mesmo), três atuadores ( $A_1$ ,  $A_2$  e  $A_3$ ), um *buffer* de entrada (que recebe peças), um *buffer* de saída, um *buffer* de peças rejeitadas e três localizações específicas ( $L_1$ ,  $L_2$  e  $L_3$ ) para testar peças. Neste sistema, o braço robótico pode pegar peças no *buffer* de entrada e colocá-los nos lugares  $L_1$  e  $L_2$ , bem como movimentar as peças entre estes dois lugares de teste. O atuador  $A_1$  pode empurrar a peça do lugar  $L_1$  para o *buffer* de peças rejeitadas. O atuador  $A_2$  pode empurrar peças para uma posição intermediária entre  $L_2$  e o *buffer* de saída ( $L_3$ ) ou diretamente para



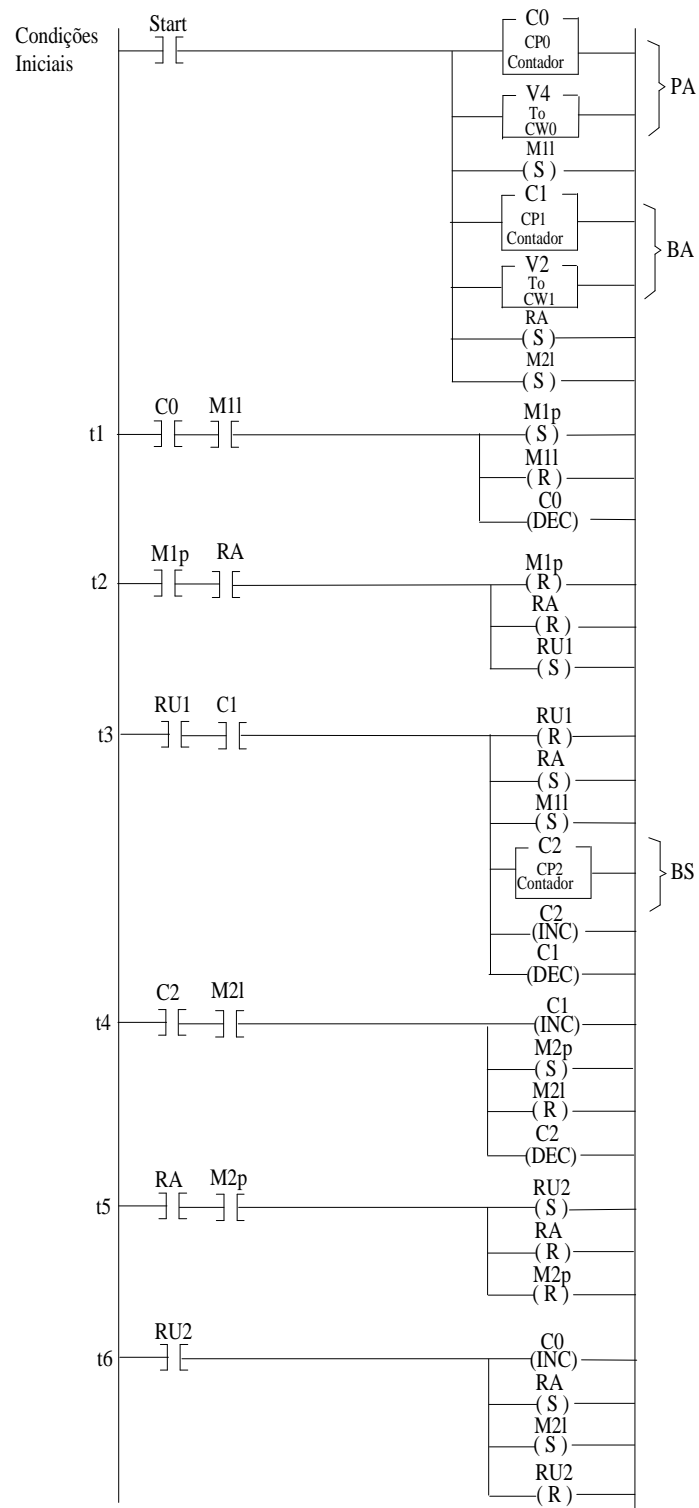


Figura 0.32: Diagrama Ladder para a rede de Petri da Figura 0.13.

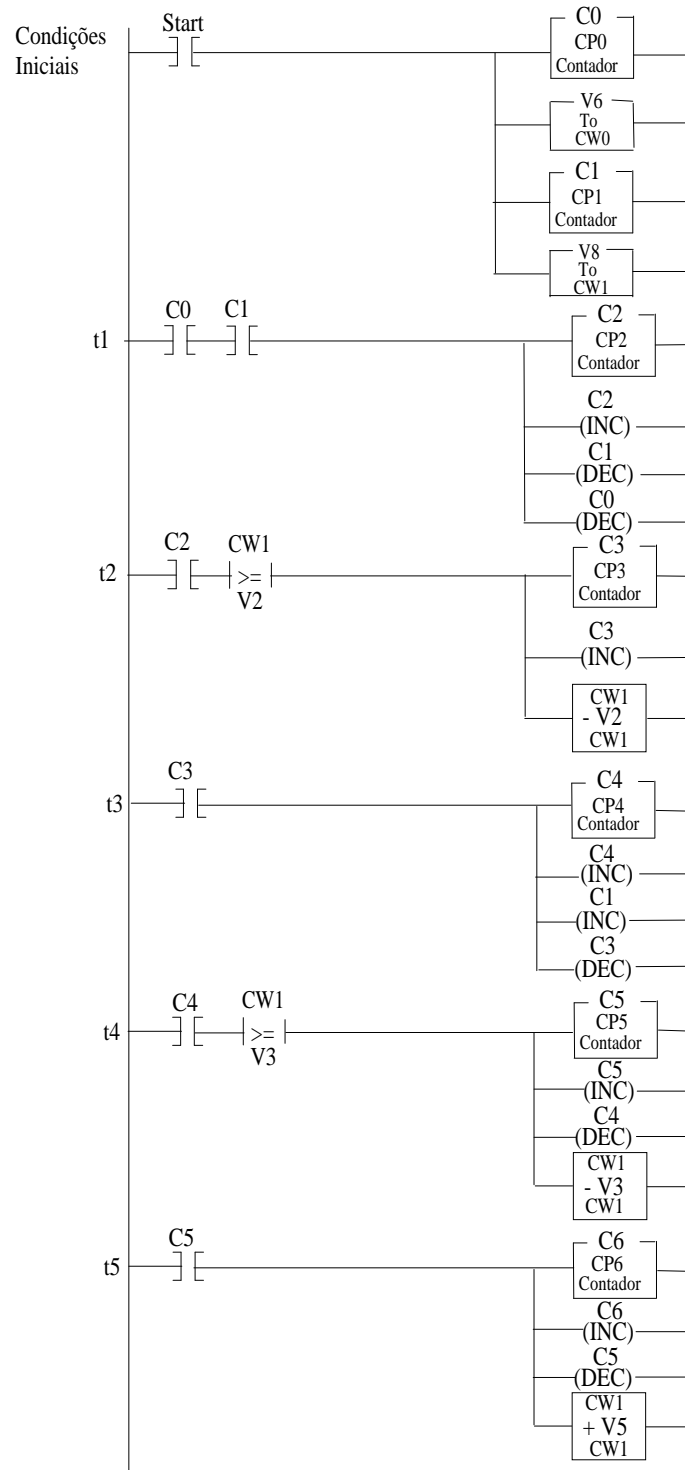


Figura 0.33: Diagrama Ladder para a rede de Petri da Figura 0.18.



- Lugar  $p_0$  - *buffer* (lugar) de entrada;
- Lugar  $p_1$  - peça na posição para ser pega pelo braço robótico;
- Lugar  $p_2$  - braço robótico sem peça;
- Lugar  $p_3$  - braço robótico se dirigindo para o centro (lugar  $L_1$ );
- Lugar  $p_4$  - braço robótico com peça;
- Lugar  $p_5$  - lugar de teste  $L_1$ ;
- Lugar  $p_6$  - lugar de teste  $L_2$ ;
- Lugar  $p_7$  - atuador  $A_1$  recolhido;
- Lugar  $p_8$  - atuador  $A_1$  adiantado;
- Lugar  $p_9$  - atuador  $A_2$  adiantado em meio passo (lugar de teste  $L_3$ );
- Lugar  $p_{10}$  - atuador  $A_2$  recolhido;
- Lugar  $p_{11}$  - atuador  $A_2$  adiantado completamente para a saída (*buffer* de saída);
- Lugar  $p_{12}$  - *buffer* de saída;
- Lugar  $p_{13}$  - atuador  $A_3$  recolhido;
- Lugar  $p_{14}$  - atuador  $A_3$  adiantado (da posição de meio passo para a saída);
- Lugar  $p_{15}$  - lugar de saída de peças rejeitadas;
- Lugar  $p_{16}$  - braço robótico se movimentando para a posição de entrada ou para  $L_2$ ;
- Lugar  $p_{17}$  - braço robótico se movimentando para esquerda (em direção ao centro - lugar  $L_2$ ).

• **Transições:**

- Transição  $t_1$  - chegada de peça na posição que o braço robótico pode pegar;
- Transição  $t_2$  - braço robótico chegar à posição de pegar peça no *buffer* de entrada (pegar peça no *buffer* de entrada);
- Transição  $t_3$  - braço robótico sem peça se movimentar para o centro (em direção à  $L_1$ );

- Transição  $t_4$  - braço robótico sem peça se movimentar para  $L_2$ ;
- Transição  $t_5$  - braço robótico soltar peça em  $L_1$ ;
- Transição  $t_6$  - braço robótico soltar peça em  $L_2$ ;
- Transição  $t_7$  - braço robótico pegar peça em  $L_1$ ;
- Transição  $t_8$  - atuador  $A_1$  se adiantar;
- Transição  $t_9$  - atuador  $A_3$  se adiantar;
- Transição  $t_{10}$  - atuador  $A_2$  se adiantar pela metade (meio passo até  $L_3$ );
- Transição  $t_{11}$  - atuador  $A_2$  se adiantar por completo de uma só vez (para o *buffer* de saída);
- Transição  $t_{12}$  - peça ser depositada na saída;
- Transição  $t_{13}$  - atuador  $A_1$  ser recolhido;
- Transição  $t_{14}$  - atuador  $A_3$  ser recolhido e peça ser rejeitada;
- Transição  $t_{15}$  - atuador  $A_2$  ser adiantado da posição de meio passo para o final (de  $L_3$  para o *buffer* de saída);
- Transição  $t_{16}$  - braço robótico chegar no lugar de entrada;
- Transição  $t_{17}$  - braço robótico chegar no centro;
- Transição  $t_{18}$  - braço robótico com peça se movimentar para o centro (lugar de teste  $L_1$ );
- Transição  $t_{19}$  - braço robótico com peça se movimentar para o lugar de teste  $L_2$ .

Neste modelo não há diferenciação entre as peças a serem trabalhadas no sistema (ao todo oito peças), cabendo aos sensores detectarem os tipos. Para este caso, é considerado que há três peças plásticas de cor laranja, três peças plásticas de cor preta e duas peças metálicas.

Neste sistema, são transições não controláveis:  $t_3$ ,  $t_4$ ,  $t_{12}$ ,  $t_{13}$ ,  $t_{14}$ ,  $t_{16}$ ,  $t_{17}$ ,  $t_{18}$  e  $t_{19}$  que se referem às chegadas de peças aos *buffers* de saída/peças rejeitadas através dos atuadores. As demais são transições controláveis.

Para este sistema, é desejado que o braço robótico só pegue e libere peças nos devidos lugares, garantindo que cada lugar contenha no máximo uma peça, e que as

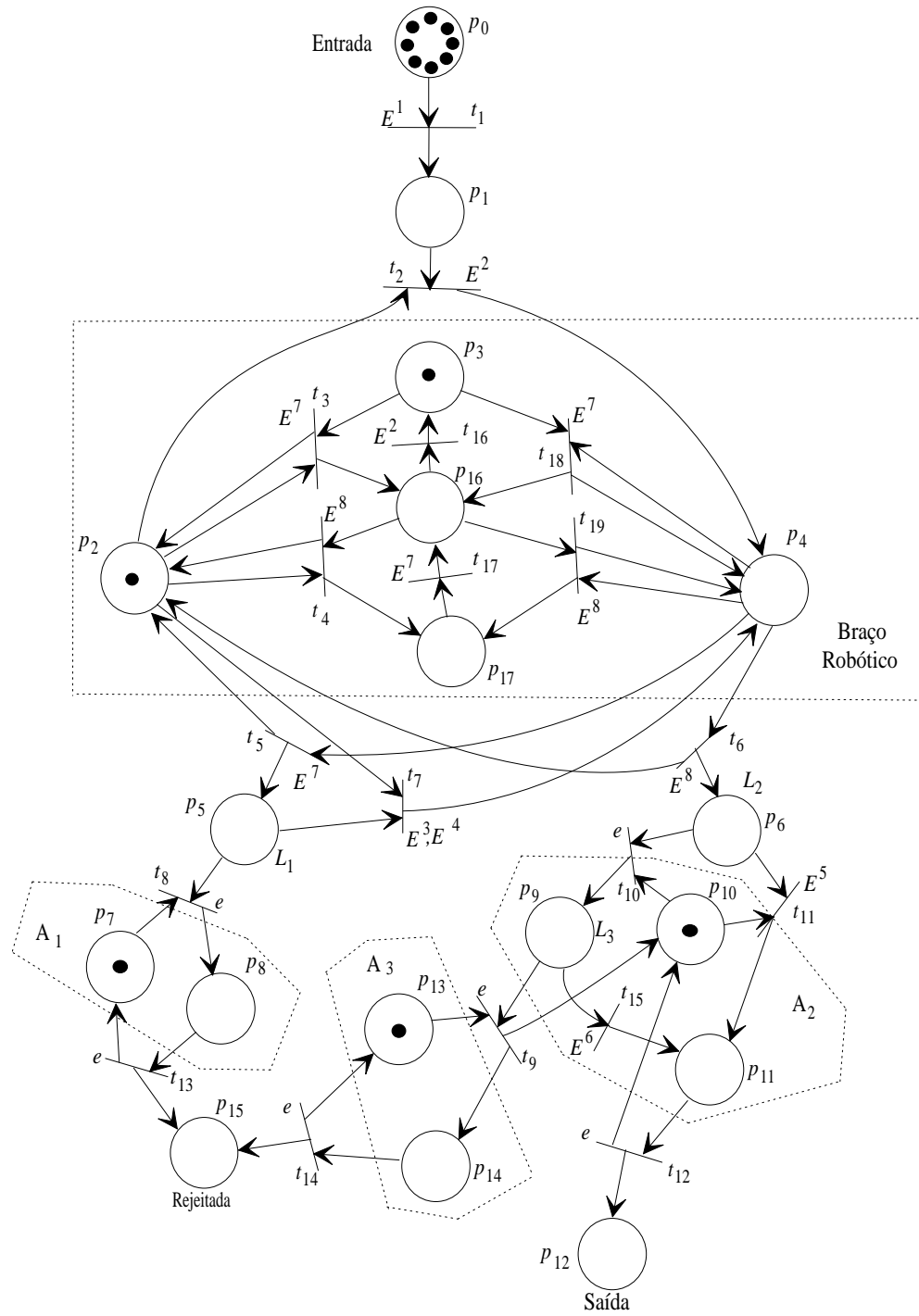


Figura 0.36: Modelo do sistema via RPS.

peças rejeitadas sejam as peças plásticas de cor preta, em que o sistema deve trabalhar continuamente. Ou seja, a marcação final a ser alcançada deve ser

$$M = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 5 & 1 & 0 & 3 & 0 & 0 \end{bmatrix}^T,$$

em que se entende que após terminado o processamento das oito peças, o braço robótico retorne a sua posição inicial e todos os atuadores estejam recuados com todas as peças plásticas de cor preta no lugar de peças rejeitadas e as demais, na saída.

Utilizando o AMArA, constrói-se a árvore de alcançabilidade da rede, na qual nenhuma marcação é bloqueada. Entretanto, algumas marcações são não permitidas, como é o caso de marcações alcançadas que definam mais de uma peça em algum lugar de teste (*buffer* de entrada -  $p_1$ ,  $L_1 - p_5$  e  $L_2 - p_6$ :  $M(p_1) > 1$ ,  $M(p_5) > 1$  e  $M(p_6) > 1$ , desde que cada lugar destes somente podem conter uma única peça, e o posicionamento do braço robótico sobre o lugar de entrada para pegar uma peça. Neste caso, através da execução do ACGS, encontram-se as funções de habilitação de transições definidas por:

$$\begin{aligned} \phi_1 &= [M(p_1) \leq 1], \\ \phi_2 &= [(M(p_3) = 1) \wedge ((M(p_5) < 1) \vee (M(p_6) < 1))], \\ \phi_5 &= [(M(p_5) < 1) \wedge (M(p_{16}) = 1) \wedge (M(p_7) = 1)], \\ \phi_6 &= [(M(p_6) < 1) \wedge (M(p_{17}) = 1) \wedge (M(p_{10}) = 1)], \\ \phi_8 &= 0. \end{aligned}$$

Estas funções são as condições mínimas para a execução do controle do sistema. A rede supervisora tem a mesma estrutura do modelo e garante a execução contínua do sistema para a realização da tarefa especificada.

No geral, para este caso, o supervisor garante que o sistema não alcance estados não permitidos, realizando a tarefa determinada.

Agora, considerando este sistema e o supervisor construído, tem-se que a utilização do ACRPFHTL gera o diagrama Ladder apresentado nas Figuras 0.37, 0.38, 0.39, 0.40, 0.41 e 0.41, as quais são sequenciais. Nestas Figuras, vê-se que:

- A linha 1 corresponde à definição das condições iniciais (**passo 1** do algoritmo) estabelecendo o disparo do contador decrementador  $C0$ , as posições que partem com fichas,  $P2$ ,  $P3$ ,  $P7$ ,  $P10$  e  $P13$ .

- As demais linhas correspondem à parte de atualização de estados e modelagem dos lugares de entradas das transições (**passo 2** do algoritmo).
- As partes executáveis de todas as linhas, excluindo a linha de inicialização do sistema (linha 1), correspondem aos lugares de saídas das transições (**passo 2.iv** do algoritmo).
- Observando por exemplo, a linha 3, tem-se a formalização da transição  $t_2$ . Nesta linha encontra-se a definição da função  $\phi_2$ , em que se pode ver na parte condicional a lógica *AND* das entradas da transição ( $p_1 \wedge p_2$ ) com a função  $\phi_2 = p_3 \wedge (\overline{p_5} \vee \overline{p_6})$ . Em seguida a estas condições (condição de disparo clássico e condição de disparo da função  $\phi_2$ ), encontra-se, também em lógica *AND* com este conjunto, o canal de entrada *B2* (que representa a ocorrência do evento  $E^2$ ). Na parte executável, encontra-se a atualização dos estados da saída da transição: ficha em  $p_4$  - *flag P4* setado; retirada de ficha em  $p_1$  e  $p_2$  - *flags P1* e *P2* resetados, e, em sequência, o avanço do braço robótico (*AVBR* setado), recuo do braço robótico (*RECBR* resetado), vácuo ligado (*VACUO* setado) e chamada da linha 5 que é uma linha auxiliar para realizar a inversão do movimento do atuador do braço robótico e partir o motor (movimento do braço). Esta linha 5 recebe em sua parte condicional o sinal do canal de entrada (*B9* que é um sensor magnético que se encontra no final de curso do atuador) e inverte a sequência da linha que a chama: reseto o avanço do braço robótico (*AVBR* resetado), liga o recuo do braço robótico (*RECBR* setado), liga motor para giro no sentido horário (*MOTORH* setado) e reseto o giro do motor no sentido anti-horário (*MOTORAH* resetado) e atualiza o próprio estado (reseto a linha *AUX*).



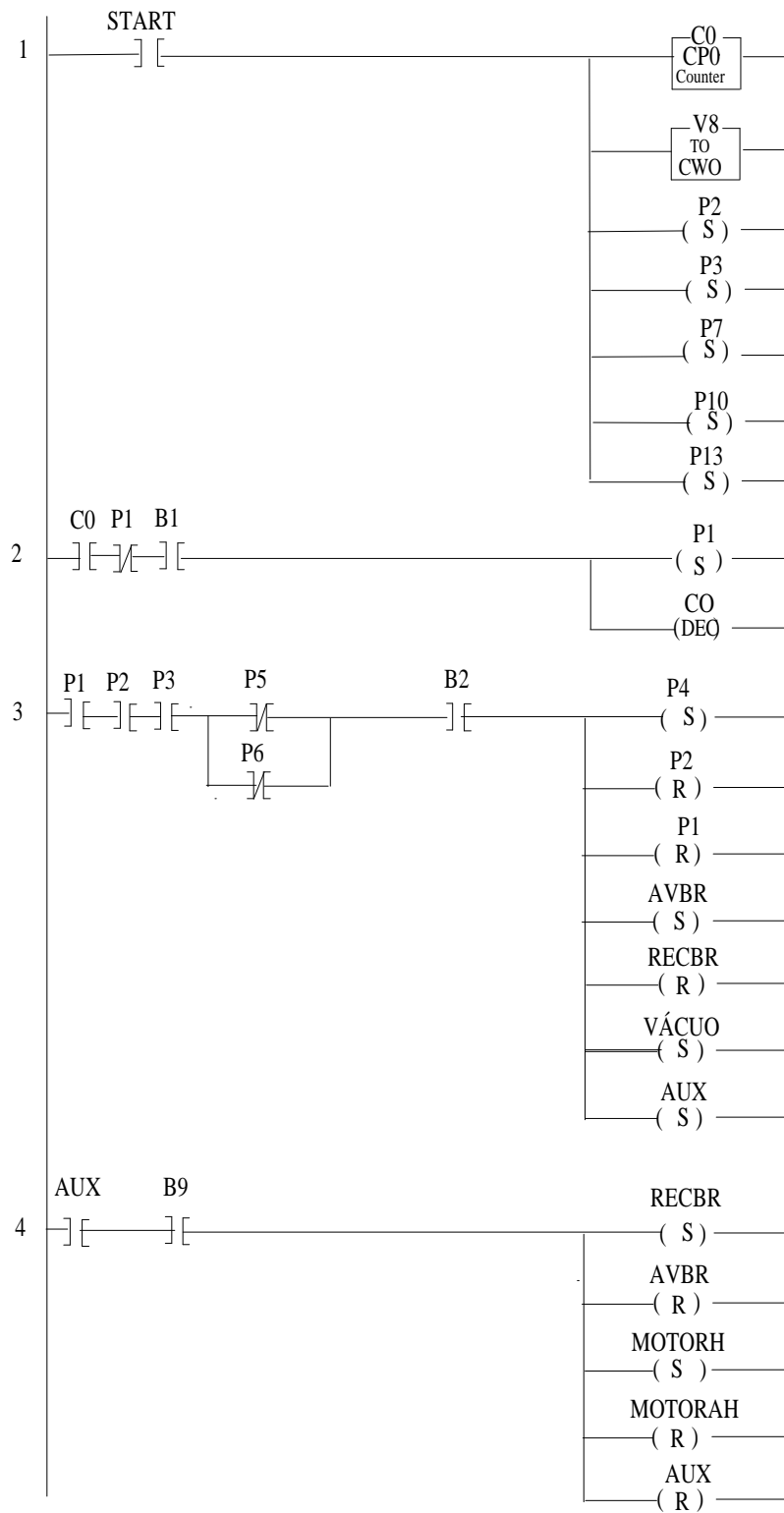


Figura 0.37: Parte 1 do diagrama Ladder do supervisor.

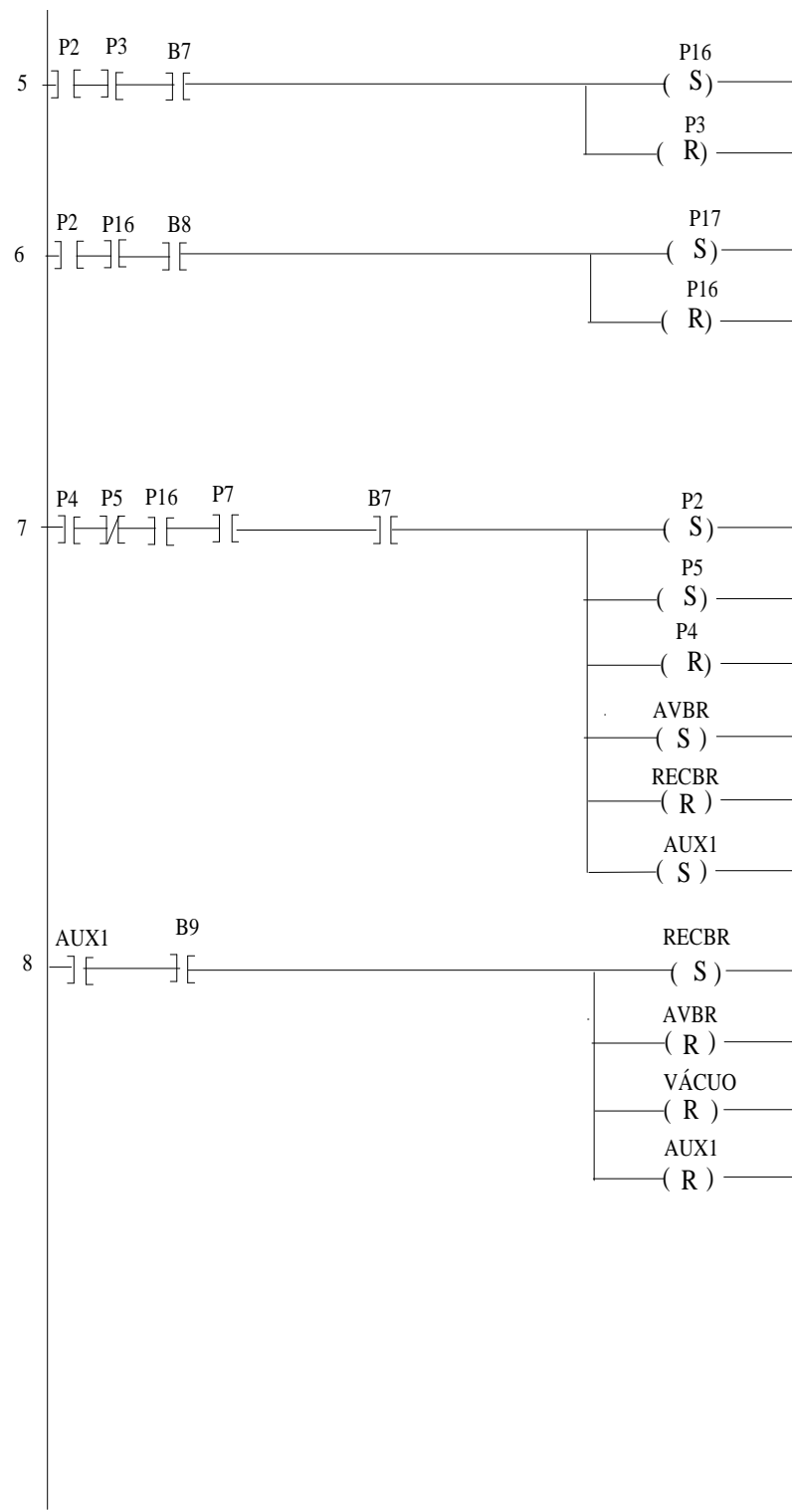


Figura 0.38: Parte 2 do diagrama Ladder do supervisor.

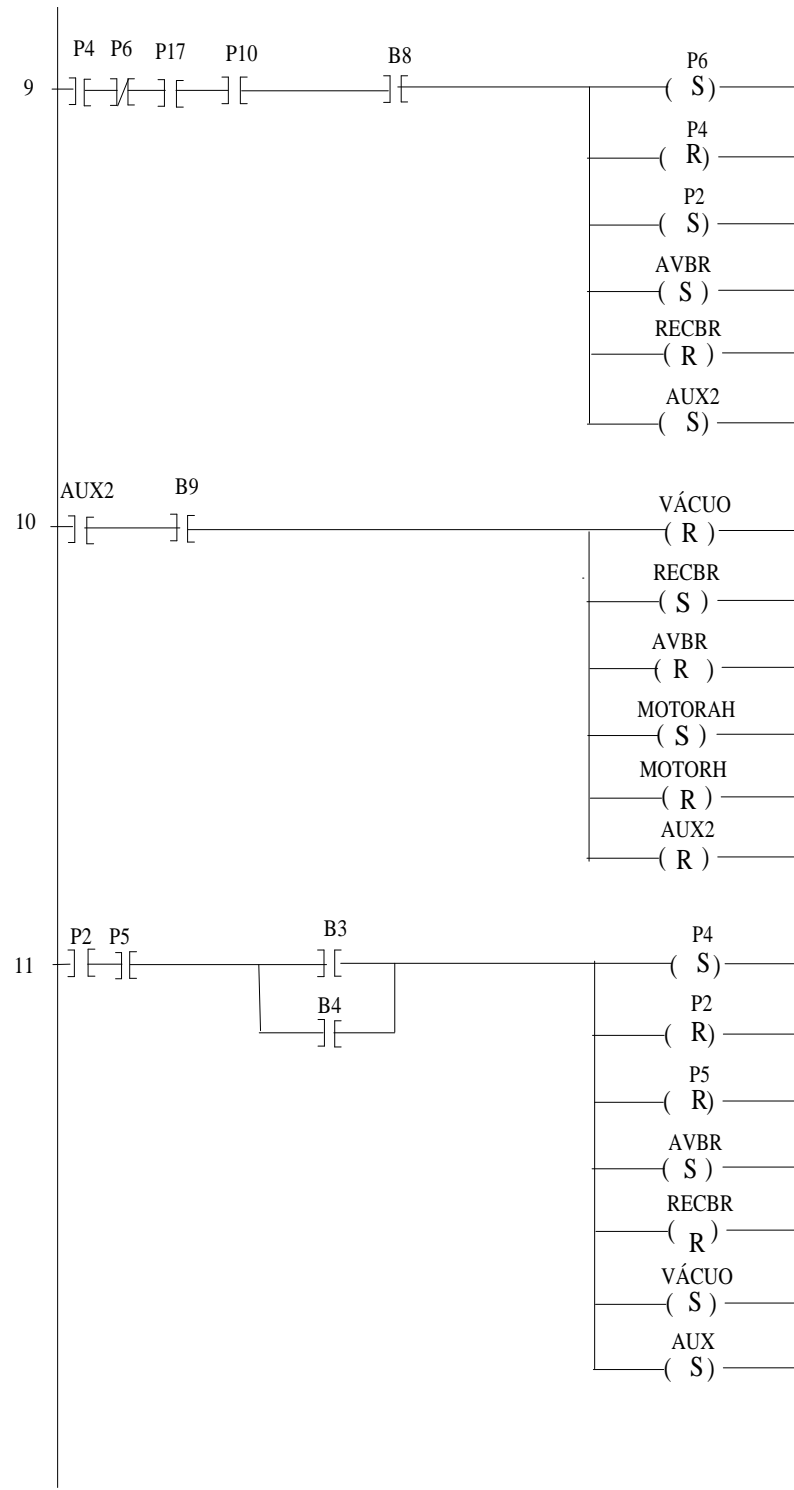


Figura 0.39: Parte 3 do diagrama Ladder do supervisor.

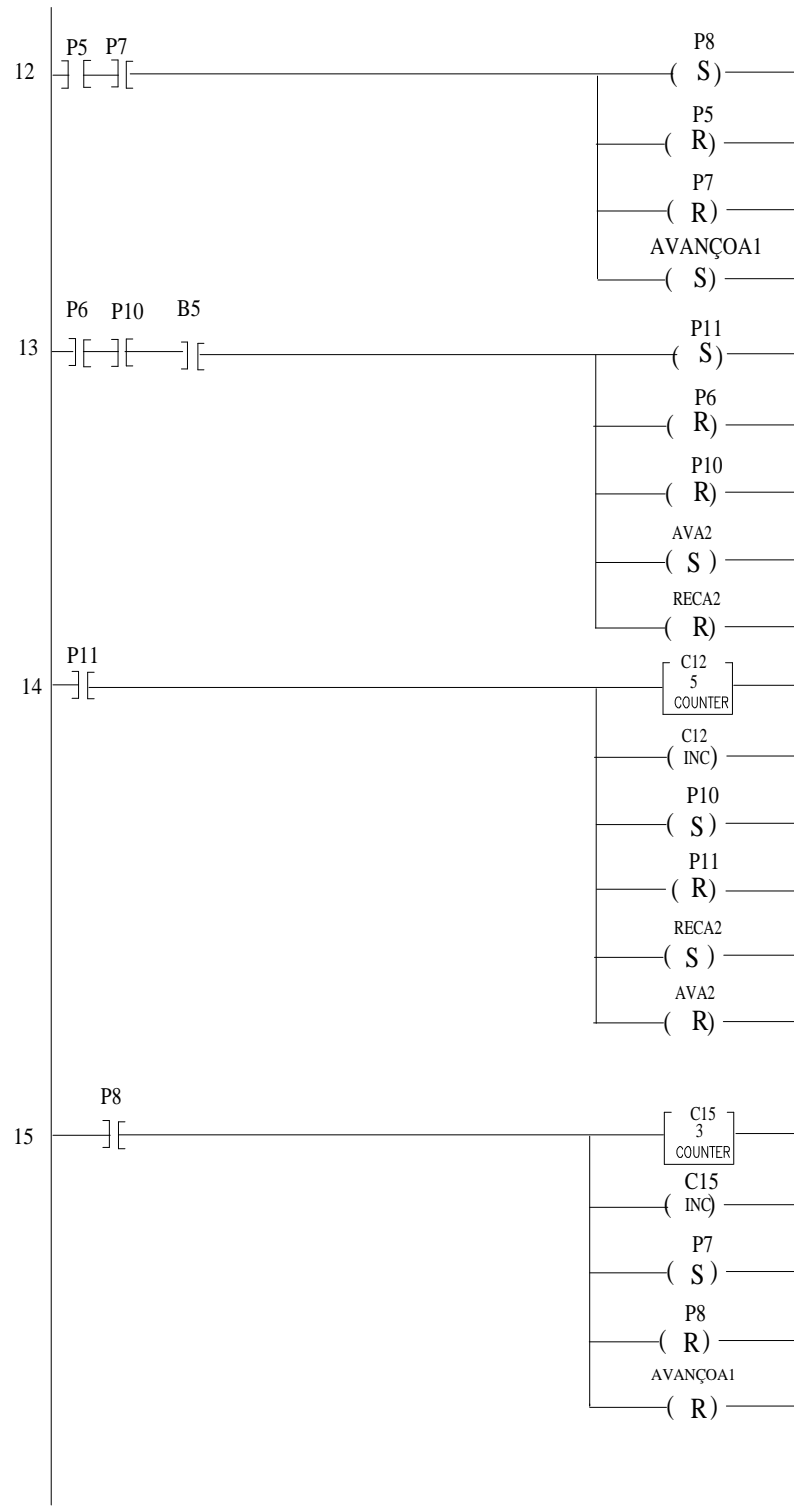


Figura 0.40: Parte 4 do diagrama Ladder do supervisor.

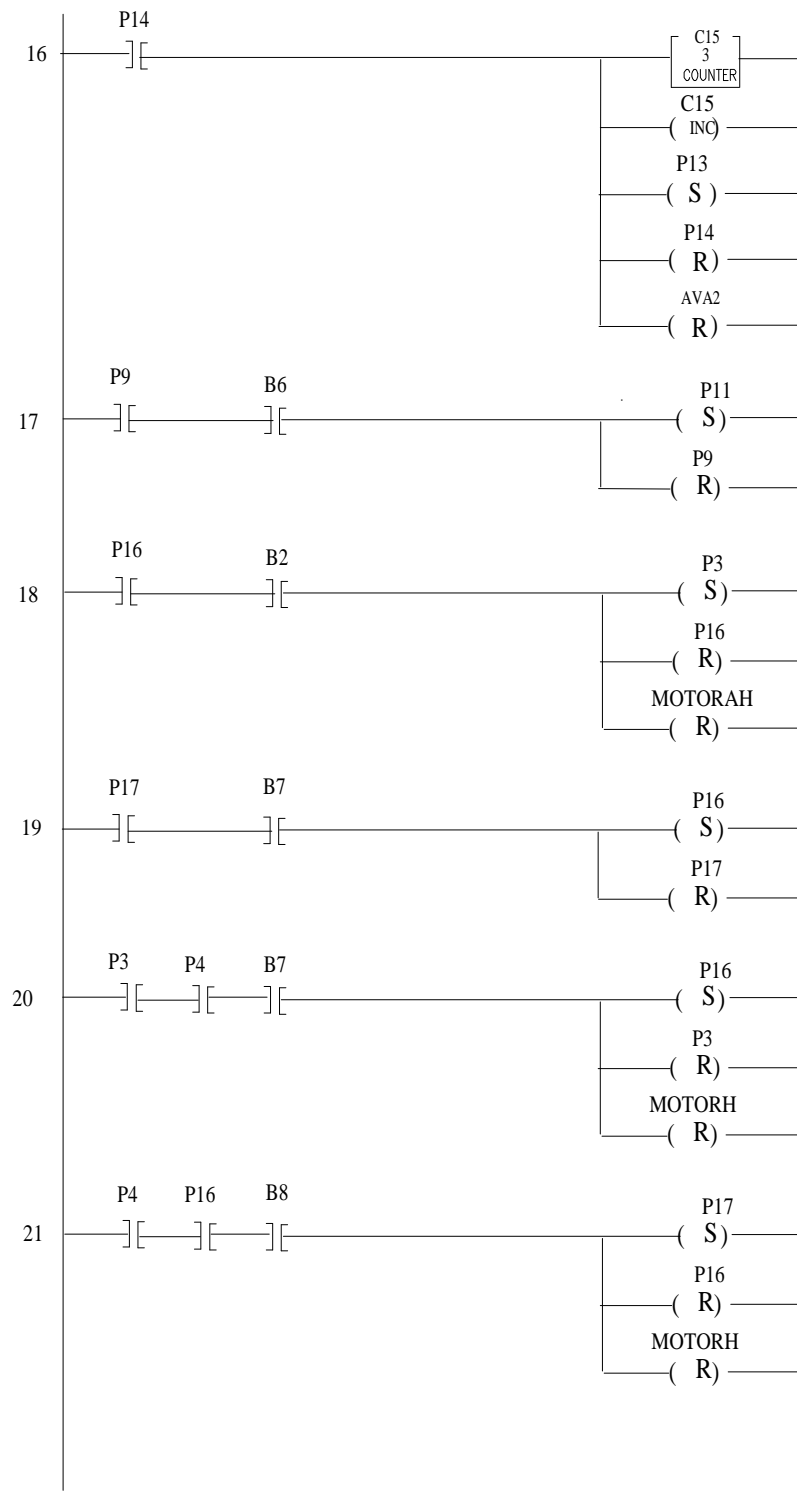


Figura 0.41: Parte 5 do diagrama Ladder do supervisor.

# Bibliografia

- [1] FESTO DIDACTIC. *Técnicas de Automação Industrial*. Partes 1, 2 e 3, Brasil, 1991.
- [2] B. Coriat. *Automação programável, novas formas e conceitos da produção*. H. SCHMITZ & R.Q. Carvalho, (1988). Automação, Competitividade e trabalho: a experiência internacional. Ed. Hucitec, São Paulo., 1988.
- [3] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [4] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
- [5] J.L. Peterson. *Petri Net Theory and Modelling of Systems*. Prentice Hall, 1981.
- [6] A.A. Desrochers and R.Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems: Modelling, Control, and Performance Analysis*. IEEE Press, 1995.
- [7] P.E. Miyagi. *Controle Programável: Fundamentos do Controle de Sistemas a Eventos Discretos*. Edgard Blücher Editora Ltda., 1996.
- [8] C.C. de Moraes and P.L. Castrucci. *Engenharia de Automação Industrial*. LTC Editora, 2001.
- [9] X.R. Cao and Y.C. Ho. Models of discrete event dynamic systems. *IEEE Control System Magazine*, 10(4):69–76, 1990.
- [10] R. Milner. *A Calculus of Communicating Systems*. Springer Verlag, New York, USA, 1980.
- [11] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot. A linear system theoretic view of discrete event process and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control*, 30(3):210–220, 1985.

- [12] E.M.M. Costa. *Contribuição ao Uso da Lógica Temporal na Especificação de Comportamentos de Sistemas a Eventos Discretos*. Dissertação de Mestrado. Universidade Federal da Paraíba, Campus II, Campina Grande, PB, Brasil, Outubro 1997.
- [13] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, USA, 1979.
- [14] P.J.G. Ramadge and W.M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, May 1987.
- [15] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
- [16] L.E. Holloway and B.H. Krogh. Synthesis of feedback control logic for a class of controlled petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, 1990.
- [17] S.R. Sreenivas and B.H. Krogh. On petri net models of infinite state supervisors. *IEEE Transactions on Automatic Control*, 37(2):818–823, 1992.
- [18] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman and Hall, 1993.
- [19] J.M. Proth and X. Xie. *Petri Nets: A Tool for Design and Management of Manufacturing Systems*. John Wiley - Sons Ltd, 1996.
- [20] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of petri net methods for controlled discrete event systems. *Discrete Event Systems: Theory and Applications*, pages 131–190, 1997.
- [21] A. Papelis and T.L. Casavant. Specification and analysis of parallel/distributed software and systems by petri nets with transition enabling functions. *IEEE Transactions on Software Engineering*, 18(3):252–261, 1992.
- [22] G.C. Barroso. *Uma Nova Abordagem para a Síntese de Supervisores de Sistemas a Eventos Discretos*. Tese de Doutorado. Universidade Federal da Paraíba, Campus II, Campina Grande, PB, Brasil, 1996.
- [23] M.C. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers, 1993.

- [24] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, 1994.
- [25] S. Gaubert, M. Akian, G. Cohen, R. Nikoukhah, and J.P. Quadrat. Linear systems in  $(\max, +)$  algebra. In *Proc. Of the 29th Conference on Decision and Control*, 1990.
- [26] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, (126):183–235, 1994.
- [27] S. Gaubert. Performance evaluation of  $(\max, +)$  automata. *IEEE Transactions on Automatic Control*, 40(12):2014–2025, December 1995.
- [28] E.M.M. Costa. *Síntese de Supervisores de Sistemas a Eventos Discretos Temporizados e Não Temporizados*. Tese de Doutorado. Universidade Federal da Paraíba - UFPB - Campus II, Campina Grande, Paraíba, Brasil., Novembro de 2001.
- [29] E.M.M. Costa and A.M.N. Lima. Utilizando dióides na síntese do controlador de sistemas a eventos discretos temporizados. *Anais do Congresso Brasileiro de Automática - CBA2002.*, 2002.
- [30] G. Cohen, S. Gaubert, and J.P. Quadrat. Algebraic system analysis of timed petri nets. In Cambridge University Press J. Gunawardena Ed., editor, *Idempotency - Collection of the Isaac Newton Institute*, 1995.
- [31] L. Libeaut. *Sur l'utilisation des Dioïdes pour la Commande des Systèmes a Événements Discrets*. PhD thesis, École Doctorale Sciences pour L'Ingenieur de Nantes, 1996.
- [32] V.P. Silva. *Uma Abordagem para a Síntese de Supervisores de Sistemas a Eventos Discretos a partir do Modelo Temporizado*. Dissertação de Mestrado. Universidade Federal da Paraíba, Campus II, Campina Grande, PB, Brasil, 1999.
- [33] B.H. Krogh. Controlled petri nets and maximally permissive feedback logic. In *25th Annual Allerton Conference*, pages 317–326. University of Illinois Urbana, USA, September 1987.
- [34] R.S. Sreenivas. On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modeled by controlled petri nets. *IEEE Transactions on Automatic Control*, 42(7):928–945, 1997.
- [35] K. Jensen and G. Rozenberg. *High Level Petri Nets Theory and Applications*. Springer Verlag, 1991.



- [36] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 1 : Basic Concepts. Springer-Verlag, 1992.
- [37] J.O. Moody and P.J. Antsaklis. *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers, USA, 1998.
- [38] J.O. Moody, M.V. Iordache, and P.J. Antsaklis. *Automated Synthesis of Liveness Enforcing Supervisors Using Petri Nets*. Technical Report of the ISIS Group at the University of Notre Dame, 2001.
- [39] E.A. Lima. *Uma Contribuição ao Estudo e ao Projeto de Controle Supervisório de Sistemas a Eventos Discretos Baseado em Invariantes de Lugar de Redes de Petri*. Dissertação de Mestrado, Universidade Federal da Bahia - UFBA, 2002.
- [40] E.J. Lima II. *Uma Metodologia para a Implementação Através de CLPs de Controle Supervisório de Células de Manufatura Utilizando Redes de Petri*. Dissertação de Mestrado, Universidade Federal da Bahia - UFBA, 2002.
- [41] J-L. Chirn and D.C. McFarlane. Petri nets based design of ladder logic diagrams. *Control 2000, Cambridge, UK*, September 2000.
- [42] M. Uzam, A.H. Jones, and N. Ajlouni. Conversion of petri nets controllers for manufacturing systems into ladder logic diagrams. *Intelligent Machinery Division, Research Institute for Design, Manufacture and Marketing, University of Salford, UK*, 1996.
- [43] M. Uzam. *Petri-Net-Based Supervisory Control of Discrete Event Systems and their Ladder Logic Diagram Implementation*. PhD thesis, University of Salford, UK, 1998.
- [44] G. Frey and L. Litz. Formal methods in PLC programming. *0-7803-6583-6/00*, pages 2431–2436–IEEE, 2000.
- [45] G. Frey and L. Litz. Correctness analysis of petri net based logic controllers. *Proc. of the American Control Conference, Chicago, Illinois*, pages 3165–3166, June 2000.
- [46] G. Frey. Automatic implementation of petri net based control algorithms on PLC. *Proc. of the American Control Conference, Chicago, Illinois*, pages 2819–2823, June 2000.
- [47] A.H. Jones, M. Uzam, and N. Ajlouni. Design of discrete event control systems for programmable logic controllers using t-timed petri nets. *Proc. of the 1996 IEEE Int. Symp. on Comp.-Aided Control System Design*, pages 212–217, September 1996.

- [48] R. Hilal and P. Ladet. Synchronous petri nets: Formalisation and interpretation. *Laboratoire d'Automatique de Grenoble, ENSIEG/INPG*, 1994.
- [49] FESTO Didatic Brasil. *E - 311 - Introdução a Controladores Lógicos Programáveis*. FESTO Didatic - Brasil, março de 1991.
- [50] FESTO. *Festo Software-Tools - Ladder Diagram FPC-100 - User Manual FST 100*. FESTO KG.
- [51] G.B.S. Góes. *Uma Abordagem para a Síntese de Supervisores de Sistemas a Eventos Discretos Utilizando Redes de Petri Síncronas*. Dissertação de Mestrado, DEE/EP/UFBA, 2003.