TATIANA RENATA GARCIA

CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS: UMA ABORDAGEM POR MODELO CONDIÇÃO/EVENTO

FLORIANÓPOLIS
2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS: UMA ABORDAGEM POR MODELO CONDIÇÃO/EVENTO

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

TATIANA RENATA GARCIA

Florianópolis, março de 2002.

CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS: UMA ABORDAGEM POR MODELO CONDIÇÃO/EVENTO

Tatiana Renata Garcia

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica, Área de Concentração em *Controle, Automação e Informática Industrial*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Prof. José Eduardo Ribeiro Cury, Dr. d'Etat
Orientador

Prof. Edson Roberto de Pieri, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. José Eduardo Ribeiro Cury, Dr. d'Etat
Presidente

Prof. Eduardo Camponogara, Ph.D.

Prof. Victor Juliano De Negri, Dr. Eng.

Dedico este trabalho ao meu pai, que infelizmente partiu durante o seu desenvolvimento.

AGRADECIMENTOS

Em primeiro lugar quero agradecer a minha família, por todo apoio concedido para alcançar este objetivo.

Um agradecimento especial ao Rafael, meu amor, que sempre está presente nos momentos alegres e tristes de minha vida.

Sinceros agradecimentos ao meu orientador José Eduardo Cury, por todos conselhos e ensinamentos. Além dele, algumas pessoas foram fundamentais para elaboração deste trabalho, os colegas Antônio Eduardo Carrilho da Cunha e André Leal, que contribuíram com sugestões e correções.

A todos amigos do LCMI, como Cássia, Lu, Silvia, Michelle, Cris, Karina, César, e muitos outros que não citarei aqui mas que merecem meus sinceros agradecimentos pelos bons momentos e pela amizade e companheirismo.

A CAPES pelo financiamento econômico.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS: UMA ABORDAGEM POR MODELO CONDIÇÃO/EVENTO

Tatiana Renata Garcia

Março/2002

Orientador: José Eduardo Ribeiro Cury
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: sistemas a eventos discretos, sistemas condição/evento, controle supervisório, síntese de supervisores
Número de Páginas: xiii + 72

O presente trabalho visa utilizar os sistemas condição/evento (sistemas C/E) para resolver um problema de síntese de supervisores para sistemas a eventos discretos. Os sistemas C/E definem claramente sinais de entrada e saída, mostrando-se eficientes para modelar sistemas cujos controladores são implementados em CLPs. Uma metodologia de modelagem dos sistemas baseada nos sistemas C/E é proposta. Nesta metodologia o sistema é dividido em subsistemas, e para cada um deles é encontrado um modelo C/E. Partindo dos subsistemas encontra-se o modelo C/E da planta livre do sistema, cujos elementos (sinais de entrada e saída, estados) são analisados vetorialmente, cada elemento do vetor sendo referente a um subsistema. Operações para obter a planta livre são definidas, assim como algoritmos que foram desenvolvidos e implementados. Após obter o modelo C/E da planta livre um supervisor C/E para controlá-la é definido. O problema de supervisão pode ser abordado do ponto de vista da teoria de controle supervisório para SEDs, onde a planta é modelada como um gerador de eventos, pela definição de um problema equivalente. Mostra-se que solucionando o problema de supervisão do ponto de vista da teoria de controle supervisório para SEDs, ou seja, obtendo um supervisor chamado aqui de supervisor SED é possível obter um supervisor C/E equivalente e resolver o problema proposto.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

SUPERVISORY CONTROL OF DISCRETE-EVENT SYSTEMS: A CONDITION/EVENT MODEL APPROACH

Tatiana Renata Garcia

March/2002

Advisor: José Eduardo Ribeiro Cury
Area of Concentration: Control, Automation and Industrial Computing
Key words: discrete-event systems, condition/event systems, supervisory control, synthesis of supervisors
Number of Pages: xiii + 72

This work aims to use condition/event (C/E) systems to solve a problem of synthesis of supervisors for discrete-event systems. In C/E systems, input and output signals are clearly defined, making this kind of systems attractive for the modeling of systems whose controllers are implemented with programmable logic controllers (PLCs). We propose a methodology for system modeling that is based on C/E systems. Using this methodology, we divide a system into subsystems, and for each of these subsystems we develop a C/E model. From these partial models we find a C/E model that represents the plant's free, uncontrolled behavior. The elements—input and output signals, states—of this model are analyzed vectorially, where each subsystem represents an element of the vector. We define operations and algorithms for obtaining the system's free behavior; these algorithms have been successfully implemented. After obtaining a C/E model for the system's free behavior, we define a C/E supervisor that can control it. To solve this supervision problem we can use the theory of supervisory control of DES, where the plant is modeled as an event generator, by defining an equivalent problem. We show that, by solving this supervision problem from a supervisory control of DES standpoint, that is, obtaining what we call here a DES supervisor, it is possible to obtain an equivalent C/E supervisor that correctly solves the proposed problem.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O controle de sistemas a eventos discretos (SEDs) é uma área onde a pesquisa é intensa, devido a grande gama de aplicações possíveis. Sistemas de manufatura, sistemas de tráfego, protocolos de comunicação, sistemas de gerência de banco de dados, sistemas operacionais, entre outros, são exemplos de algumas aplicações.

Um SED pode ser definido como um sistema dinâmico a estado discreto que evolui conforme a ocorrência assíncrona de eventos físicos. Exemplos de eventos podem ser a chegada ou saída de uma peça em uma célula de manufatura, início ou fim de operação e a indicação de quebra de uma máquina.

A teoria de controle supervisório foi originalmente apresentada por Ramadge e Wonham [12, 13], e estabelece que os eventos que afetam o comportamento de um sistema a eventos discretos são inibidos por um agente de controle, segundo regras estabelecidas através de uma estrutura de controle. Nesta teoria o sistema é modelado como um gerador espontâneo de eventos denominado planta, e o agente de controle, chamado supervisor, deve fazer com que a planta siga um comportamento desejado.

Um fato relevante é o de que o conjunto de eventos é considerado como gerado pela planta controlada, e o controlador age de forma permissiva e não como uma unidade que gera eventos.

Em muitos casos, no entanto, como em controle de sistemas de dinâmica híbrida, onde um supervisor discreto seleciona dinâmicas de um sistema contínuo [7, 8], ou em sistemas onde se considera um nível de abstração mais baixo na modelagem do problema e onde o agente de controle atua diretamente na geração de comandos (por exemplo, quando este é implementado por um CLP) [1, 6, 11], pode ser conveniente considerar uma abordagem onde eventos seriam associados a comandos gerados pelo agente de controle ou a sinais de sensores, definindo o comportamento do sistema controlado. Neste caso, considera-se a possibilidade de modelar o sistema de forma a que a associação de comandos e respostas respectivamente a entradas e saídas apareça naturalmente.

1.1 Motivação e Objetivos

Um controlador lógico programável (CLP) é similar a um microcomputador, especializado para problemas de controle. Os CLPs são utilizados por serem confiáveis e robustos. Hoje em dia eles são largamente utilizados nas indústrias, e os sistemas de manufatura são um exemplo.

O funcionamento de um CLP pode ser resumido da seguinte maneira: em um ciclo de varredura analisa-se as entradas, atualiza-se o estado interno, calcula-se as saídas, e por fim coloca-se as saídas no valor desejado. As saídas do CLP podem ser interpretadas como comandos a um sistema a ser controlado, e as entradas do CLP são sinais de resposta a estes comandos. Estes sinais de resposta são saídas do sistema a ser controlado.

Os sistemas condição/evento apresentados em [14] se mostram bastante interessantes para tratar os problemas citados anteriormente, pois definem claramente sinais de entrada e saída. Os sistemas condição/evento fornecem uma forma de se definir sistemas em tempo contínuo como a interconexão de subsistemas com sinais de entrada e saída que assumem valores discretos. Os sistemas C/E definem claramente sinais de entrada e saída, mostrando-se eficientes para modelar sistemas cujos controladores são implementados por CLPs. Este fato motivou o desenvolvimento deste trabalho, que tem como objetivo utilizar os sistemas condição/evento para resolver um problema de síntese de supervisores.

Para resolver este problema, uma metodologia de modelagem dos sistemas baseada nos sistemas C/E é proposta. Nesta metodologia o sistema é dividido em subsistemas, e para cada um deles é encontrado um modelo C/E. Partindo dos subsistemas encontra-se a planta livre do sistema, cujos elementos (sinais de entrada e saída, estados) são analisados vetorialmente, onde cada elemento do vetor é referente a um subsistema. Operações para obter a planta livre foram definidas, assim como algoritmos que foram desenvolvidos e implementados.

O próximo passo para resolver o problema proposto é definir um supervisor C/E. O problema também pode ser abordado do ponto de vista da teoria de controle supervisório para SEDs, definindo um supervisor chamado supervisor SED para o sistema C/E. A equivalência dos supervisores é um caminho para resolver o problema de supervisão.

1.2 Estrutura da Dissertação

O capítulo 2 apresenta os conceitos básicos da teoria de linguagens e autômatos. Além disso, apresenta resumidamente a teoria de controle supervisório de Ramadge e Wonham.

No capítulo 3 os sistemas condição/evento são apresentados. Uma extensão destes sistemas, os sistemas condição/evento booleanos também são introduzidos.

O capítulo 4 apresenta uma metodologia de modelagem de sistemas, baseados nos sistemas condição/evento. Toda a metodologia é desenvolvida através de um exemplo.

No capítulo 5 o controle supervisório de sistemas C/E é explorado. Novamente um exemplo é apresentado.

No capítulo 6 um exemplo de um sistema mais complexo é apresentado.

O capítulo 7 apresenta as conclusões globais sobre o trabalho, além de sugerir trabalhos futuros.

Por fim, o apêndice A traz informações sobre ferramentas computacionais: o pacote Grail e os algoritmos implementados neste trabalho.

Capítulo 2

Conceitos Básicos

Neste capítulo serão apresentados conceitos básicos sobre linguagens e autômatos necessários para o entendimento de conceitos definidos ao longo do trabalho. Além disto, será brevemente apresentado, o modelo clássico de Ramadge-Wonham para o controle supervisório de sistemas a eventos discretos (também chamados SEDs). Este capítulo está bastante resumido pois o assunto já foi bastante detalhado em dissertações e teses anteriores, desenvolvidas no LCMI (Laboratório de Controle e Microinformática), tais como [4, 7, 15, 17].

2.1 Linguagens e Autômatos

Maiores detalhes sobre esta seção podem ser obtidos em [2, 16].

Uma linguagem L definida sobre um alfabeto Σ, é um conjunto de cadeias formadas por eventos pertencentes a Σ. O conjunto finito de todas as possíveis cadeias compostas com elementos de Σ é denotado Σ^*; e, estando neste conjunto a cadeia vazia, denotada por ε, denota-se $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Assim, uma linguagem sobre Σ é sempre um subconjunto (não necessariamente próprio) de Σ^*. Em particular $\emptyset$, Σ e Σ^* são linguagens.

Se $t\ u\ v = s$, com $t, u, v \in \Sigma^*$, então:

- t é chamado prefixo de s;
- u é chamada uma subcadeia de s;
- v é chamado sufixo de s.

Determinadas operações podem ser executadas sobre linguagens. Algumas são usuais, como as operações sobre conjuntos, mas aqui serão apresentadas três outras operações.

1. *Concatenação*: Sejam duas linguagens $L_1, L_2 \subseteq \Sigma^*$, então

$$L_1 L_2 := \{s \in \Sigma^* : (s = s_1 s_2) \text{ onde } (s_1 \in L_1) \ e \ (s_2 \in L_2)\}$$

Em palavras, uma cadeia está em $L_1 L_2$ se ela pode ser escrita como a concatenação de uma cadeia de L_1 com uma cadeia de L_2.

2. *Prefixo-Fechamento*: Seja uma linguagem $L \in \Sigma^*$, então

$$\overline{L} := \{s \in \Sigma^* : \exists t \in \Sigma^* (st \in L)\}$$

Em palavras, o prefixo-fechamento de uma linguagem L, denotado por $\overline{L}$, consiste de todas as cadeias de L que são prefixos de L. Em geral, L é dita prefixo-fechada se $L = \overline{L}$. Assim, uma linguagem L é prefixo-fechada se qualquer prefixo de qualquer cadeia de L é também uma cadeia de L.

3. *Fechamento-Kleene*: Seja uma linguagem $L \subseteq \Sigma^*$, então

$$L^* := \{\varepsilon\} \cup L \cup LL \cup LLL \cup \cdots$$

Uma cadeia de L^* é formada pela concatenação de um número finito de cadeias de L, incluindo a concatenação de nenhuma cadeia, que é a cadeia vazia ε.

Um autômato determinístico de estados finitos é uma sêxtupla $G = (X, \Sigma, f, x_0, X_m)$, onde:

- X é o conjunto de estados do autômato, possivelmente infinito;
- Σ é o conjunto de símbolos (eventos) que definem o alfabeto;
- $f : X \times \Sigma \to X$ é a função de transição, possivelmente parcial, ou seja, não há necessidade da função ser definida para cada evento de Σ em cada estado de X;
- x_0 é o estado inicial do autômato;
- X_m é o conjunto de estados marcados ou finais, $X_m \subseteq X$.

Na teoria de linguagens e autômatos apresentada em [9] o autômato deveria ter função de transição completa, e a estrutura apresentada anteriormente seria na verdade um gerador, como definido em [16]. Neste trabalho será utilizado o termo autômato para a definição anterior ($G = (X, \Sigma, f, x_0, X_m)$ com f possivelmente parcial), sem causar prejuízos.

A função f pode ser estendida do domínio $X \times \Sigma$ para o domínio $X \times \Sigma^*$, trabalhando com cadeias, recursivamente, da seguinte maneira:

$$\begin{aligned} \hat{f}(x,\varepsilon) &:= x \\ \hat{f}(x,e) &:= f(x,e), \ \ e \in \Sigma \\ \hat{f}(x,se) &:= f(\hat{f}(x,s),e) \ \text{ para } s \in \Sigma^* \ \text{ e } \ e \in \Sigma. \end{aligned}$$

Uma cadeia s é reconhecida por um autômato G se $\hat{f}(x_0,s) = x$, onde $x \in X_m$.

Um autômato G está associado a duas linguagens, a linguagem gerada $L(G)$ e a linguagem marcada $L_m(G)$.

A linguagem gerada por $G = (X,\Sigma,f,x_0,X_m)$ é:

$$L(G) := \{s \in \Sigma^* : \hat{f}(x_0,s) \ \text{ é definida}\}.$$

A linguagem marcada de G é:

$$L_m(G) := \{s \in L(G) : \hat{f}(x_0,s) \in X_m\}.$$

A linguagem $L(G)$ representa todas as cadeias que podem ser seguidas no autômato, partindo do estado inicial. A linguagem $L_m(G)$ possui todas as cadeias que partindo do estado inicial chegam a um estado marcado.

2.2 Controle Supervisório de SEDs

A teoria clássica de controle supervisório para modelar e controlar sistemas a eventos discretos foi formulada inicialmente por Ramadge-Wonham [12] e está expressa em termos da observação e inibição de eventos, realizados por um supervisor minimamente restritivo. Outros detalhes sobre esta seção podem ser obtidos em [16] e [3].

Um sistema a eventos discretos pode ser caracterizado por um par de linguagens $(L,L_m) \subseteq \Sigma^* \times \Sigma^*$, onde Σ é um conjunto de símbolos que representam os eventos. A linguagem $L \subseteq \Sigma^*$ é prefixo-fechada e representa todas as cadeias de eventos que o sistema pode gerar, enquanto a linguagem $L_m \subseteq L$ representa todas as cadeias que identificam tarefas completas do sistema.

Classicamente, define-se uma estrutura de controle Γ para o SED (L,L_m) pelo particionamento de Σ em $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$, onde Σ_c é o conjunto de eventos controláveis, que podem ser inibidos de ocorrer no sistema, e Σ_u é o conjunto de eventos não controláveis, que não podem ser inibidos de ocorrer no sistema.

Uma opção de controle $\gamma \in \Gamma$ aplicada ao sistema, contém o conjunto ativo de eventos que está habilitado a ocorrer no sistema. Com a definição da controlabilidade de eventos, a estrutura de controle tem a seguinte forma

$$\Gamma = \{\gamma \in 2^{\Sigma} : \gamma \supseteq \Sigma_u\}$$

onde a condição $\gamma \supseteq \Sigma_u$ indica simplesmente que os eventos não controláveis não podem ser desabilitados.

Um supervisor é definido pelo mapa $f : L \to \Gamma$ que aplica uma entrada de controle γ a um sistema a eventos discretos S como função da seqüência de eventos gerados pelo sistema. O sistema a eventos discretos S controlado por f é denotado por f/S. A linguagem $L(f/S) \subseteq \Sigma^*$ representa o comportamento gerado do sistema, e é definida como segue:

1. $\varepsilon \in L(f/S)$, e
2. $s\sigma \in L(f/S) \iff s \in L(f/S) \wedge s\sigma \in L \wedge \sigma \in f(s)$

A linguagem que representa o comportamento marcado do sistema sob supervisão é $L_m(f/S) = L(f/S) \cap L_m(S)$, e representa a parte de $L_m(S)$ que sobrevive sob ação de controle. Diz-se que f é um supervisor não bloqueante para S se $L(f/S) = \overline{L_m(f/S)}$.

O problema de controle supervisório é apresentado a seguir:

Problema 2.1 (Problema de Controle Supervisório (PCS)) *Dado um sistema $S = (L, L_m)$ com estrutura de controle Γ, definidos sobre o conjunto de eventos Σ; e especificações definidas por $A \subseteq E \subseteq \Sigma^*$; encontre um supervisor não bloqueante f para S tal que*

$$A \subseteq L_m(f/S) \subseteq E$$

O conceito de controlabilidade de linguagens é fundamental para a solução do problema enunciado.

Definição 2.1 (Controlabilidade) *Dado um sistema a eventos discretos $S = (L, L_m)$ com estrutura de controle Γ e uma linguagem $E \subseteq L$, E é controlável com respeito a S se,*

$$\overline{E}\Sigma_u \cap L \subseteq \overline{E}$$

Além de definir a controlabilidade de linguagens, é necessário também definir o conceito de L_m-fechamento.

Definição 2.2 (L_m**-Fechamento)** *Sejam E e L_m duas linguagens onde $E \subseteq L_m \subseteq \Sigma^*$, a linguagem E é dita fechada em relação a L_m ou L_m-fechada se*

$$E = \overline{E} \cap L_m$$

O seguinte resultado fornece a solução para o problema de controle supervisório definido anteriormente:

Proposição 2.1 *Dado um sistema $S = (L, L_m)$ com estrutura de controle Γ, definidos sobre o conjunto de eventos Σ e $K \subseteq L_m(S)$; existe um supervisor não bloqueante f para S tal que*

$$L_m(f/S) = K$$

se e somente se K for controlável e L_m-fechada.

Para uma linguagem $E \subseteq \Sigma^*$, seja $CF(E)$ o conjunto de linguagens controláveis e L_m-fechadas contidas em E. Pela propriedade de fechamento para união da estrutura de controle Γ e pela definição de controlabilidade, o conjunto $CF(E)$ é não vazio, fechado para união e possui um elemento supremo sup$CF(E)$. Uma solução para o PCS é fornecida através da seguinte proposição:

Proposição 2.2 *O problema de controle supervisório possui solução se e somente se $A \subseteq supCF(E)$.*

O comportamento minimamente restritivo é representado por sup$CF(E)$ e um supervisor ótimo é obtido quando $L_m(f/S) = supCF(E)$.

2.3 Conclusão

Este capítulo apresentou diversos conceitos sobre linguagens e autômatos, assim como o modelo clássico de controle supervisório de Ramadge-Wonham.

O comportamento de um sistema a eventos discretos pode ser modelado por uma linguagem, e esta linguagem pode ser representada por um autômato.

A teoria clássica de controle supervisório está expressa em termos de observação e inibição de eventos. O problema de controle supervisório visa encontrar um supervisor minimamente restritivo, e para isto é fundamental o conceito de controlabilidade.

Capítulo 3

Sistemas Condição/Evento

Este capítulo visa apresentar os sistemas C/E, os quais servirão de base para o trabalho. São apresentados também os sistemas C/E Booleanos, os quais são uma extensão dos sistemas C/E, onde os sinais são vetores de elementos booleanos.

3.1 Introdução aos Sistemas C/E

Em [14] foi apresentado um formalismo de modelagem para sistemas a eventos discretos, os sistemas condição/evento, ou sistemas C/E. Neste formalismo existem duas classes de sinais, os sinais condição e os sinais evento. Um sinal condição é um sinal constante por partes que toma valores de um conjunto finito de condições. Um sinal evento é nulo, exceto em pontos discretos do tempo, quando assume valores de um conjunto finito de eventos.

Um sistema C/E pode ser representado como mostra o diagrama da figura 3.1, onde $u(t), y(t)$ representam sinais condição e $v(t), z(t)$ representam sinais eventos.

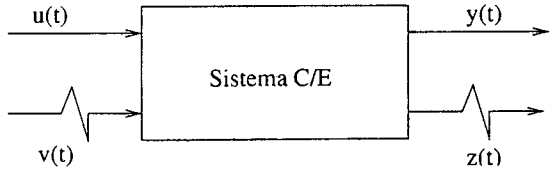

Figura 3.1: Sistema Condição/Evento

Quando a dinâmica interna do sistema C/E é modelada por uma realização de estado discreto (definida a seguir), transições de estado podem ser inibidas ou habilitadas pelo sinal de entrada condição e forçadas pelo sinal entrada evento. O valor do sinal de saída condição é uma função do estado do sistema e do sinal de entrada condição, enquanto que o evento de saída é uma função da transição de estado e da entrada evento. A trajetória de estado é um sinal condição.

A principal motivação para utilizar uma abordagem com sistemas C/E é poder modelar sistemas de tempo contínuo como SEDs, pois os sinais de entrada e saída assumem valores discretos. Em aplicações de controle discreto este sinais caracterizam o fluxo de informações entre subsistemas. Valores discretos dos sensores, por exemplo, são sinais condição, enquanto interruptores e *clocks* são naturalmente modelados como eventos. Sinais condição e evento são também utilizados para representar as interações de causalidade entre componentes físicos de um sistema de controle discreto. Sistemas C/E permitem modelar sistemas interconectados onde transições de sinais condição e evento ocorrem assincronamente. Sinais de entrada e saída apropriados podem garantir especificações de sincronização quando os subsistemas são conectados.

3.1.1 Definições

Esta seção tem o intuito de apresentar os sistemas C/E. Todas as definições, teoremas e corolários foram baseados em [14].

Sejam X um conjunto não vazio e enumerável, η o símbolo nulo, $\eta \notin X$, e o conjunto $X^+ = X \cup \{\eta\}$ (esta definição vale para todos conjuntos). Algumas definições devem ser feitas sobre os sistemas C/E.

Definição 3.1 (Sinal condição) *A função $x : [0,\infty) \to X$ é um sinal condição sobre X em $[0,\infty)$ se $x(\cdot)$ é um sinal contínuo à direita com limites à esquerda.*

Definição 3.2 (Sinal evento) *A função $x : [0,\infty) \to X^+$ é um sinal evento sobre X em $[0,\infty)$ se $x(0) = \eta$, e, para qualquer intervalo de tempo $[t_1,t_2] \in [0,\infty)$, o conjunto $\{t \in [t_1,t_2] : x(t) \neq \eta\}$ for finito.*

A figura 3.2 ilustra os sinais evento e condição.

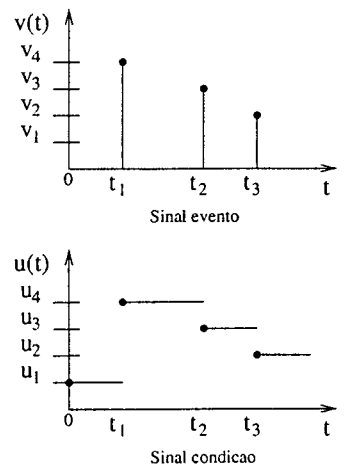

Figura 3.2: Sinais evento e condição

O conjunto de todos os sinais condição ou evento sobre X em $[0,\infty)$ é referenciado por $\mathcal{X}$ (X caligráfico).

Definição 3.3 (Sistema C/E) *Sejam U,V,Y e Z conjuntos finitos e disjuntos. Um sistema condição/evento $\mathcal{S}$ em $[0,\infty)$ com entradas condição $\mathcal{U}$, entradas evento $\mathcal{V}$, saídas condição $\mathcal{Y}$ e saídas evento $\mathcal{Z}$, é um mapa $\mathcal{S} : \mathcal{U} \times \mathcal{V} \to 2^{\mathcal{Y} \times \mathcal{Z}}$ tal que, para cada entrada $(u(\cdot), v(\cdot)) \in \mathcal{U} \times \mathcal{V}$ existe pelo menos uma saída $(y(\cdot), z(\cdot)) \in \mathcal{Y} \times \mathcal{Z}$ tal que $(y(\cdot), z(\cdot)) \in \mathcal{S}(u(\cdot), v(\cdot))$.*

Descrevendo em palavras, um sistema C/E identifica um conjunto de sinais de saída $(y(\cdot), z(\cdot))$ com cada sinal de entrada $(u(\cdot), v(\cdot))$. Logo, os sistemas C/E são em geral não-determinísticos.

Algumas propriedades são definidas para estabelecer classes úteis de sistemas C/E. São elas as propriedades de causalidade, invariância na escala de tempo e espontaneidade.

- Um sistema C/E é dito **causal** quando dadas duas entradas iguais até certo instante de tempo, assegura-se que há saídas (para as respectivas entradas) que também são iguais até o mesmo instante.
- Um sistema C/E possui **invariância na escala de tempo** quando classes de equivalência de pares admissíveis de sinais de entrada e saída do sistema podem ser identificadas, independentemente da escala de tempo.
- Um sistema C/E é dito **espontâneo** quando os sinais de entrada condição apenas podem inibir opções de sinais de saída evento. A espontaneidade expressa que mudanças nas entradas condição não forçam mudanças nas saídas evento do sistema, e sim habilitam possíveis saídas num dado momento.

Um sistema C/E pode ser representado através de um modelo de estado discreto. Mas para definir um modelo de estado discreto são necessárias algumas definições preliminares.

Definição 3.4 (Sinal de entrada síncrono) *Um sinal de entrada $(u(\cdot), v(\cdot)) \in \mathcal{U} \times \mathcal{V}$ é dito sinal de entrada síncrono quando para todo $t \in [0,\infty), u(t^-) \neq u(t) \Rightarrow v(t) \neq \eta$.*

Em um sinal de entrada síncrono, descontinuidades no sinal condição ocorrem somente quando há descontinuidades no sinal evento.

Definição 3.5 (Sistema C/E em tempo discreto) *Um sistema C/E $\mathcal{S}$ é um sistema C/E em tempo discreto se o conjunto de entrada evento é não vazio e se para qualquer sinal de entrada síncrono $(u(\cdot), v(\cdot)) \in \mathcal{U} \times \mathcal{V}$ e qualquer sinal de saída $(y(\cdot), z(\cdot)) \in \mathcal{S}(u(\cdot), v(\cdot))$ a seguinte condição é satisfeita: para qualquer $t \in [0,\infty), y(t^-) \neq y(t)$ ou $z(t) \neq \eta$ implica $v(t) \neq \eta$.*

Resumidamente, em um sistema C/E em tempo discreto a alteração do valor da saída condição e a existência de saída evento só ocorrem quando o sinal de entrada evento for não nulo.

Os sistemas tratados neste trabalho não possuem um sinal de entrada síncrono, mas o sinal de entrada é sincronizado com a saída através de um supervisor.

Definição 3.6 (Sistema C/E determinístico) *Um sistema C/E é dito determinístico se para cada entrada* $(u(\cdot),v(\cdot))$ *existe apenas uma saída* $(y(\cdot),z(\cdot)) = S(u(\cdot),v(\cdot))$.

Definição 3.7 (Modelo C/E) *Sejam os conjuntos finitos* U,V,Y,Z, *um modelo C/E* $\mathcal{M}$ *é um conjunto de equações de modo que para cada entrada* $(u(\cdot),v(\cdot)) \in \mathcal{U} \times \mathcal{V}$ *exista pelo menos uma saída* $(y(\cdot),z(\cdot)) \in \mathcal{Y} \times \mathcal{Z}$ *que satisfaça as equações para* $\mathcal{M}$.

Um modelo condição/evento $\mathcal{M}$ implicitamente define um sistema C/E, denotado por $S_{\mathcal{M}}$. Se para um sistema C/E S, $S = S_{\mathcal{M}}$, diz-se que $\mathcal{M}$ é uma realização de S.

Definição 3.8 (Modelo de estado discreto) *Um modelo de estado discreto para um sistema C/E* S : $\mathcal{U} \times \mathcal{V} \rightarrow 2^{\mathcal{Y} \times \mathcal{Z}}$ *é uma quíntupla* (X,f,g,h,x_0) *onde:*

- X *é um conjunto enumerável de estados;*
- $f : X \times U \times V^+ \rightarrow 2^X - \emptyset$ *é a função de transição de estados, satisfazendo* $x \in f(x,u,\eta)$ $\forall x \in X$ *e* $\forall u \in U$. *Ou seja, o sistema deve estar apto a permanecer em qualquer estado se não houverem sinais de entrada evento;*
- $g : X \times U \rightarrow Y$ *é a função de saída condição;*
- $h : X \times X \times V^+ \rightarrow Z^+$ *é a função de saída evento, satisfazendo* $h(x,x,\eta) = \eta$ *para todo* $x \in X$. *Ou seja, a saída evento só será diferente de* η *se houver uma transição de estado;*
- x_0 *é o estado inicial.*

Um sinal de entrada $(u(\cdot),v(\cdot))$, um sinal de saída $(y(\cdot),z(\cdot))$ e uma trajetória de estados $x(\cdot) \in \mathcal{X}$ satisfazem o modelo de estado discreto quando $x(0) = x_0$ e as seguintes equações são satisfeitas para $t > 0$:

$$\begin{aligned} x(t) &\in f(x(t^-),u(t^-),v(t)) \\ y(t) &= g(x(t),u(t)) \\ z(t) &= h(x(t^-),x(t),v(t)) \end{aligned}$$

Dado um modelo estado discreto Σ, S_Σ denota o sistema C/E definido por este modelo. Alternativamente, se um sistema C/E S é igual a S_Σ para algum modelo de estado discreto Σ, entao Σ será referenciado como uma realização de estado discreto para o sistema C/E S.

Teorema 3.1 *Se um sistema condição/evento S tem realização de estado discreto, ele é causal, invariante em escala de tempo e espontâneo.*

É possível associar um comportamento lógico aos sistemas C/E, já que eles possuem sinais de entrada e saída assumindo valores discretos. Dado um espaço de sinais condição (ou evento) $\mathcal{A}$ sobre um conjunto A definem-se:

Definição 3.9 (Projeção discreta) *A projeção discreta $Pr : \mathcal{A} \rightarrow A^{\#}$, onde $A^{\#} = A^* \cup A^w$ e A^* e A^w denotam respectivamente todas as seqüências não vazias de comprimento finito e todas seqüências de comprimento infinito com elementos em A. Dado o sinal $a(\cdot) \in \mathcal{A}$, $Pr[a(\cdot)]$ registra numa palavra, de comprimento possivelmente infinito, os valores que $a(\cdot)$ assume em A nos instantes t_0 e os pontos de descontinuidade, respeitando a ordem temporal.*

Definição 3.10 (Operador prefixo pre) *O operador prefixo $pre : A^{\#} \rightarrow 2^{A^*}$ relaciona uma palavra possivelmente infinita $a \in A^{\#}$ com $pre(a) \subseteq A^*$, o conjunto de todos os seus prefixos de comprimento finito.*

Definição 3.11 (Operador linguagem) *O operador linguagem $\mathcal{L}(\mathcal{A}) = \{pre[Pr(a(\cdot))] : a(\cdot) \in \mathcal{A}\}$ gera o prefixo-fechamento de todas as cadeias de comprimento finito formadas por registro dos valores dos sinais em $\mathcal{A}$ em seus momentos de descontinuidade.*

Definição 3.12 (Linguagem de um sistema C/E) *A linguagem (de cadeias de comprimento finito) de um sistema C/E S, $\mathcal{L}(S) \subseteq (U \times V^+ \times Y \times Z^+)^*$ é definida da seguinte maneira:*

$$\mathcal{L}(S) = \{pre[Pr(u(\cdot), v(\cdot), y(\cdot), z(\cdot))] : (y(\cdot), z(\cdot)) \in S(u(\cdot), v(\cdot))\}$$

Para explorar o relacionamento entre sistemas C/E com realizações de estado discreto e linguagens C/E, foram definidos modelos baseados em autômatos.

Definição 3.13 (Autômato de um sistema C/E) *Um autômato G é uma 5-tupla $G = (Q, \delta, A, q_0, Q_m)$, onde Q é o conjunto finito de estados, A é o alfabeto finito, $\delta : Q \times A \cup \{\varepsilon\} \rightarrow 2^Q$ é a função de transição (não determinística), $\varepsilon \notin A$ é a palavra vazia, q_0 é o estado inicial, e Q_m é o conjunto de estados marcados.*

Neste trabalho trabalha-se apenas com linguagens marcadas, logo em todos os autômatos $Q = Q_m$.

Definição 3.14 (Gerador) *Dado um alfabeto A e uma linguagem $L \subseteq A^*$, um autômato $G = (Q, \delta, A, q_0, Q_m)$, com $Q = Q_m$, é dito ser um gerador para L se $w \in L$ se e somente se w é reconhecida por G. $L(G)$ denota a (única) linguagem gerada por G.*

Uma palavra é reconhecida por um gerador quando ela alcança um estado marcado.

A relação entre sistemas C/E com realização de estado discreto e linguagens C/E com geradores (finitos) é dada pelo seguinte teorema:

Teorema 3.2 *Se um sistema C/E $\mathcal{S}$ tem uma realização de estado discreto, então existe um gerador para $\mathcal{L}(\mathcal{S})$, a linguagem de $\mathcal{S}$.*

Definição 3.15 (Obtenção de um autômato) *Suponha que $\Sigma = (X, f, g, h, x_0)$ é uma realização de estado discreto para o sistema C/E $\mathcal{S}$. O autômato $G = (Q, \delta, A, q_0, Q_m)$, tal que $L(G) = \mathcal{L}(\mathcal{S})$, é definido da seguinte maneira:*

$$\begin{aligned} Q &= X \times U \cup \{x_0\} \\ A &= U \times V^+ \times Y \times Z^+ \\ q_0 &= x_0 \end{aligned}$$

e a função $\delta : Q \times A \to 2^Q$ é definida como segue:

$$\delta(q_0, \sigma) = \begin{cases} (x_0, u) & \text{se } \sigma = (u, \eta, g(x_0, u), \eta), \\ \emptyset & \text{caso contrário.} \end{cases}$$

e

$$\delta((x,u),\sigma) = \{(x',u') \neq (x,u) : \; x' \in f(x,u,v'), y' = g(x',u'), \text{ e } z' = h(x,x',v') \\ \text{onde } \sigma = (u',v',y',z')\}$$

para $(x,u) \in X \times U$.

3.2 Sistemas C/E Interconectados

Uma das principais motivações para introduzir SEDs com estrutura explícita de sinais de entrada e saída é permitir definir com clareza e consistência sistemas interconectados. Em [14] foram apresentadas duas modalidades de conexão: conexão em cascata ou conexão com realimentação. O objetivo é definir uma realização de estado discreto para o sistema, tendo como base realizações de estado discreto dos subsistemas e a definição natural dos fluxos de sinais entre estes subsistemas.

3.2.1 Sistemas em Cascata

Considere dois sistemas C/E, $\mathcal{S}_1$ e $\mathcal{S}_2$, com realizações de estado discreto, Σ_1 e Σ_2, respectivamente. A conexão em cascata destes dois sistemas consiste na conexão das saídas de $\mathcal{S}_1$ nas entradas de $\mathcal{S}_2$, como mostra a figura 3.3. É possível perceber que $Y_1 = U_2$ e que $Z_1 = V_2$.

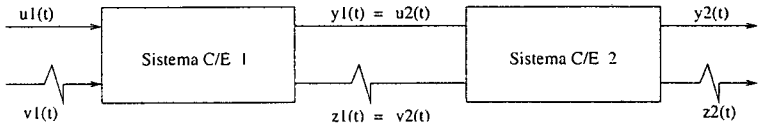

Figura 3.3: Sistema C/E em cascata

O sistema resultante é $\mathcal{S}_{\mathcal{S}_1 \to \mathcal{S}_2} : \mathcal{U} \times \mathcal{V} \to 2^{\mathcal{Y} \times \mathcal{Z}}$. Em particular, se $(y_2(\cdot), z_2(\cdot)) \in \mathcal{S}_{\mathcal{S}_1 \to \mathcal{S}_2}\ (u_1(\cdot), v_1(\cdot))$, então existem trajetórias de estado $x_1(\cdot)$ e $x_2(\cdot)$ satisfazendo as equações:

$$\begin{aligned} x_1(t) &\in f_1(x_1(t^-), u_1(t^-), v_1(t)) \\ x_2(t) &\in f_2(x_2(t^-), g_1(x_1(t^-), u_1(t^-)), h_1(x_1(t^-), x_1(t), v_1(t))) \\ y_2(t) &= g_2(x_2(t), g_1(x_1(t), u_1(t))) \\ z_2(t) &= h_2(x_2(t^-), x_2(t), h_1(x_1(t^-), x_1(t), v_1(t))) \end{aligned}$$

onde os sinais $y_1(\cdot) = u_2(\cdot)$ e $z_1(\cdot) = v_2(\cdot)$ são eliminados.

Sendo o espaço de estados $X = X_1 \times X_2$, com estado inicial $x_0 = \{x_1, x_2\}$ é possível definir funções de saída $g : X \times U_1 \to Y_2$ e $h : X \times X \times V_1^+ \to Z_2$ para realização de estado discreto do sistema $\mathcal{S}_{\mathcal{S}_1 \to \mathcal{S}_2}$ como:

$$g(x, u_1) = g_2(x_2, g_1(x_1, u_1)) \text{ para } x = (x_1, x_2) \in X, u_1 \in U_1$$

e

$$h(x, x', v_1) = h_2(x_2, x_2', h_1(x_1, x_1', v_1)) \text{ para } x = (x_1, x_2), x' = (x_1', x_2') \in X, v_1 \in V_1.$$

A função de transição $f(\cdot, \cdot, \cdot)$ é definida da seguinte maneira:

$$f(x, u_1, v_1) = \{x' = (x_1', x_2') \in X : x_1' \in f_1(x_1, u_1, v_1) \text{ e } x_2' \in f_2(x_2, g_1(x_1, u_1), h_1(x_1, x_1', v_1))\}.$$

Transições no segundo componente do estado em X_2 só são permitidas quando são consistentes com transições do primeiro componente do estado em X_1, gerando eventos forçados para S_2 via

$v_2(t) = z_1(t)$. Nesta abordagem as transições de estado em um sistema ($\mathcal{S}_1$) são independentes das transições de estado do outro sistema ($\mathcal{S}_2$), mas as transições no segundo sistema podem ser dependentes de eventos do primeiro sistema.

3.2.2 Sistemas com Realimentação

Em um sistema com realimentação, as saídas condição e evento são ligadas nas entradas condição e evento, ou seja, os conjuntos de sinais de entrada e saída são os mesmos, $U = Y$ e $V = Z$, como mostra a figura 3.4.

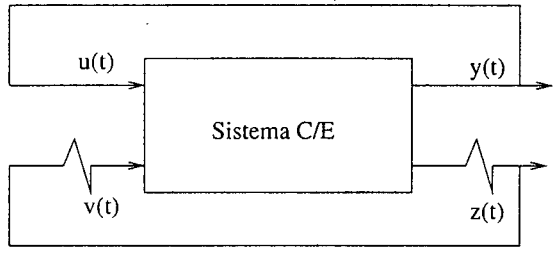

Figura 3.4: Sistema C/E com realimentação

Se o sistema $\mathcal{S}$ possui realização de estado discreto (X, f, g, h, x_0), após a conexão com realimentação, as seguintes equações são satisfeitas quando $x(0) = x_0$

$$\begin{aligned} x(t) &\in f(x(t^-), y(t^-), z(t)) \\ y(t) &= g(x(t), y(t)) \\ z(t) &= h(x(t^-), x(t), z(t)) \end{aligned}$$

A conexão com realimentação de um sistema C/E identifica o subconjunto de sinais de saída para o sistema C/E em malha aberta que pode ser gerado autonomamente sob realimentação.

Dado o estado inicial x_0 para uma realização de estado discreto Σ de um sistema C/E em malha aberta $\mathcal{S}$, é definido o conjunto de valores admissíveis para $y(t_0)$ ($t_0 \in \Re$ é o tempo inicial) como $Y_0(t_0) = \{y \in Y : y = g(x_0, y)\}$. Se $Y_0(x_0)$ é vazio, não há solução para as equações da realização de estado discreto Σ_F que definem o sistema C/E com realimentação ($\mathcal{S}_{\Sigma_F}$), e o conjunto de sinais de saída é vazio.

O modelo de estado discreto $\Sigma' = (X', f', g', h', x_0')$ para o sistema $\mathcal{S}_{\Sigma_F}$ é definido como $X' = X \times Y \times Z^+$ e $x_0' = (x_0, y(t_0), \eta)$. Sendo $x_1', x_2' \in X'$ estados arbitrários com $x_i' = (x_i, y_i, z_i) \in X \times Y \times Z^+$ para $i = 1, 2$, as funções da saída são definidas da seguinte maneira:

$$g'(x_1') = y_1$$

e

$$h'(x'_1,x'_2) = \begin{cases} z_2 & \text{se } z_2 \neq z_1, \\ \eta & \text{caso contrário} \end{cases}$$

e a função de transição $f' : X' \to 2^{x'}$ é:

$$f'(x'_1) = \begin{cases} \Gamma(x'_1) & \text{se } z_1 = \eta \text{ e } \Gamma(x'_1) \neq \emptyset \\ \{x'_1\} & \text{se } z_1 = \eta \text{ e } \Gamma(x'_1) = \emptyset \\ \{x'_1,(x_1,y_1,\eta)\} & \text{se } z_1 \neq \eta \end{cases}$$

onde

$$\Gamma(x'_1) = \{x'_2 \in X' : y_2 = g(x_2,y_2), z_2 = h(x_1,x_2,z_2), \text{ e } x_2 \in f(x_1,y_1,z_2)\}.$$

Agora Σ' é um modelo de estado discreto, sendo que $f'(x')$ é não vazio e $h'(x',x') = \eta$ para todo $x' \in X'$.

3.2.3 Extensões dos Sistemas Condição/Evento

Em [10] é introduzida uma representação computacional de SEDs em tempo contínuo, que possibilita a modelagem e a análise de sistemas complexos usando ferramentas de software existentes para análise booleana. A representação é baseada em multi-entradas e multi-saídas, uma extensão dos sistemas C/E apresentados em [14], e é chamada de **Sistemas Condição/Evento Booleanos.**

Considere um sistema C/E com representação de estado discreto da seguinte forma:

$$\begin{aligned} x(t) &\in f(x(t^-),u(t^-),v(t)) \\ y(t) &= g(x(t),u(t)) \\ z(t) &= h(x(t^-),x(t),v(t)) \end{aligned}$$

Nos sistemas C/E booleanos assume-se que os sinais x,u,v,y,z são vetores de dimensões n_x, n_u, n_v, n_y, n_z, respectivamente, e os componentes dos sinais condição e evento são valores binários, onde o valor (ou vetor) 0 (zero) corresponde ao valor nulo para o sinal evento.

Em um modelo de um sistema C/E, a função f de transição de estado satisfaz $x \in f(x,u,0)$ para todo estado x e entrada condição u, refletindo o fato que a transição de estado pode ser forçada somente

pelo sinal evento. A função saída evento h satisfaz $0 = h(x,x,0)$ para todo estado x, indicando que uma saída evento não pode ocorrer sem uma transição de estado ou uma entrada evento.

Em [10] são definidas duas operações para construção de modelos de sistemas C/E interconectados que possuem multi-entradas e multi-saídas (como num vetor): **empilhar** e **conectar**. Estas operações não são válidas exclusivamente para sistemas onde os sinais possuem valores booleanos, e sim para sistemas onde os sinais assumem qualquer valor.

Estas operações são semelhantes às operações de cascata e realimentação apresentadas em [14], entretanto permitem sistemas com multi-entradas e multi-saidas. Além disso são mais flexíveis pois a operação conectar permite realizar a conexão em cascata ou com realimentação.

Definição 3.16 (Empilhar) *Dadas realizações de estado discreto $(X_i, f_i, g_i, h_i, x_{0_i})$ de sistemas C/E $S^1, \cdots, S^n$ a operação concorrente de n subsistemas pode ser representada como um sistema C/E denotado por $S = EMPILHAR(S^1, \cdots, S^n)$. O sistema C/E S tem sinais de entrada $u = [u^1 : u^2 : \cdots : u^n]$ e $v = [v^1 : v^2 : \cdots : v^n]$, onde $[\,.\,:\,.\,:\cdots:\,.\,]$ denota a concatenação dos argumentos dos vetores. Os sinais de saída são definidos similarmente. A realização de estado discreto é dada em termos do vetor de estado $x = [x^1 : x^2 : \cdots : x^n]$ e pelas equações f, g, h, sendo que $f = [f^1 : f^2 : \cdots : f^n]$, g e h são definidadas similarmente.*

Definição 3.17 (Conectar) *Sistemas interconectados são definidos pela conexão dos sinais de saída nos sinais de entrada usando a operação $CONECTAR(S, YUMAP, ZVMAP)$, onde $YUMAP$ é uma lista de pares ordenados $YUMAP = [(j_1, l_1), (j_2, l_2), \cdots] \subset \{1, \cdots, n_y\} \times \{1, \cdots, n_u\}$ especificando as conexões dos sinais saída condição com os sinais entrada condição, e $ZVMAP$ é uma lista de pares ordenados $ZVMAP = [(k_1, m_1), (k_2, m_2), \cdots] \subset \{1, \cdots, n_z\} \times \{1, \cdots, n_v\}$ especificando as conexões dos sinais saída evento com os sinais entrada evento.*

Os segundos componentes dos pares, l ou m, devem ser distintos de maneira que somente um componente do sinal de saída possa ser conectado a um dado componente do sinal de entrada.

A operação empilhar permite modelar a operação concorrente de todos os subsistemas, independente de suas relações. Suponha um sistema S formado pelos subsistemas S_1, S_2, S_3. Aplicar a operação empilhar sobre eles retorna um sistema onde eles operam concorrentemente. A figura 3.5 apresenta os subsistemas do sistema S, enquanto que a figura 3.6 apresenta o sistema S após a operação empilhar.

Após aplicar a operação empilhar pode ser desejável conectar saídas de alguns subsistemas em entradas de outros, obtendo a conexão em cascata, ou realimentar alguns subsistemas, obtendo a conexão com realimentação. Estas operações são possíveis através da operação conectar. A figura 3.7 mostra como o sistema se comporta quando conectamos o sistema S_1 e S_2 em cascata.

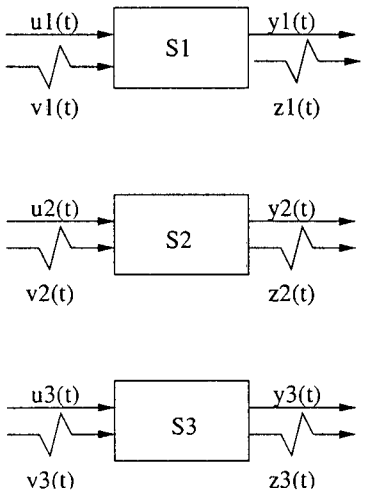

Figura 3.5: Subsistemas do sistema S

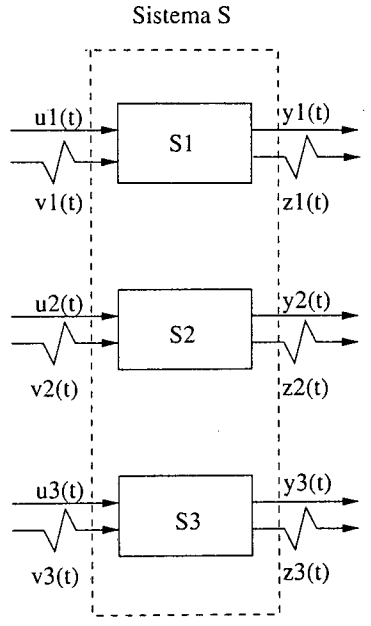

Figura 3.6: Sistema S após operação empilhar

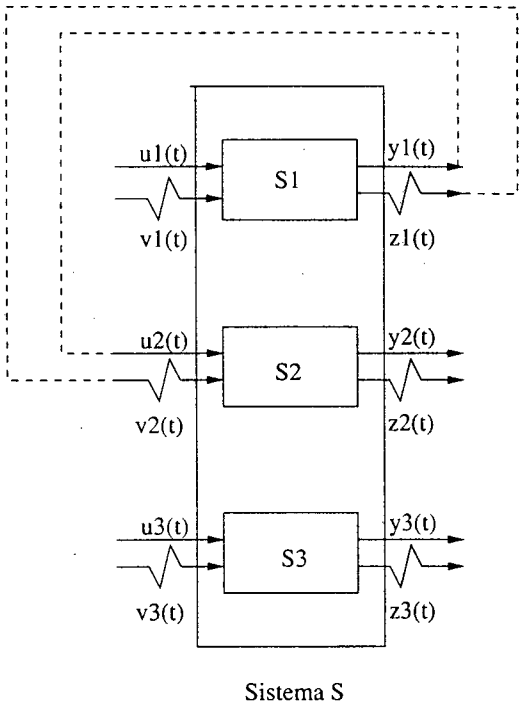

Figura 3.7: Sistema S após operação conectar

Detalhes sobre a representação dos sistemas C/E booleanos podem ser obtidas em [10].

3.3 Conclusão

Em [14] foi apresentado um formalismo de modelagem de SEDs chamado sistemas C/E. Estes sistemas possuem duas classes de sinais: os sinais condição e os sinais eventos. Um sistema C/E pode ser representado por uma realização de estado discreto, e um autômato para representar a linguagem que define o funcionamento do sistema foi definido. Sistemas C/E podem ser conectados em cascata ou por realimentação. Neste capítulo foram apresentadas as definições e teoremas sobre sistemas C/E.

Esta abordagem será utilizada neste trabalho para representar uma determinada classe de sistemas, onde visa-se associar comandos e respostas aos sinais de entrada e saída.

Foram apresentados também os sistemas C/E booleanos, cuja representação do sistema através de vetores servirá de base para a metodologia de modelagem proposta neste trabalho, além das operações empilhar e conectar.

Capítulo 4

Modelagem

Neste capítulo será apresentada uma metodologia de modelagem para sistemas C/E. O objetivo é encontrar o modelo da planta livre de um sistema C/E.

Um pequeno exemplo é introduzido, e através dele a metodologia é desenvolvida. Ao final um pequeno resumo sobre a metodologia é apresentado. O resumo visa destacar os passos a serem seguidos para encontrar o modelo da planta livre para qualquer sistema.

4.1 Exemplo

Suponha uma célula de manufatura, como aquela mostrada na figura 4.1. Esta célula está localizada no Laboratório de Automação Industrial (LAI) do Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina. O sistema é composto por diversas estações: de espera, de inspeção, de rejeição e de transferência, entre outros elementos. Cada *pallet* inserido na célula é submetido à uma seqüência de atividades, passando por todas estações e sensores, podendo ser aceito ou rejeitado ao final da seqüência.

Neste capítulo será modelada a estação de espera desta célula, que está representada na figura 4.2.

Os elementos da estação de espera são:

- Os atuadores A_x e A_y;
- Os sensores S_i e S_k.

Os atuadores podem ser estendidos ou recolhidos. Caso o atuador A_x esteja estendido, um novo *pallet* deve esperar para entrar no buffer. O sensor S_i é sensibilizado sempre que um *pallet* chega

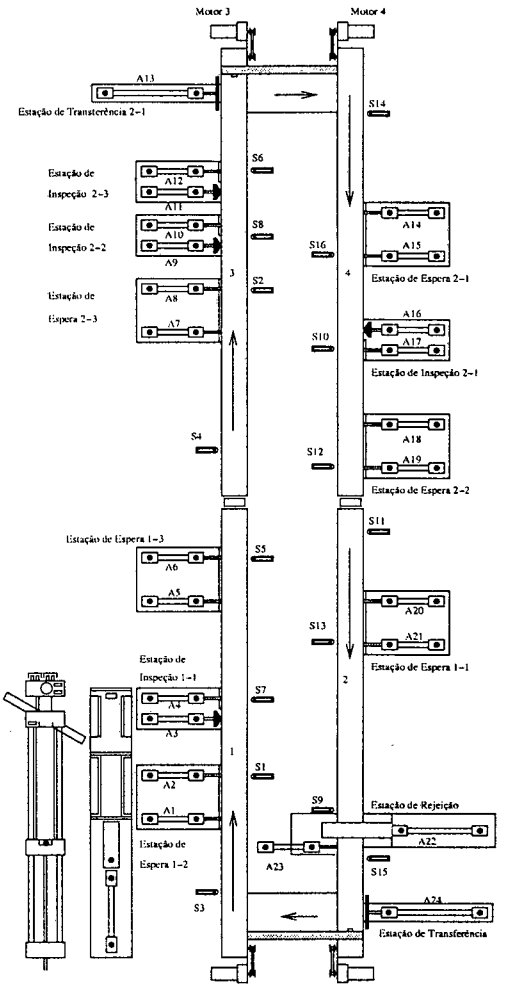

Figura 4.1: Célula de manufatura

na estação. Neste trabalho o sensor S_k não será modelado para simplificar a obtenção manual do supervisor.

Este sistema será utilizado como exemplo para apresentar uma metodologia de modelagem e controle supervisório para sistemas C/E.

4.2 Modelagem da Estação de Espera

O exemplo descrito anteriormente pode ser modelado como um sistema C/E. Os elementos da estação de espera relacionam-se entre si, e o esquema da figura 4.3 mostra como isto ocorre do ponto de vista de condições e eventos.

O sistema possui como entrada sinais condição (representados pela letra u) e como saída sinais evento (representados pela letra v). As saídas condição (como as do sensor S_i) não são relevantes para

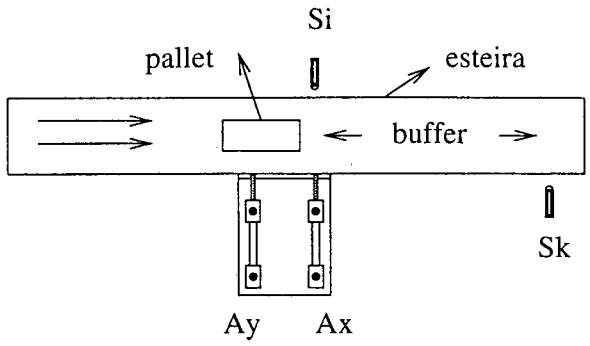

Figura 4.2: Estação de espera

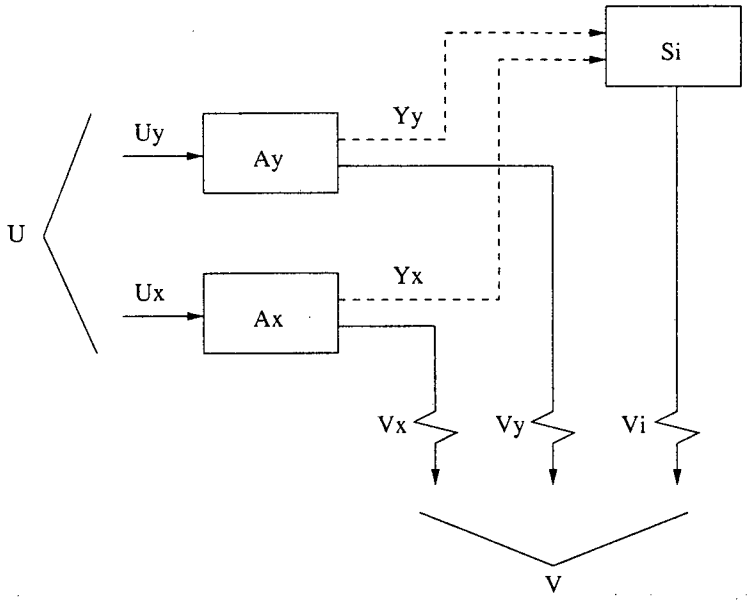

Figura 4.3: Relacionamento C/E dos elementos do sistema da estação de espera

o sistema global, e são representadas pela letra y. Um caso mais geral poderia ter como entrada e saída ambos sinais condição e evento. Cada elemento do sistema é modelado separadamente.

A seguir será apresentado o modelo C/E de cada um dos elementos e da planta livre.

4.2.1 Atuador A_x

O atuador A_x (sistema C/E $\mathcal{A}_x$) pode ser modelado como uma realização de estado discreto $A_x = (X_x, f_x, h_x, g_x, x_x)$ com os conjuntos U_x, Y_x, V_x, onde:

- Os índices est, rec se relacionam com a ação de estender e recolher, respectivamente, o atuador;
- X_x é o conjunto de estados $X_x = \{x_x^{est}, x_x^{rec}\}$;

- U_x é o conjunto de entradas condição $U_x = \{u_x^{est}, u_x^{rec}\}$;
- Y_x é o conjunto de saídas condição $Y_x = \{y_x^{est}, y_x^{rec}\}$;
- V_x é o conjunto de saídas evento $V_x = \{v_x^{est}, v_x^{rec}\}$;
- f_x é a função de transição $f_x : X_x \times U_x \rightarrow X_x$ e está definida na tabela 4.1;

x_x^-	u_x^-	x_x
x_x^{est}	u_x^{est}	x_x^{est}
x_x^{est}	u_x^{rec}	x_x^{est}
x_x^{est}	u_x^{rec}	x_x^{rec}
x_x^{rec}	u_x^{rec}	x_x^{rec}
x_x^{rec}	u_x^{est}	x_x^{rec}
x_x^{rec}	u_x^{est}	x_x^{est}

Tabela 4.1: Função de transição do atuador A_x

- h_x é a função saída evento $h_x : X_x \times X_x \rightarrow V_x^+$ e está definida na tabela 4.2;

x_x^-	x_x	v_x
x_x^{est}	x_x^{est}	η
x_x^{est}	x_x^{rec}	v_x^{rec}
x_x^{rec}	x_x^{rec}	η
x_x^{rec}	x_x^{est}	v_x^{est}

Tabela 4.2: Função saída evento do atuador A_x

- g_x é a função saída condição $g_x : X_x \rightarrow Y_x$, onde $y_x = g_x(x_x) = x_x$;
- x_x é o estado inicial, $x_x = x_x^{est}$

O atuador pode encontrar-se em dois estados distintos, estendido (x_x^{est}) ou recolhido (x_x^{rec}).

As entradas condição representam comandos para o atuador. A entrada condição u_x^{est} simboliza o comando de extensão do atuador, enquanto u_x^{rec} simboliza o comando de recolhimento do atuador. Os eventos de resposta a estes comandos são representados pelas saídas evento v_x^{est} (resposta à extensão) e v_x^{rec} (resposta ao recolhimento). As saídas condição representam os estados do atuador.

Algumas transições indicam que quando não há ocorrência de entradas evento o sistema pode permanecer no mesmo estado com qualquer condição de entrada, e neste trabalho algumas delas foram ignoradas (tabela 4.3) para simplificar o comportamento do sistema, para que a definição do mesmo fique mais legível. Entretanto, a planta livre não permite transições onde há mudanças do sinal condição em mais de um subsistema (o que permitiria ao supervisor, a ser definido, aplicar dois comandos na mesma entrada).

Nos subsistemas apresentados posteriormente as funções de transição apresentadas serão as simplificadas. Para representar a permanência do sistema no mesmo estado só serão indicadas algumas entradas condição. Por exemplo, para indicar que o atuador continue estendido, apenas a transição $x_x^{est} u_x^{est} x_x^{est}$ será indicada. Ou seja, se o atuador esta estendido, e a entrada condição é a que representa o comando de extensão, o atuador continuará estendido.

x_x^-	u_x^-	x_x
x_x^{est}	u_x^{rec}	x_x^{est}
x_x^{rec}	u_x^{est}	x_x^{rec}

Tabela 4.3: Transições ignoradas da função de transição do atuador A_x

Após eliminar estas linhas da tabela, a função de transição é a apresentada na tabela 4.4.

x_x^-	u_x^-	x_x
x_x^{est}	u_x^{est}	x_x^{est}
x_x^{est}	u_x^{rec}	x_x^{rec}
x_x^{rec}	u_x^{rec}	x_x^{rec}
x_x^{rec}	u_x^{est}	x_x^{est}

Tabela 4.4: Nova função de transição do atuador A_x

Após definir o modelo C/E do atuador é possível encontrar um autômato que represente a linguagem do mesmo. O autômato é obtido como na definição 3.15 e está mostrado na figura 4.4. Para construir este autômato foram analisadas as funções das tabelas 4.4 e 4.2.

Vale ressaltar que neste trabalho todas as linguagens são prefixo fechadas, logo todos os estados dos autômatos apresentados são finais.

A notação utilizada para representar uma transição em um autômato neste trabalho é um par v/u, onde v é a saída evento e u a entrada condição.

4.2.2 Atuador A_y

O atuador A_y (sistema C/E $\mathcal{A}_y$) pode ser modelado como uma realização de estado discreto $A_y = (X_y, f_y, h_y, g_y, x_y)$ com os conjuntos U_y, Y_y, V_y, onde:

- Os índices est, rec se relacionam com a ação de estender e recolher, respectivamente, o atuador;
- X_y é o conjunto de estados $X_y = \{x_y^{rec}, x_y^{est}\}$;
- U_y é o conjunto de entradas condição $U_y = \{u_y^{rec}, u_y^{est}\}$;
- Y_y é o conjunto de saídas condição $Y_y = \{y_y^{rec}, y_y^{est}\}$;

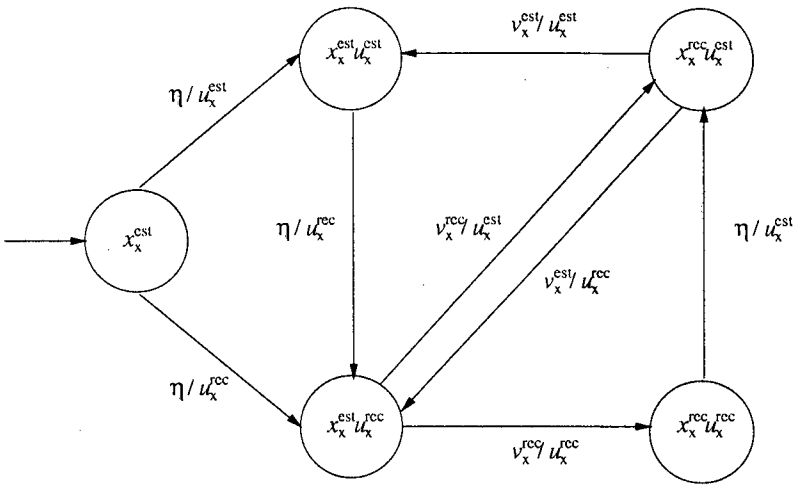

Figura 4.4: Autômato atuador A_x

- V_y é o conjunto de saídas evento $V_y = \{v_y^{rec}, v_y^{est}\}$;
- f_y é a função de transição $f_y : X_y \times U_y \rightarrow X_y$ e está definida na tabela 4.5;

x_y^-	u_y^-	x_y
x_y^{rec}	u_y^{rec}	x_y^{rec}
x_y^{rec}	u_y^{est}	x_y^{est}
x_y^{est}	u_y^{est}	x_y^{est}
x_y^{est}	u_y^{rec}	x_y^{rec}

Tabela 4.5: Função de transição do atuador A_y

- h_y é a função saída evento $h_y : X_y \times X_y \rightarrow V_y^+$ e está definida na tabela 4.6;

x_y^-	x_y	v_y
x_y^{rec}	x_y^{rec}	η
x_y^{rec}	x_y^{est}	v_y^{est}
x_y^{est}	x_y^{est}	η
x_y^{est}	x_y^{rec}	v_y^{rec}

Tabela 4.6: Função saída evento do atuador A_y

- g_y é a função saída condição $g_y : X_y \rightarrow Y_y$, onde $g_y(x_y) = x_y$;
- x_y é o estado inicial, $x_y = x_y^{rec}$

O atuador pode encontrar-se em dois estados distintos, estendido (x_y^{est}) ou recolhido (x_y^{rec}).

A entrada condição u_y^{est} simboliza o comando de extensão do atuador, enquanto u_y^{rec} o comando de recolhimento do atuador. Os eventos de resposta a estes comandos são representados pelas saídas

evento v_y^{est} (resposta à extensão) e v_y^{rec} (resposta ao recolhimento). As saídas condição representam os estados do atuador.

Após definir o modelo C/E do atuador é possível encontrar um autômato que represente a linguagem do mesmo. O autômato está ilustrado na figura 4.5.

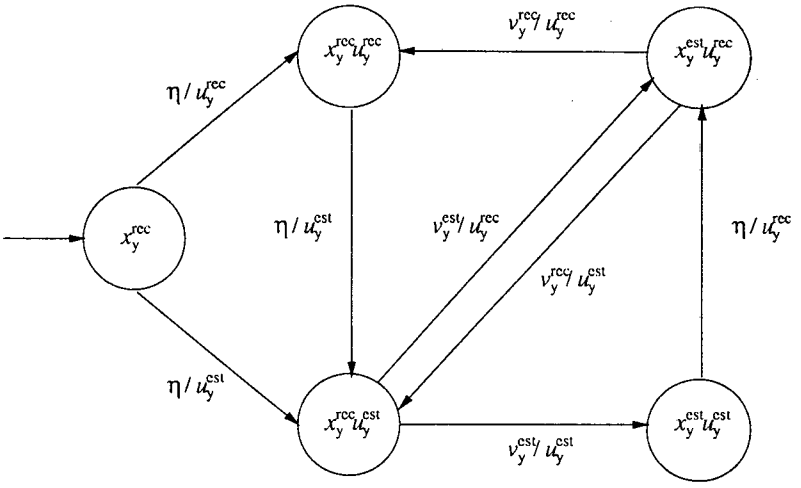

Figura 4.5: Autômato atuador A_y

4.2.3 Sensor S_i

Ao analisar o esquema de relacionamento dos elementos do sistema da figura 4.3 é possível perceber que o sensor se relaciona com os dois atuadores. As linhas pontilhadas indicam que o sensor é conectado aos atuadores através da operação de **conexão**, onde as saídas condição dos atuadores são as entradas condição do sensor.

Existem restrições físicas do sistema relacionadas com o sensor, tais como:

- O sensor S_i não pode ser sensibilizado se o atuador A_y estiver estendido;
- O sensor S_i não pode ser dessensibilizado antes do atuador A_x ser recolhido

Estas restrições podem ser tratadas na definição da função de transição do sensor S_i, impedindo transições para estados dos atuadores que violem as restrições. Ou seja, S_i só pode ser sensibilizado se o atuador A_y estiver no estado x_y^{rec} (recolhido) e dessensibilizado se o atuador A_x estiver no estado x_x^{rec} (recolhido). Isto é possível porque as entradas condição do sensor são as saídas condição dos atuadores, que por sua vez armazenam os estados dos atuadores.

Finalmente, o sensor S_i (sistema C/E S_i) pode ser modelado como uma realização de estado discreto $S_i = (X_i, f_i, h_i, g_i, x_i)$ com os conjuntos U_i, Y_i, V_i, onde:

- Os índices on, off se relacionam com a sensibilização e dessensibilização, respectivamente, do sensor;
- X_i é o conjunto de estados $X_i = \{x_i^{off}, x_i^{on}\}$;
- U_i é o conjunto de entradas condição, onde $U_i = Y_x \times Y_y$, $U_i = \{(y_x^{est}, y_y^{rec}), (y_x^{est}, y_y^{est}), (y_x^{rec}, y_y^{rec}), (y_x^{rec}, y_y^{est})\}$;
- Y_i é o conjunto de saídas condição $Y_i = \{y_i^{off}, y_i^{on}\}$;
- V_i é o conjunto de saídas evento $V_i = \{v_i^{off}, v_i^{on}\}$;
- f_i é a função de transição $f_i : X_i \times U_i \rightarrow X_i$ e está definida na tabela 4.7;

x_i^-	u_i^-	x_i
x_i^{off}	(y_x^{est}, y_y^{est})	x_i^{off}
x_i^{off}	(y_x^{rec}, y_y^{est})	x_i^{off}
x_i^{off}	(y_x^{est}, y_y^{rec})	x_i^{on}
x_i^{off}	(y_x^{rec}, y_y^{rec})	x_i^{on}
x_i^{on}	(y_x^{est}, y_y^{est})	x_i^{on}
x_i^{on}	(y_x^{est}, y_y^{rec})	x_i^{on}
x_i^{on}	(y_x^{rec}, y_y^{rec})	x_i^{off}
x_i^{on}	(y_x^{rec}, y_y^{est})	x_i^{off}

Tabela 4.7: Função de transição do sensor S_i

- h_i é a função saída evento $h_i : X_i \times X_i \rightarrow V_i^+$ e está definida na tabela 4.8;

x_i^-	x_i	v_i
x_i^{off}	x_i^{off}	η
x_i^{off}	x_i^{on}	v_i^{on}
x_i^{on}	x_i^{on}	η
x_i^{on}	x_i^{off}	v_i^{off}

Tabela 4.8: Função saída evento do sensor S_i

- g_i é a função saída condição $g_i : X_i \rightarrow Y_i$, onde $g_i(x_i) = x_i$;
- x_i é o estado inicial, $x_i = x_i^{off}$

O autômato do sensor é semelhante ao dos atuadores, e não é apresentado devido ao fato de possuir mais transições e estados que os anteriores, dificultando a visualização.

4.2.4 Planta Livre

A planta livre do sistema C/E $\mathcal{S}$ é modelada como um sistema C/E, chamado $\mathcal{P}$. A planta possui uma realização de estado discreto $P = (X_P, f_P, h_P, x_P)$ com os conjuntos U_P, V_P e os elementos do sistema funcionam concorrentemente. A sua modelagem foi baseada nos sistemas C/E booleanos [10] (mas neste trabalho os sistemas não são necessariamente booleanos), os estados e os sinais são modelados como vetores, onde cada elemento do vetor está relacionado com um elemento do sistema.

A função g e o conjunto Y não são definidos para a planta livre. As saídas condição não são relevantes para modelar a planta livre, pois apenas indicam os estados do sistema. A definição destes sinais nos modelos dos subsistemas são importantes para modelar a operação de conexão (quando necessária).

Para obter a planta livre são utilizadas as operações **empilhar** e **conectar**, definidas no capítulo anterior.

O conjunto de estados da planta livre é formado por vetores, $X_P = X_x \times X_y \times X_i$:

$$X_P = \left\{ \begin{array}{lll} [\ x_x^{est} & x_y^{rec} & x_i^{off}\ ] \\ [\ x_x^{est} & x_y^{rec} & x_i^{on}\ ] \\ [\ x_x^{est} & x_y^{est} & x_i^{off}\ ] \\ [\ x_x^{est} & x_y^{est} & x_i^{on}\ ] \\ [\ x_x^{rec} & x_y^{rec} & x_i^{off}\ ] \\ [\ x_x^{rec} & x_y^{rec} & x_i^{on}\ ] \\ [\ x_x^{rec} & x_y^{est} & x_i^{off}\ ] \\ [\ x_x^{rec} & x_y^{est} & x_i^{on}\ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ x_0\ ] \\ [\ x_1\ ] \\ [\ x_2\ ] \\ [\ x_3\ ] \\ [\ x_4\ ] \\ [\ x_5\ ] \\ [\ x_6\ ] \\ [\ x_7\ ] \end{array} \right\}$$

O conjunto de entradas condição da planta também é um conjunto de vetores, $U_P = U_x \times U_y$:

$$U_P = \left\{ \begin{array}{ll} [\ u_x^{est} & u_y^{rec}\ ] \\ [\ u_x^{est} & u_y^{est}\ ] \\ [\ u_x^{rec} & u_y^{rec}\ ] \\ [\ u_x^{rec} & u_y^{est}\ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ u_1\ ] \\ [\ u_2\ ] \\ [\ u_3\ ] \\ [\ u_4\ ] \end{array} \right\}$$

Os elementos de U_P são entradas condição dos subsistemas que são entradas da planta livre. Entretanto, existem subsistemas que possuem entradas condição que não fazem parte dos elementos de U_P, pois são internas à planta. Estas entradas condição são internas por serem resultado da operação de conexão, onde saídas de alguns subsistemas são conectados a entradas de outros.

Para realizar a operação de empilhar, e obter as funções de transição e saída evento, é necessário definir um conjunto estendido de U_P, onde seus elementos contenham entradas condição de todos

subsistemas que compõem a planta. Esta definição é necessária porque a operação empilhar retorna o comportamento da planta quando todos subsistemas atuam concorrentemente, incluindo os que sofreram a operação de conexão.

O conjunto $U_{P_{est}}$ é um conjunto de vetores $U_{P_{est}} = U_x \times U_y \times U_i$:

$$U_{P_{est}} = \left\{ \begin{array}{llll} [& u_x^{est} & u_y^{rec} & (y_x^{est}, y_y^{rec}) \ ] \\ [& u_x^{est} & u_y^{rec} & (y_x^{est}, y_y^{est}) \ ] \\ [& u_x^{est} & u_y^{rec} & (y_x^{rec}, y_y^{rec}) \ ] \\ [& u_x^{est} & u_y^{rec} & (y_x^{rec}, y_y^{est}) \ ] \\ [& u_x^{est} & u_y^{est} & (y_x^{est}, y_y^{rec}) \ ] \\ [& u_x^{est} & u_y^{est} & (y_x^{est}, y_y^{est}) \ ] \\ [& u_x^{est} & u_y^{est} & (y_x^{rec}, y_y^{rec}) \ ] \\ [& u_x^{est} & u_y^{est} & (y_x^{rec}, y_y^{est}) \ ] \\ [& u_x^{rec} & u_y^{rec} & (y_x^{est}, y_y^{rec}) \ ] \\ [& u_x^{rec} & u_y^{rec} & (y_x^{est}, y_y^{est}) \ ] \\ [& u_x^{rec} & u_y^{rec} & (y_x^{rec}, y_y^{rec}) \ ] \\ [& u_x^{rec} & u_y^{rec} & (y_x^{rec}, y_y^{est}) \ ] \\ [& u_x^{rec} & u_y^{est} & (y_x^{est}, y_y^{rec}) \ ] \\ [& u_x^{rec} & u_y^{est} & (y_x^{est}, y_y^{est}) \ ] \\ [& u_x^{rec} & u_y^{est} & (y_x^{rec}, y_y^{rec}) \ ] \\ [& u_x^{rec} & u_y^{est} & (y_x^{rec}, y_y^{est}) \ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ u_{0e} \ ] \\ [\ u_{1e} \ ] \\ [\ u_{2e} \ ] \\ [\ u_{3e} \ ] \\ [\ u_{4e} \ ] \\ [\ u_{5e} \ ] \\ [\ u_{6e} \ ] \\ [\ u_{7e} \ ] \\ [\ u_{8e} \ ] \\ [\ u_{9e} \ ] \\ [\ u_{10e} \ ] \\ [\ u_{11e} \ ] \\ [\ u_{12e} \ ] \\ [\ u_{13e} \ ] \\ [\ u_{14e} \ ] \\ [\ u_{15e} \ ] \end{array} \right\}$$

Nesta modelagem foi assumido que não ocorre simultaneidade de eventos. O conjunto de saídas evento é formado por vetores de $V_x^+ \times V_y^+ \times V_i^+$. O conjunto de saídas evento válido é:

$$V_P = \left\{ \begin{array}{llll} [& v_x^{est} & \eta & \eta \ ] \\ [& v_x^{rec} & \eta & \eta \ ] \\ [& \eta & v_y^{rec} & \eta \ ] \\ [& \eta & v_y^{est} & \eta \ ] \\ [& \eta & \eta & v_i^{on} \ ] \\ [& \eta & \eta & v_i^{off} \ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ v_1 \ ] \\ [\ v_2 \ ] \\ [\ v_3 \ ] \\ [\ v_4 \ ] \\ [\ v_5 \ ] \\ [\ v_6 \ ] \end{array} \right\}$$

Através da operação empilhar é possível encontrar as funções de transição e saída evento da planta livre. Como a operação utiliza o conjunto $U_{P_{est}}$, a função de transição obtida será referenciada como $f_{P_{est}}$.

A função $f_{P_{est}}$ é obtida da seguinte forma: para cada estado x^- busca-se as transições para um estado x com os vetores entrada condição, e isto é obtido inspecionando a função f de cada elemento do sistema. Algumas transições da função f_P estão na tabela 4.9.

x_P^-	$u_{P_{est}}^-$	x_P
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{rec}, (x_x^{est}, x_y^{rec})]$	$[x_x^{est}, x_y^{rec}, x_i^{on}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{rec}, (x_x^{rec}, x_y^{rec})]$	$[x_x^{est}, x_y^{rec}, x_i^{on}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{est}, (x_x^{est}, x_y^{est})]$	$[x_x^{est}, x_y^{est}, x_i^{off}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{est}, (x_x^{rec}, x_y^{est})]$	$[x_x^{est}, x_y^{est}, x_i^{off}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{rec}, u_y^{rec}, (x_x^{est}, x_y^{est})]$	$[x_x^{rec}, x_y^{rec}, x_i^{off}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{rec}, u_y^{rec}, (x_x^{rec}, x_y^{est})]$	$[x_x^{rec}, x_y^{rec}, x_i^{off}]$

Tabela 4.9: Função de transição estendida da planta livre

Entretanto, as entradas condição internas que pertencem aos elementos de $U_{P_{est}}$ não devem aparecer explicitamente na função de transição da planta livre. É necessário filtrar em $f_{P_{est}}$ as entradas condição internas da planta livre, obtendo assim f_P, a função de transição da mesma. Algumas transições de f_P estão mostradas na tabela 4.10

x_P^-	u_P^-	x_P
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{rec}]$	$[x_x^{est}, x_y^{rec}, x_i^{on}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{est}, u_y^{est}]$	$[x_x^{est}, x_y^{est}, x_i^{off}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[u_x^{rec}, u_y^{rec}]$	$[x_x^{rec}, x_y^{rec}, x_i^{off}]$

Tabela 4.10: Função de transição da planta livre

O procedimento para encontrar a função h_P é semelhante ao da função f_P. Para cada transição do estado x^- para x verifica-se quais vetores saída evento podem ocorrer. Algumas transições da função h_P estão na tabela 4.11.

x_P^-	x_P	v_P
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[x_x^{est}, x_y^{rec}, x_i^{on}]$	$[\eta, \eta, v_i^{on}]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[x_x^{est}, x_y^{est}, x_i^{off}]$	$[\eta, v_y^{est}, \eta]$
$[x_x^{est}, x_y^{rec}, x_i^{off}]$	$[x_x^{rec}, x_y^{rec}, x_i^{off}]$	$[v_x^{rec}, \eta, \eta]$

Tabela 4.11: Função saída evento da planta livre

Vale salientar que só são válidas transições de estado que aparecem em ambas as funções, e as saídas eventos válidas são as que pertencem à V_P. Este fato pode acarretar no descarte de algumas transições da função f_P, que ocorrem quando há simultaneidade de eventos, o que não está definido em V_P.

O estado inicial é o vetor $x_P = [x_x^{est}, x_y^{rec}, x_i^{off}] \in X_P$.

Após definir o modelo de estado discreto, é possível encontrar um autômato seguindo a definição 3.15. O fato de se trabalhar com vetores não interfere no procedimento, mas agora as transições são compostas de pares v/u, onde v e u são vetores.

Em virtude do seu tamanho (excessivo) o autômato da planta livre não será mostrado aqui. Entre-

tanto, algumas observações devem ser feitas.

A planta livre do sistema é composta por vários subsistemas. Quando os subsistemas são analisados em conjunto seus sinais são sincronizados, ocorrendo transições apenas quando ocorrem eventos. Como a linguagem da planta representa apenas seus pontos de descontinuidade, transições com o evento nulo $\eta = [\eta\eta\eta]$ não aparecem no autômato. A única ocorrência do evento nulo se dá no estado inicial da planta, quando nenhuma condição está associada ao estado.

Entretanto, transições com o evento nulo η ocorrem nos autômatos que representam os subsistemas, por exemplo a transição η/u_x^{rec} e η/u_x^{est} da figura 4.4. O subsistema do atuador esta sendo analisado independentemente do resto do sistema, logo não esta sincronizado com outros subsistemas. Porém um evento irrelevante ao subsistema do atuador A_x permitiu a troca de condição do mesmo, registrando um momento de descontinuidade no sinal condição. O fato de ter ocorrido uma descontinuidade no sinal permite uma transição como a citada anteriormente.

Uma vantagem desta modelagem é a possibilidade de aplicar vários comandos à planta através de uma única entrada condição, e isto é possível devido ao uso de vetores para representar os sinais condição e evento. Apesar de ser possível aplicar vários comandos ao mesmo tempo, o modelo não permite que todos os subsistemas respondam a estes comandos através de uma única saída evento, devido ao fato de não ser permitida a simultaneidade de eventos. Esta vantagem não está presente no exemplo desenvolvido neste trabalho porque as funções de transição e saída evento dos subsistemas não estão completas, e sim simplificadas, como citado na seção 4.2.1.

4.3 Algoritmos

A partir do procedimento apresentado informalmente no decorrer do desenvolvimento do exemplo, um algoritmo para encontrar um autômato que representa a linguagem de um sistema C/E é apresentado, e é o algoritmo 4.1.

Para compreender o algoritmo 4.1 são necessárias algumas definições:

- O sistema é definido como $S = (X, f, h, g, x_0)$ com os conjuntos U, Y, V, onde X é o conjunto de estados, f é a função de transição de estados, h é a função saída evento, g é a função saída condição, x_0 é o estado inicial. Os conjuntos U, Y, V possuem as entradas condição, saídas condição e saídas evento, respectivamente. A não definição da função g e do conjunto Y não comprometem o resultado do algoritmo;
- São utilizados conjuntos para armazenar os estados do autômato (Q e Q'), e a função de transição (δ);
- Os estados $q \in Q$ e $q' \in Q'$ são pares $(x,u), x \in X, u \in U$;

- As transições são pares $v/u, v \in V, u \in U$ e são armazenadas no conjunto γ;
- η é o evento nulo.

O algoritmo pode ser dividido em duas etapas, a primeira obtém as transições do estado inicial, quando nenhuma condição u está associada ao estado x_0, e a segunda as transições dos estados com condições associadas. Para cada estado (x,u) são obtidas transições v/u, sendo que *self-loops* não são permitidos pela definição. Ao final da execução do algoritmo, o conjunto Q possui o conjunto de estados, e o conjunto δ as transições entre os estados.

Algoritmo 4.1 Algoritmo para encontrar autômato C/E

$Q \leftarrow \{q_0\}$; $Q' \leftarrow Q$; Conjuntos para armazenar os estados
$\delta = \emptyset$; $\gamma \leftarrow \emptyset$; Conjuntos para armazenar as transições de estado
Selecione e remova q de Q, $q = q_0 = x_0$
para $\forall u \in U$ **faça**
  Definição das transições do estado inicial q_0
  $\gamma \leftarrow \eta/u$
  $q' \leftarrow (q,u)$
  $\delta(q,\gamma) \leftarrow q'$
  Adicione q' a Q e Q'
fim para
enquanto $Q' \neq \emptyset$ **faça**
  Selecione e remova $q = (x,u)$ de Q'
  $v \leftarrow h(x,q)$
  para $\forall u \in U$ **faça**
    $\gamma \leftarrow v/u$
    $q' \leftarrow (x,u)$
    se $q' \neq q$ **então**
      $\delta(q,\gamma) \leftarrow q'$
      se $q' \notin Q'$ **então**
        Adicione q' a Q e Q'
      fim se
    fim se
  fim para
fim enquanto

Além do algoritmo para obter um autômato de um sistema C/E foi desenvolvido um algoritmo para realizar a operação de empilhamento de sistemas C/E, o qual é apresentado como algoritmo 4.2.

Um sistema C/E S é formado por subsistemas da seguinte forma:

$$\begin{aligned} S_1 &= (X_1, f_1, h_1, g_1, x_{0_1}) \text{ e } U_1, Y_1, V_1 \\ S_2 &= (X_2, f_2, h_2, g_2, x_{0_2}) \text{ e } U_2, Y_2, V_2 \\ \vdots &= \qquad\qquad \vdots \\ S_n &= (X_n, f_n, h_n, g_n, x_{0_n}) \text{ e } U_n, Y_n, V_n \end{aligned}$$

E o sistema formado pelos subsistemas é definido segundo a realização de estado discreto $S = (X_P, f_P, h_P, g_P, x_{0_P})$ com os conjuntos U_P, Y_P, V_P.

Para compreender o algoritmo 4.2 são necessárias algumas definições, sendo que em todas elas n é o número de sistemas.

- O conjunto de estados é $X_P = X_1 \times X_2 \times \cdots X_n$. Um elemento $x \in X_P$ pode ser interpretado como $x = [x_1, x_2, \cdots, x_n]$, onde x_n é o estado do subsistema n;
- O conjunto de entradas condição estendido é $U_{P_{est}} = U_1 \times U_2 \times \cdots U_n$. Um elemento $u \in U_{P_{est}}$ pode ser interpretado como $u = [u_1, u_2, \cdots, u_n]$, onde u_n é a entrada condição do subsistema n;
- $U_P \in U_{P_{est}}$ é o conjunto de entradas condição do sistema, onde as entradas condição internas foram filtradas;
- O conjunto de saídas condição é $Y_P = Y_1 \times Y_2 \times \cdots X_n$. Um elemento $y \in Y_P$ pode ser interpretado como $y = [y_1, y_2, \cdots, y_n]$, onde y_n é o estado do subsistema n;
- O conjunto de saídas evento V_P é um conjunto de elementos pertencentes a $V_1 \times V_2 \times \cdots V_n$, onde apenas um dos elementos de $v = [v_1, v_2, \cdots, v_n]$ é diferente de η (evento nulo);
- O estado inicial é $x_{0_P} \in X_P$ e $x_{0_P} = [x_{0_1}, x_{0_2}, \cdots, x_{0_n}]$;
- A função de transição f_P é armazenada no conjunto δ;
- A função saída evento h_P é armazenada no conjunto ψ;
- A função saída condição g_P é a função identidade $g(x_p) = x_p$, e não é retornada pelo algoritmo;
- As condições, os eventos e os estados são tratados vetorialmente no algoritmo, onde o índice $i = 1 \cdots n$ corresponde ao subsistema i.

O algoritmo pode ser resumido da seguinte forma: para cada estado x do conjunto X_P são verificadas as transições de estado com cada entrada condição u do conjunto $U_{P_{est}}$. Além disto, para cada transição de estado é verificada a saída evento. Só são transições de estado válidas aquelas onde a saída evento $v \in V_P$. Com o resultado obtem-se dois conjuntos, δ e ψ, respectivamente a função de tansição e a função saída evento.

4.4 Procedimento de Modelagem

Ao longo do capítulo foi ilustrada, através de um exemplo, uma metodologia de modelagem de sistemas. Os sistemas são modelados sob a abordagem dos sistemas C/E. Os subsistemas analisados

Algoritmo 4.2 Algoritmo para executar a operação empilhar

$\delta \leftarrow \emptyset$; $\psi \leftarrow \emptyset$; Conjuntos para armazenar a função de transição e saída evento
$aux \leftarrow \emptyset$; $v \leftarrow \emptyset$; $n =$ número de subsistemas
para $\forall x \in X_P$ **faça**
  para $\forall u \in U_{P_{est}}$ **faça**
    para i de 1 até n **faça**
      $aux[i] \leftarrow f(x[i], u[i])$
      $v[i] \leftarrow h(x[i], aux[i])$
    fim para
    se $v \in V_P$ **então**
      $\delta(x, u) \leftarrow aux$
      $\psi(x, aux) \leftarrow v$
    fim se
  fim para
fim para

possuem entradas condição, saídas condição (representam os estados do sistema) e saídas evento. A planta livre não possui saídas condição. A simultaneidade de eventos não é permitida e os sinais do sistema global (ou planta livre) são modelados como vetores, onde cada elemento do vetor possui informações sobre um subsistema.

A metodologia de modelagem pode ser resumida em alguns ítens. Para se obter um modelo C/E de um sistema deve-se seguir os seguintes passos:

1. Analisar o sistema e identificar os subsistemas do mesmo, por exemplo: dispositivos de entrada/saída, dispositivos de processamento, atuadores, sensores, entre outros;
2. Analisar o relacionamento entre os subsistemas, verificando as entradas condição da planta livre e se os subsistemas podem ser conectados;
3. Encontrar uma realização de estado discreto para cada subsistema, definindo:
   (a) O conjunto de estados X;
   (b) O conjunto de entradas condição U;
   (c) O conjunto de saídas condição Y;
   (d) O conjunto de saídas evento V;
   (e) O estado inicial x_0;
   (f) A função de transição f do subsistema;
   (g) A função saída evento h do subsistema;
   (h) A função saída condição g do subsistema, sendo que g é uma função identidade dos estados do subsistema;

4. Após encontrar uma realização de estado discreto para cada subsistema é possível encontrar o modelo global do sistema (ou planta livre), através da operação empilhar descrita no algoritmo 4.2;

5. Se alguns subsistemas foram conectados é preciso refinar o modelo encontrado através da operação empilhar, pois o conjunto de entradas condição da planta livre não deve possuir entradas condição internas do sistema;

6. Após obter o modelo C/E da planta livre é possível encontrar um autômato que a represente, para isto basta aplicar o algoritmo 4.1.

Ao longo do desenvolvimento do exemplo estes passos foram seguidos, e foram analisados pontos relevantes dos mesmos. O intuito deste resumo é guiar o desenvolvimento do modelo de um sistema.

4.5 Conclusão

Este capítulo apresentou um exemplo de um sistema modelado segundo a abordagem dos sistemas C/E.

O sistema analisado consiste em uma estação de espera de uma célula flexível de manufatura, sistema este formado por dois atuadores e um sensor. Para cada subsistema foi encontrada uma realização de estado discreto.

Aspectos sobre a modelagem foram abordados ao longo do desenvolvimento do sistema e o modelo global (ou a planta livre) do sistema foi obtida.

Ao final foram apresentados dois algoritmos, um para encontrar um autômato de um sistema C/E e outro para executar a operação empilhar, a qual permite modelar a operação concorrente de vários subsistemas. Um resumo dos passos da metodologia também foi apresentado.

A natureza dos sistemas, com transições formadas por pares v/u, possibilita a definição de uma estrutura de controle para controlá-los. O próximo capítulo vai apresentar aspectos de controle supervisório para os sistemas C/E definidos e modelados neste capítulo.

Capítulo 5

Controle Supervisório de Sistemas C/E

Este capítulo apresenta o problema da síntese de um supervisor C/E para sistemas C/E, como o modelado no capítulo anterior. O supervisor é obtido resolvendo um problema equivalente, abordado do ponto de vista da teoria de controle supervisório para SEDs.

Aspectos teóricos, como definição de uma estrutura de controle e definição de controlabilidade são apresentados. Além disso, algoritmos para obter um supervisor também são desenvolvidos. Ao final, um supervisor para o sistema modelado no capítulo anterior é encontrado.

5.1 Introdução

Uma planta $\mathcal{P}$ é um sistema C/E com realização de estado discreto $P = (X, f, h, x_0)$ e os conjuntos U e V (entradas condição e saídas evento), tal que, para cada entrada $u(\cdot) \in \mathcal{U}$ exista pelo menos uma saída $v(\cdot) \in \mathcal{V}$ tal que $v(\cdot) \in \mathcal{P}(u(\cdot))$. Visa-se encontrar um supervisor C/E $\mathcal{S}$ para supervisioná-la. A figura 5.1 mostra o relacionamento entre $\mathcal{P}$ e $\mathcal{S}$.

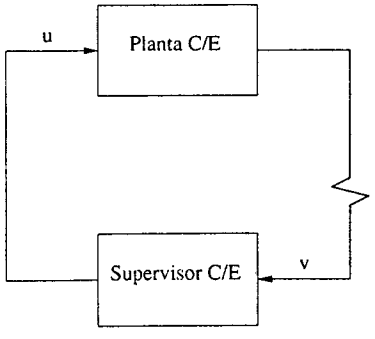

Figura 5.1: Relacionamento da planta C/E e do supervisor C/E

O sistema a ser controlado possui como entrada sinais condição, sinais eventos não são considerados. As saídas do sistema são sinais eventos, sendo que os sinais condição não são considerados para planta.

O supervisor proposto para o sistema lê as saídas evento v da planta e escolhe um u, no conjunto de entradas condição, que será habilitado na planta. O supervisor habilita somente entradas condição, entradas evento não são definidas nos sistemas analisados neste trabalho, devido a natureza dos sistemas. O supervisor pode ser implementado em um CLP, e pode-se fazer a seguinte interpretação: o CLP lê os eventos da planta e aplica comandos na mesma, onde saídas evento seriam respostas às entradas condição (comandos) aplicadas.

O supervisor desejado é um sistema C/E e pode ser definido como uma realização de estado discreto $S = (Y, \xi, \psi, y_0)$ com os conjuntos V, U (os mesmos da planta a ser controlada), onde:

- Y é o conjunto de estados;
- ξ é a funcão de transição $\xi : Y \times V^+ \to 2^Y$;
- ψ é a função saída condição $\psi : Y \to U$;
- y_0 é o estado inicial;
- V é o conjunto de entradas evento;
- U é o conjunto de saídas condição.

Um supervisor S para planta P é um mapa $\mathcal{S} : \mathcal{V} \to \mathcal{U}$, ou seja, para cada entrada $v(\cdot) \in \mathcal{V}$ existe pelo menos uma saída $u(\cdot) \in \mathcal{U}$, tal que $u(\cdot) \in \mathcal{S}(v(\cdot))$.

O sistema em malha fechada é um sistema condição/evento $\mathcal{S}/\mathcal{P} \subseteq \mathcal{V} \times \mathcal{U}$ e o modelo de estado discreto é obtido pela conexão cascata e realimentação dos modelos do supervisor $\mathcal{S}$ e da planta $\mathcal{P}$.

Comportamentos lógicos desejados são expressos em termos de V^*, ou seja, em termos de seqüências de eventos geradas pela planta.

Para expressar o comportamento em malha fechada como uma linguagem em V^* é necessário definir uma projeção.

Definição 5.1 (Projeção p_v) *A projeção $p_v : (V^+ \times U)^* \to V^*$, que apaga os símbolos η e $u \in U$ de palavras em $(V^+ \times U)^*$, é definida recursivamente como:*

$$p_v(\varepsilon) = \varepsilon$$

e para $m \in (V^+ \times U)^*$ *e* $\sigma = vu \in V \times U$

$$p_v(m \cdot \sigma) = p_v(m) \cdot v$$

A projeção p_v pode ser estendida de forma natural para linguagens em $(V^+ \times U)^*$.

A projeção inversa de uma linguagem também é definida.

Definição 5.2 (Projeção inversa p_v^{-1} de uma linguagem) *A projeção inversa* $p_v^{-1} : V^* \to (\{\eta\} \times U)(V \times U)^*$ *de uma linguagem* E *é definida como:*

$$p_v^{-1}(E) = \{s \in ((\{\eta\} \times U)(V \times U)^*) : p_v(s) \in E\}$$

A notação $vu \in V \times U$ é uma simplificação da notação $(v,u) \in V \times U$ apresentada nos capítulos anteriores.

Desta forma pode-se enunciar o problema de supervisão para sistemas C/E, como segue.

Problema 5.1 (Síntese de Supervisores para Sistemas C/E (SSCE)) *Dado o modelo C/E da planta* $\mathcal{P} : \mathcal{U} \to \mathcal{V}$ *e a especificação* $E \subseteq V^*$ *encontrar um supervisor condição/evento* $\mathcal{S} : \mathcal{V} \to \mathcal{U}$ *tal que*

$$p_v(\mathcal{L}(\mathcal{S}/\mathcal{P})) \subseteq E$$

5.2 Abordagem pela Teoria de Controle Supervisório de SEDs

O problema de síntese de supervisores para sistemas C/E pode ser traduzido para uma abordagem de controle puramente discreta, segundo a teoria de controle supervisório de SEDs.

O modelo do sistema a eventos discretos com estrutura de controle para sistemas C/E é o par $P = (L, \Gamma)$, onde L é uma linguagem prefixo-fechada e Γ é uma estrutura de controle.

A linguagem L de P é definida como:

$$L = \mathcal{L}(\mathcal{P}) \subseteq ((\{\eta\} \times U)(V \times U)^*)$$

A estrutura de controle, definida em [8], é um mapa $\Gamma : L \to 2^{2^{V^+ \times U}}$, tal que para todo $m \in L$

$$\Gamma(m) = \{\gamma \in 2^{V^+ \times U} : (\forall v \in V_L(m))(\exists u \in U_L(m,v))vu \in \gamma\}$$

Para compreender a estrutura de controle é necessário definir alguns conjuntos:

- $V_L(m)$ é o conjunto ativo de eventos em L após $m \in L$, $V_L(m) = \{v \in V^+ : (\exists u \in U) m \circ vu \in L\}$;
- $U_L(m,v)$ é o conjunto ativo de condições em L para $m \in L$ e $v \in V_L(m)$, $U_L(m,v) = \{u \in U : m \circ vu \in L\}$.

A estrutura de controle é dependente da palavra gerada pelo sistema. Como ela foi definida para impedir que, para um dado evento do conjunto ativo de eventos em m, todas as condições possíveis sejam desabilitadas, sempre será possível habilitar uma condição. A inibição de todas as condições possíveis para um dado evento levaria a uma situação sem contrapartida física, já que o supervisor inibiria todas as condições aplicáveis em resposta a um dado evento.

Um sistema C/E pode ser modelado através de dois modelos lógicos: um é a linguagem $\mathcal{L}(\mathcal{P}) \subseteq (\{\eta\} \times U)(V \times U)^*)$ e o outro é o sistema a eventos discretos $P = (L, \Gamma)$. Pode-se afirmar que os dois modelos são equivalentes quando:

1. $\mathcal{L}(\mathcal{P}) = L$;
2. $\forall w = \eta u_0 \circ v_1 u_1 \circ \cdots \circ v_k u_k \in \mathcal{L}(\mathcal{P})$ e $\alpha = vu \in V \times U$, $w \circ \alpha \in \mathcal{L}(\mathcal{P})$ se e somente se para $m = \eta u_0 \circ v_1 u_1 \circ \cdots \circ v_k u_k \in L$, $\exists \gamma \in \Gamma(m)$ de modo que $vu \in \gamma$.

Um supervisor S para o sistema $P = (L, \Gamma)$ é definido pelo mapa $S : L \to 2^{V^+ \times U}$, tal que, para $m \in L$, $S(m) \in \Gamma(m)$. Um supervisor C/E equivalente ao S pode ser definido como um sistema C/E $\mathcal{S} : \mathcal{L} \to \mathcal{V} \times \mathcal{U}$, tal que:

$$(\forall w \in \mathcal{L}(\mathcal{S}))(\forall vu \in V^+ \times U) w \circ vu \in \mathcal{L}(\mathcal{S}) \iff vu \in S(w)$$

ou seja, existe equivalência da ação de controle.

A linguagem $L(S/P) \subseteq L$ modela o comportamento em malha fechada do supervisor S para $P = (L, \Gamma)$, e é definida recursivamente como:

1. $\varepsilon \in L(S/P)$ e
2. $m \circ \sigma \in L(S/P) \iff m \in L(S/P) \wedge m \circ \sigma \in L \wedge \sigma \in S(m)$.

Neste momento é possível fazer a seguinte proposição, baseada em [8]:

Proposição 5.1 *Para o supervisor S para P e um supervisor equivalente $\mathcal{S}$ para $\mathcal{P}$, $L(S/P) = \mathcal{L}(\mathcal{S}/\mathcal{P})$.*

A proposição indica que o problema de encontrar um supervisor para o sistema C/E pode ser resolvido por um problema de controle supervisório equivalente. Este problema é definido a seguir.

Problema 5.2 (Problema de Controle Supervisório para Sistemas C/E (PCS-CE)) *Para o problema SSCE definido para planta $\mathcal{P}$ e especificação $E \subseteq V^*$, define-se um problema de controle supervisório equivalente para o modelo sistema a eventos discretos controlado para o sistema C/E $P = (L, \Gamma)$, como o de encontrar um supervisor S para P tal que*

$$p_v(L(S/P)) \subseteq E$$

Para resolver este problema deve ser considerado o conceito de controlabilidade apresentado em [8].

Definição 5.3 (Controlabilidade) *Dadas as linguagens $K \subseteq L \subseteq (\{\eta\} \times U)(V \times U)^*$, K é dita ser controlável em relação a L se*

$$V_L(m) = V_K(m) \ (\forall m \in \overline{K})$$

Logo, a linguagem K é controlável em relação a L se o conjunto ativo de eventos for igual para ambas linguagens após toda palavra do prefixo de K.

Em [8] foi definido um teorema que assegura a existência de um supervisor para um dado comportamento desejado, expresso em termos de seqüência de eventos, conforme mostrado a seguir.

Teorema 5.1 *Dados $P = (L, \Gamma)$ (modelo sistema a eventos discretos controlado para um sistema C/E) e uma linguagem $E \subseteq V^*$ (especificação) existe um supervisor S para P tal que $p_v(L(S/P)) = E$ se e somente se existir uma linguagem $K \subseteq L$ tal que K seja controlável com respeito a L, prefixo-fechada e tal que $p_v(K) = E$.*

Segundo [7], é possível provar que a estrutura de controle Γ para a linguagem L é fechada para a união para cada $m \in L$, então para $K \subseteq L$ o conjunto de linguagens prefixo-fechadas e controláveis contidas em K, $CP(K)$ é fechado para a união.

Quando o sistema pode ser modelado como um autômato de estados finitos, em [7] é proposto um algoritmo para cálculo da máxima linguagem controlável e prefixo-fechada. O algoritmo será apresentado em seção posterior.

Dados $P = (L, \Gamma)$ (o sistema) e $E \subseteq V^*$ (especificação em termos de eventos), a linguagem alvo correspondente em P é definida como $K = (p_v^{-1}(E) \cap L)$.

O seguinte teorema, baseado em [8], fornece uma solução para o problema PCS-CE:

Teorema 5.2 *O PCS-CE tem solução se e somente se* $supCP(p_v^{-1}(E) \cap L)$ *for não vazio.*

Uma solução ótima, ou minimamente restritiva, para o PCS-CE ocorre quando $L(S/P) = supCP(p_v^{-1}(E) \cap L)$.

Para resolver o problema de controle de sistemas C/E (SSCE) é necessário encontrar o supervisor $\mathcal{S}$ equivalente ao supervisor S, tal que $L(S/P) = \mathcal{L}(\mathcal{S}/\mathcal{P})$. O supervisor S (chamado de supervisor SED) é representado por um autômato $G = (Z, V \times U, \rho, z_0, Z_m)$, tal que $L(G) \| L = L(S/P)$ e $L_m(G) = L(G)$.

Teorema 5.3 *O problema SSCE tem solução se e somente se o PCS-CE equivalente tiver solução. Se S é o supervisor SED encontrado como solução do PCS-CE equivalente, o supervisor* $\mathcal{S}$ *obtido a partir de S, tal que* $L(S/P) = \mathcal{L}(\mathcal{S}/\mathcal{P})$*, é a solução do problema SSCE.*

Foi desenvolvido um algoritmo para obter o supervisor C/E equivalente $\mathcal{S}$ dado por (Y, ξ, ψ, y_0) com os conjuntos V, U. Este algoritmo também será apresentado em seção posterior.

5.3 Algoritmos

Em [7] é proposto um algoritmo para calcular a máxima linguagem controlável e prefixo-fechada de um sistema C/E modelado sob o ponto de vista da teoria de controle supervisório de SEDs, e representado através de um autômato de estados finitos.

Para compreender o algoritmo 5.1 a ser apresentado são necessárias algumas definições.

- A planta do sistema é modelada como um autômato $P = (Z, V \times U, \rho, z_0, Z_m)$, onde Z é o conjunto de estados, $V \times U$ o alfabeto, ρ a função de transição, z_0 o estado inicial e Z_m o conjunto de estados marcados. Para os sistemas modelados neste trabalho $Z = Z_m$, ou seja, todos os estados são finais;

- A linguagem alvo também é modelada como um autômato $K = (X, V \times U, \rho, x_0, X_m)$, onde $X = X_m$;

- A operação de composição síncrona de dois sistemas é dada por:

  Dados dois autômatos $G_1 = (X_1, \Sigma_1, f_1, x_{0_1}, X_{m_1})$ e $G_2 = (X_2, \Sigma_2, f_2, x_{0_2}, X_{m_2})$, define-se a composição síncrona $G_1 \| G_2$ como:

$$G_1 \| G_2 = (X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1\|2}, (x_{0_1}, x_{0_2}), X_{m_1} \times X_{m_2})$$

onde:

$$f_{1\|2} : (X_1 \times X_2) \times (\Sigma_1 \cup \Sigma_2) \rightarrow (X_1 \times X_2)$$

Ou seja:

$$f_{1\|2}((x_1,x_2),\sigma) = \begin{cases} (f_1(x_1,\sigma), f_2(x_2,\sigma)) & \text{se } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ e } \sigma \in \Sigma_1(x_1) \cup \Sigma_2(x_2) \\ (f_1(x_1,\sigma), x_2) & \text{se } \sigma \in \Sigma_1 \text{ e } \sigma \notin \Sigma_2 \text{ e } \sigma \in \Sigma_1(x_1) \\ (x_1, f_2(x_2,\sigma)) & \text{se } \sigma \in \Sigma_2 \text{ e } \sigma \notin \Sigma_1 \text{ e } \sigma \in \Sigma_2(x_2) \\ \text{indefinida} & \text{caso contrário} \end{cases}$$

- A operação $trim(A)$ retorna um autômato acessível e co-acessível;
- Cada estado do autômato obtido através da composição síncrona $A\|B$ pode ser interpretado como um par de estados $(n1,n2)$ onde $n1$ é um estado de A e $n2$ é um estado de B;
- Um estado de $C = P\|K$ pode ser analisado como um par $c = (z,x)$, onde z é um estado da planta P e x é um estado da linguagem alvo K. O estado c é um mau estado se $V_P(z) \neq V_C(z,x)$, ou seja, se o conjunto de eventos da planta no estado x não for igual ao conjunto de eventos do estado c do autômato obtido através do produto síncrono da planta com a linguagem alvo. Este estado deve ser eliminado porque o supervisor não pode desabilitar eventos, apenas condições.

Algoritmo 5.1 Algoritmo supCP

$fim \leftarrow false$; $i \leftarrow 0$
$C_0 = trim(P\|K)$
enquanto $fim = false$ **faça**
  Identifique maus estados em C_i
  Modifique C_i eliminando os maus estados, assim como suas transições de entrada e saída associadas
  $C_{i+1} = trim(C_i)$;
  se $C_{i+1} = C_i$ **então**
    $C = C_i$,
    $fim = true$
  fim se
  $i = i + 1$
fim enquanto

O algoritmo 5.1 encontra a máxima linguagem controlável do sistema, e ela é modelada em um autômato $(Z, V \times U, \rho, z_0, Z_m)$. O algoritmo 5.2 apresenta um procedimento para obter a realização de estado discreto de um supervisor C/E $\mathcal{S}$ partindo de um supervisor SED S. Para compreender o algoritmo são necessárias algumas definições:

- O supervisor SED S é definido como $(Z, V \times U, \rho, z_0, Z_m)$, onde Z é o conjunto de estados, $V \times U$ o alfabeto, ρ a função de transição, z_0 o estado inicial e Z_m o conjunto de estados marcados;
- O supervisor C/E $\mathcal{S}$ é definido como (Y, ξ, ψ, y_0) com os conjuntos V, U, onde Y é o conjunto de estados, ξ é a função de transição, ψ a função saída condição, y_0 o estado inicial, V o conjunto de entradas evento e U o conjunto de saídas condição;
- $u(p)$ é o valor do sinal condição que está entrando no estado p do autômato que modela o SED;
- $V(p)$ para $p \in P$ é o conjunto de eventos das etiquetas de transições que saem do estado p;
- Para $v \in V(p)$, $U(v,p)$ é o conjunto de todas as condições envolvidas nas transições de saída para o estado $p \in P$ que relacionam-se ao evento v.

Algoritmo 5.2 Algoritmo para encontrar a realização de estado discreto do supervisor C/E

$Y \leftarrow \{y_0\}$; $P(y_0) \leftarrow \{p_0\}$; $Y' \leftarrow Y$;
$\xi = \emptyset$; $\psi \leftarrow \emptyset$; Conjuntos para armazenar as transições de estado e as informações de saída
enquanto $Y' \neq \emptyset$ **faça**
  Selecione e remova y de Y'
  para $p \in P(y) \wedge v \in V(p)$ **faça**
    $m \leftarrow U(v,p)$
    $P' \leftarrow \cup\rho(p, vu)$, $u \in m$
    se $\forall y \in Y\ P' \neq P(y)$ **então**
      Adicione novos estados y' a Y e Y' (tantos quantos forem os elementos de P') com $P(y') \subset P'$
    fim se
    $x \leftarrow p$
    enquanto $P' \neq \emptyset$ **faça**
      Selecione p e remova de P';
      Selecione $y' \in Y$ tal que $P(y') = p$
      $\xi(y, u(x), v) \leftarrow y'$
      $\psi(y') \leftarrow u(p)$
    fim enquanto
  fim para
fim enquanto

5.4 Aplicação do Controle Supervisório

Após formalizar os conceitos relacionados ao controle supervisório de sistemas C/E, é possível voltar ao exemplo da estação de espera e encontrar um supervisor C/E para o sistema.

O teorema 5.3 afirma que encontrando um supervisor SED é possível encontrar um supervisor C/E equivalente. Este é o procedimento para encontrar um supervisor para a planta livre do sistema da estação de espera.

O modelo lógico da planta pode ser expresso em um autômato, como afirmado anteriormente. O sistema C/E passa a ser modelado como um sistema a eventos discretos com estrutura de controle, $P = (L, \Gamma)$, onde $L = \mathcal{L}(\mathcal{P})$ é a linguagem que descreve o comportamento lógico da planta livre.

O comportamento desejado para o sistema é expresso através de uma seqüência de eventos, e é representado através de autômatos de estados finitos. No exemplo o atuador A_x pode ser estendido ou recolhido livremente. Caso o atuador A_x esteja estendido, com a chegada de um *pallet* na estação de espera, evento identificado pela sensibilização do sensor S_i, o atuador A_y é acionado e só é recolhido quando o *pallet* sai da estação, evento identificado pela dessensibilização do sensor S_i. Este procedimento garante que não existam dois *pallets* contíguos detectados como um só pelo sensor S_i. Caso dois *pallets* cheguem juntos a estação, se o primeiro parar na estação, o segundo será separado "à força" pelo atuador A_y. O autômato que representa esta especificação está na figura 5.2.

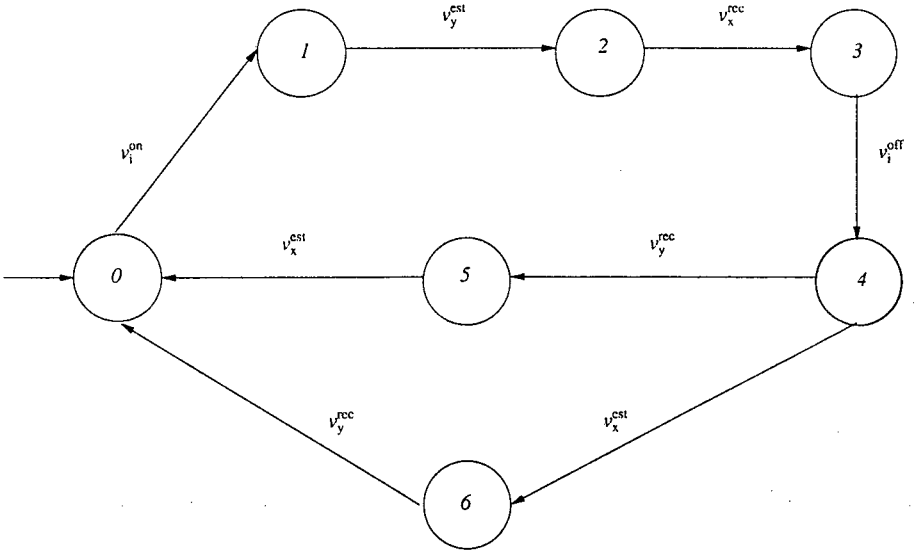

Figura 5.2: Comportamento desejado do sistema expresso por eventos dos subsistemas

A especificação da figura 5.2 apresenta apenas os eventos dos subsistemas, e não os eventos $v \in V_P$ da planta, que são vetores. O autômato que representa a especificação ($E \subseteq V_P^*$) está na figura 5.3. Uma tabela de tradução dos eventos está a seguir:

$$V_P = \left\{ \begin{array}{llll} [& v_x^{est} & \eta & \eta &] \\ [& v_x^{rec} & \eta & \eta &] \\ [& \eta & v_y^{rec} & \eta &] \\ [& \eta & v_y^{est} & \eta &] \\ [& \eta & \eta & v_i^{on} &] \\ [& \eta & \eta & v_i^{off} &] \end{array} \right\} = \left\{ \begin{array}{ll} [& v_1 \ ] \\ [& v_2 \ ] \\ [& v_3 \ ] \\ [& v_4 \ ] \\ [& v_5 \ ] \\ [& v_6 \ ] \end{array} \right\}$$

Sendo que o evento nulo é o vetor $v_0 = [\eta, \eta, \eta]$.

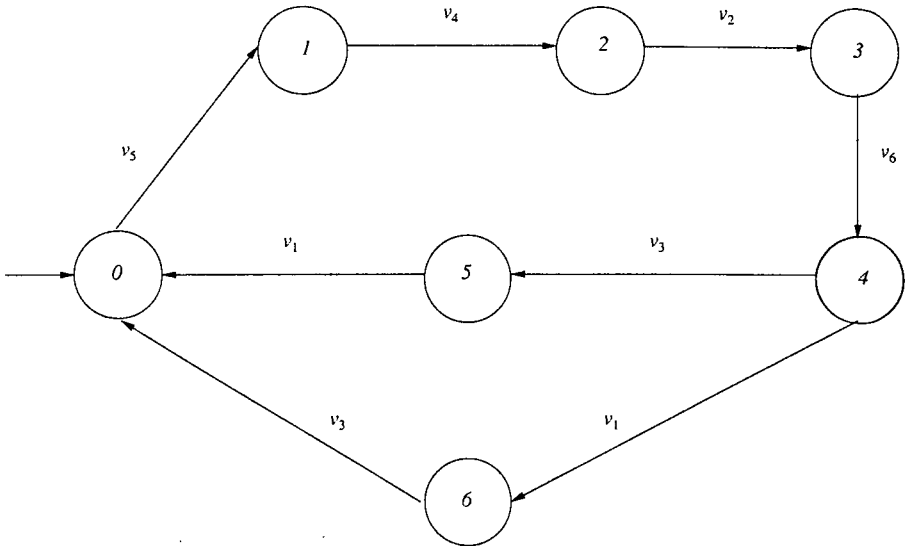

Figura 5.3: Comportamento desejado do sistema expresso por eventos da planta

Para encontrar o supervisor é necessário encontrar a linguagem alvo correspondente a planta livre. A linguagem alvo é $K = p_v^{-1}(E) \cap L$. Aplicando-se o algoritmo do *supCP* encontra-se a máxima linguagem controlável e prefixo-fechada, que é reconhecida pelo autômato da figura 5.4.

Uma tabela de tradução das condições está a seguir:

$$U_P = \left\{ \begin{array}{ll} [& u_x^{est} \quad u_y^{rec} \ ] \\ [& u_x^{est} \quad u_y^{est} \ ] \\ [& u_x^{rec} \quad u_y^{rec} \ ] \\ [& u_x^{rec} \quad u_y^{est} \ ] \end{array} \right\} = \left\{ \begin{array}{ll} [& u_1 \ ] \\ [& u_2 \ ] \\ [& u_3 \ ] \\ [& u_4 \ ] \end{array} \right\}$$

O autômato da figura 5.4 representa o supervisor SED, o próximo passo é encontrar a realização de estado discreto do supervisor C/E equivalente. Aplicando-se o algoritmo 5.2 obtém-se a realização de estado discreto do supervisor C/E, cujas funções de transição e saída evento estão respectivamente nas tabelas 5.1 e 5.2. O autômato que representa o supervisor C/E é o mesmo da figura 5.4.

Cada estado do supervisor pode ser interpretado como um par de estados $(n1, n2)$ onde $n1$ seria equivalente ao estados da planta livre e $n2$ equivalente ao estado da linguagem alvo.

5.5 Procedimento de Obtenção do Supervisor C/E

Assim como no capítulo de modelagem, é possível resumir o procedimento de obtenção de um supervisor C/E em alguns passos.

y^-	v	y
0	v_0	1
1	v_5	2
2	v_4	3
3	v_2	4
4	v_6	5
4	v_6	6
5	v_1	7
6	v_3	8
7	v_3	1
8	v_1	1

Tabela 5.1: Função de transição do supervisor C/E

y	u
0	u_1
1	u_2
2	u_4
3	u_4
4	u_2
4	u_3
5	u_1
6	u_1
7	u_1
8	u_1

Tabela 5.2: Função saída condição do supervisor C/E

Tem-se um sistema C/E modelado como uma realização de estado discreto $S = (X, f, h, x_0)$ com os conjuntos U, V. Deseja-se controlar este sistema, e para encontrar um supervisor C/E deve-se seguir os seguintes passos:

1. Obter um autômato que represente a linguagem L do sistema C/E;
2. Definir especificações para o sistema. Estas especificações devem ser feitas em termos de eventos do sistema;
3. Encontrar uma linguagem E que represente a especificação do sistema;
4. Encontrar a linguagem alvo $K = p_v^{-1}(E) \cap L$;
5. Encontrar o supervisor SED do sistema C/E através do algoritmo 5.1;
6. Encontrar a realização de estado discreto do supervisor C/E do sistema C/E a partir do supervisor SED utilizando o algoritmo 5.2.

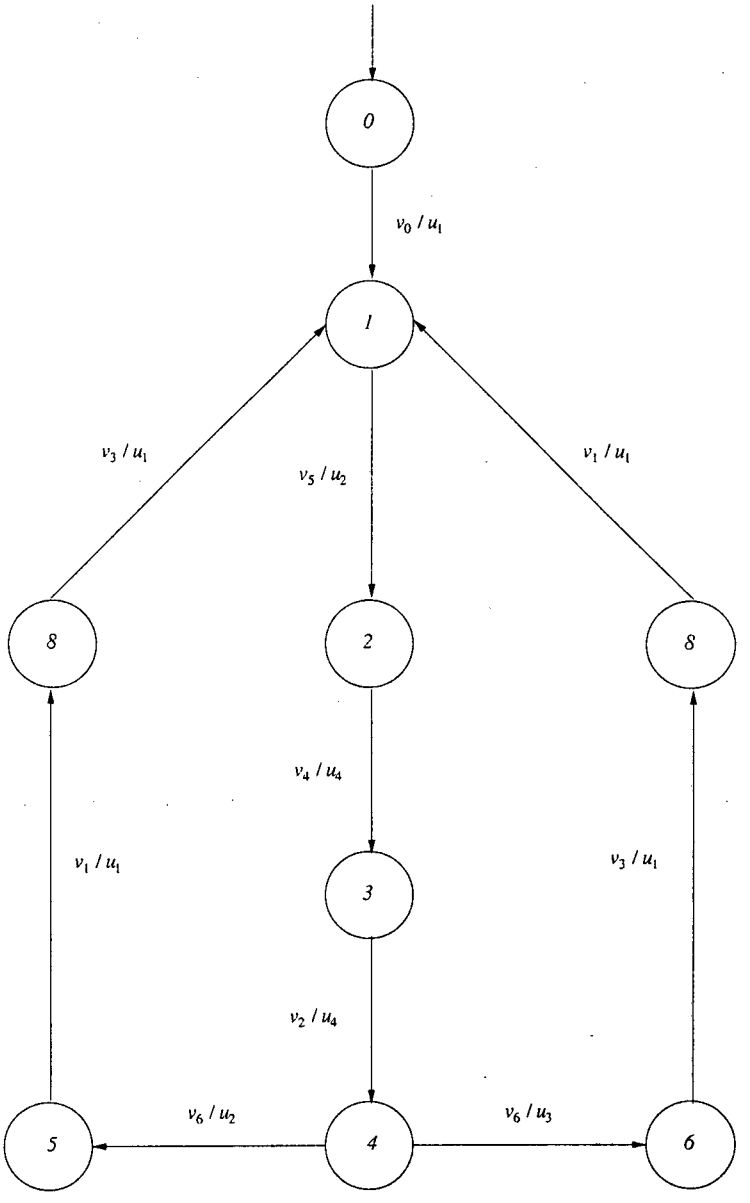

Figura 5.4: Supervisor SED

5.6 Conclusão

Este capítulo apresentou o procedimento de obtenção de um supervisor C/E para sistemas C/E, como o modelado no capítulo anterior.

Um supervisor C/E lê as saídas evento da planta e habilita uma condição na mesma, ou seja, o supervisor possui entradas evento e saídas condição.

O problema de obtenção de um supervisor C/E é resolvido através de um problema equivalente. Um sistema C/E é analisado segundo a teoria de controle supervisório de SEDs e uma estrutura de controle é proposta. Aspectos sobre controlabilidade são abordados e um supervisor SED é encontrado para o sistema. O teorema 5.3 afirma que se um supervisor SED é encontrado, um supervisor C/E equivalente também pode ser obtido.

Algoritmos para obtenção dos supervisores são apresentados e uma aplicação também. É desenvolvido o procedimento de obtenção do supervisor C/E para o exemplo da estação de espera.

Capítulo 6

Exemplo

Este capítulo tem o intuito de solidificar os conceitos apresentados anteriormente. O procedimento de modelagem será aplicado a um sistema mais complexo. Um sistema similar foi analisado em [5] . Além disso, os passos para encontrar um supervisor serão seguidos.

6.1 Definição do Problema

Na linha de produção de uma indústria existe uma mesa giratória de três posições onde é efetuada operação de furo de peças, como mostra a figura 6.1. O funcionamento da mesa giratória é comandado por um controlador lógico programável (CLP) conforme a seguinte seqüência de passos:

1. A esteira gira até que uma mesa seja posicionada em P_1;
2. A mesa gira 120°;
3. A peça é furada;
4. A mesa gira 120°;
5. O manipulador robótico retira a peça da mesa.

A mesa foi inicialmente projetada para operar em seqüência apenas uma peça por vez, ou seja, a esteira só pode ser acionada novamente depois que o manipulador retirar a peça da mesa. Entretanto, esse modo de funcionamento é muito pouco eficiente, visto que a esteira, a furadeira e o manipulador passam a maior parte do tempo parados, enquanto poderiam estar operando em paralelo. O problema a ser resolvido é encontrar um supervisor que evite a ociosidade do sistema, respeitando as restrições de segurança e os problemas de fluxo de peças.

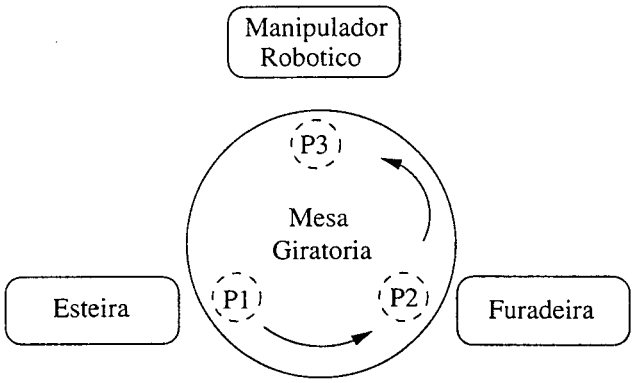

Figura 6.1: Sistema com mesa giratória

6.2 Modelagem do Sistema

O primeiro passo da modelagem do sistema é identificar os subsistemas do mesmo. Neste caso o sistema é formado por quatro subsistemas:

- A esteira;
- A furadeira;
- O manipulador robótico;
- A mesa.

Os subsistemas não se relacionam, logo não sofrem a operação de conexão. As suas entradas condição e saídas eventos são, respectivamente, as entradas e saídas do sistema global. Devido a este fato, o conjunto de saída condição e a função saída condição não serão definidos.

O próximo passo é encontrar uma realização de estado discreto para cada subsistema.

6.2.1 Esteira

O primeiro subsistema a ser analisado é a esteira (sistema C/E S_{est}). A esteira possui uma realização de estado discreto $S_{est} = (X_{est}, f_{est}, h_{est}, x_{est})$ com os conjuntos U_{est}, V_{est}, onde:

- X_{est} é o conjunto de estados $X_{est} = \{x_{est}^{on}, x_{est}^{off}\}$;
- U_{est} é o conjunto de entradas condição $U_{est} = \{u_{est}^{on}, u_{est}^{off}\}$;
- V_{est} é o conjunto de saídas evento $V_{est} = \{v_{est}^{on}, v_{est}^{off}\}$;

x_{est}^{-}	u_{est}^{-}	x_{est}
x_{est}^{off}	u_{est}^{off}	x_{est}^{off}
x_{est}^{off}	u_{est}^{on}	x_{est}^{on}
x_{est}^{on}	u_{est}^{on}	x_{est}^{on}
x_{est}^{on}	u_{est}^{off}	x_{est}^{off}

Tabela 6.1: Função de transição da esteira

x_{est}^{-}	x_{est}	v_{est}
x_{est}^{off}	x_{est}^{off}	η
x_{est}^{off}	x_{est}^{on}	v_{est}^{on}
x_{est}^{on}	x_{est}^{on}	η
x_{est}^{on}	x_{est}^{off}	v_{est}^{off}

Tabela 6.2: Função saída evento da esteira

- f_{est} é a função de transição $f_{est} : X_{est} \times U_{est} \rightarrow X_{est}$ e está definida na tabela 6.1;
- h_{est} é a função saída evento $h_{est} : X_{est} \times X_{est} \rightarrow V_{est}^{+}$ e está definida na tabela 6.2;
- x_{est} é o estado inicial, $x_{est} = x_{est}^{off}$

A esteira pode encontrar-se em dois estados distintos, trabalhando (x_{est}^{on}) ou parada (x_{est}^{off}).

A entrada condição u_{est}^{on} simboliza o comando de ligar a esteira, enquanto u_{est}^{off} o comando de desligar a esteira, uma vez que a peça esteja sobre a mesa. Os eventos de resposta a estes comandos são representados pelas saídas evento v_{est}^{on} (resposta ao comando ligar) e v_{est}^{off} (resposta ao comando desligar).

6.2.2 Furadeira

O segundo subsistema a ser analisado é a furadeira (sistema C/E S_{fur}). A furadeira possui uma realização de estado discreto $S_{fur} = (X_{fur}, f_{fur}, h_{fur}, x_{fur})$ com os conjuntos U_{fur}, V_{fur}, onde:

- X_{fur} é o conjunto de estados $X_{fur} = \{x_{fur}^{on}, x_{fur}^{off}\}$;
- U_{fur} é o conjunto de entradas condição $U_{fur} = \{u_{fur}^{on}, u_{fur}^{off}\}$;
- V_{fur} é o conjunto de saídas evento $V_{fur} = \{v_{fur}^{on}, v_{fur}^{off}\}$;
- f_{fur} é a função de transição $f_{fur} : X_{fur} \times U_{fur} \rightarrow X_{fur}$ e está definida na tabela 6.3;
- h_{fur} é a função saída evento $h_{fur} : X_{fur} \times X_{fur} \rightarrow V_{fur}^{+}$ e está definida na tabela 6.4;
- x_{fur} é o estado inicial, $x_{fur} = x_{fur}^{off}$

x_{fur}^{-}	u_{fur}^{-}	x_{fur}
x_{fur}^{off}	u_{fur}^{off}	x_{fur}^{off}
x_{fur}^{off}	u_{fur}^{on}	x_{fur}^{on}
x_{fur}^{on}	u_{fur}^{on}	x_{fur}^{on}
x_{fur}^{on}	u_{fur}^{off}	x_{fur}^{off}

Tabela 6.3: Função de transição da furadeira

x_{fur}^{-}	x_{fur}	v_{fur}
x_{fur}^{off}	x_{fur}^{off}	η
x_{fur}^{off}	x_{fur}^{on}	v_{fur}^{on}
x_{fur}^{on}	x_{fur}^{on}	η
x_{fur}^{on}	x_{fur}^{off}	v_{fur}^{off}

Tabela 6.4: Função saída evento da furadeira

A furadeira pode encontrar-se em dois estados distintos, trabalhando (x_{fur}^{on}) ou parada (x_{fur}^{off}).

A entrada condição u_{fur}^{on} simboliza o comando de ligar a furadeira, enquanto u_{fur}^{off} o comando de desligar a furadeira. Os eventos de resposta a estes comandos são representados pelas saídas evento v_{fur}^{on} (resposta ao comando ligar) e v_{fur}^{off} (resposta ao comando desligar).

6.2.3 Manipulador Robótico

O terceiro subsistema a ser analisado é o manipulador robótico (sistema C/E S_{rob}). O manipulador possui uma realização de estado discreto $S_{rob} = (X_{rob}, f_{rob}, h_{rob}, x_{rob})$ com os conjuntos U_{rob}, V_{rob}, onde:

- X_{rob} é o conjunto de estados $X_{rob} = \{x_{rob}^{on}, x_{rob}^{off}\}$;
- U_{rob} é o conjunto de entradas condição $U_{rob} = \{u_{rob}^{on}, u_{rob}^{off}\}$;
- V_{rob} é o conjunto de saídas evento $V_{rob} = \{v_{rob}^{on}, v_{rob}^{off}\}$;
- f_{rob} é a função de transição $f_{rob} : X_{rob} \times U_{rob} \rightarrow X_{rob}$ e está definida na tabela 6.5;

x_{rob}^{-}	u_{rob}^{-}	x_{rob}
x_{rob}^{off}	u_{rob}^{off}	x_{rob}^{off}
x_{rob}^{off}	u_{rob}^{on}	x_{rob}^{on}
x_{rob}^{on}	u_{rob}^{on}	x_{rob}^{on}
x_{rob}^{on}	u_{rob}^{off}	x_{rob}^{off}

Tabela 6.5: Função de transição do manipulador robótico

x_{rob}^{-}	x_{rob}	v_{rob}
x_{rob}^{off}	x_{rob}^{off}	η
x_{rob}^{off}	x_{rob}^{on}	v_{rob}^{on}
x_{rob}^{on}	x_{rob}^{on}	η
x_{rob}^{on}	x_{rob}^{off}	v_{rob}^{off}

Tabela 6.6: Função saída evento do manipulador robótico

- h_{rob} é a função saída evento $h_{rob} : X_{rob} \times X_{rob} \rightarrow V_{rob}^{+}$ e está definida na tabela 6.6;
- x_{rob} é o estado inicial, $x_{rob} = x_{rob}^{off}$

O manipulador pode encontrar-se em dois estados distintos, trabalhando (x_{rob}^{on}) ou parado (x_{rob}^{off}).

A entrada condição u_{rob}^{on} simboliza o comando de ligar o manipulador, enquanto u_{rob}^{off} o comando de desligar. Os eventos de resposta a estes comandos são representados pelas saídas evento v_{rob}^{on} (resposta ao comando ligar) e v_{rob}^{off} (resposta ao comando desligar).

6.2.4 Mesa

O último subsistema a ser analisado é a mesa (sistema C/E S_{mes}). A mesa possui uma realização de estado discreto $S_{mes} = (X_{mes}, f_{mes}, h_{mes}, x_{mes})$ com os conjuntos U_{mes}, V_{mes}, onde:

- X_{mes} é o conjunto de estados $X_{mes} = \{x_{mes}^{on}, x_{mes}^{off}\}$;
- U_{mes} é o conjunto de entradas condição $U_{mes} = \{u_{mes}^{on}, u_{mes}^{off}\}$;
- V_{mes} é o conjunto de saídas evento $V_{mes} = \{v_{mes}^{on}, v_{mes}^{off}\}$;
- f_{mes} é a função de transição $f_{mes} : X_{mes} \times U_{mes} \rightarrow X_{mes}$ e está definida na tabela 6.7;

x_{mes}^{-}	u_{mes}^{-}	x_{mes}
x_{mes}^{off}	u_{mes}^{off}	x_{mes}^{off}
x_{mes}^{off}	u_{mes}^{on}	x_{mes}^{on}
x_{mes}^{on}	u_{mes}^{on}	x_{mes}^{on}
x_{mes}^{on}	u_{mes}^{off}	x_{mes}^{off}

Tabela 6.7: Função de transição da mesa

- h_{mes} é a função saída evento $h_{mes} : X_{mes} \times X_{mes} \rightarrow V_{mes}^{+}$ e está definida na tabela 6.8;
- x_{mes} é o estado inicial, $x_{mes} = x_{mes}^{off}$

x^{-}_{mes}	x_{mes}	v_{mes}
x^{off}_{mes}	x^{off}_{mes}	η
x^{off}_{mes}	x^{on}_{mes}	v^{on}_{mes}
x^{on}_{mes}	x^{on}_{mes}	η
x^{on}_{mes}	x^{off}_{mes}	v^{off}_{mes}

Tabela 6.8: Função saída evento da mesa

A mesa pode encontrar-se em dois estados distintos, trabalhando (x^{on}_{mes}) ou parado (x^{off}_{mes}).

A entrada condição u^{on}_{mes} simboliza o comando de girar a mesa, enquanto u^{off}_{mes} o comando de parar a mesa. Os eventos de resposta a estes comandos são representados pelas saídas evento v^{on}_{mes} (resposta ao comando girar) e v^{off}_{mes} (resposta ao comando parar).

6.2.5 Planta Livre

Após obter os modelos dos subsistemas é possível encontrar o modelo C/E da planta livre $\mathcal{P}$. Para isto basta aplicar o algoritmo da operação empilhar, que foi implementado na linguagem C (ver Apêndice A). Através dele é possível obter uma realização de estado discreto para planta livre.

O programa retorna as funções de transição e saída evento da planta livre, bem como uma tabela de tradução, sendo possível identificar os conjuntos de estados, de entradas condição e de saídas evento.

O conjunto de estados da planta livre é formado por vetores, $X_P = X_{est} \times X_{fur} \times X_{rob} \times X_{mes}$.

$$X_P = \left\{ \begin{array}{l} [\ x_{est}^{off}\ x_{fur}^{off}\ x_{rob}^{off}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{off}\ x_{rob}^{off}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{off}\ x_{rob}^{on}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{off}\ x_{rob}^{on}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{on}\ x_{rob}^{off}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{on}\ x_{rob}^{off}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{on}\ x_{rob}^{on}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{off}\ x_{fur}^{on}\ x_{rob}^{on}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{off}\ x_{rob}^{off}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{off}\ x_{rob}^{off}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{off}\ x_{rob}^{on}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{off}\ x_{rob}^{on}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{on}\ x_{rob}^{off}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{on}\ x_{rob}^{off}\ x_{mes}^{on}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{on}\ x_{rob}^{on}\ x_{mes}^{off}\ ] \\ [\ x_{est}^{on}\ x_{fur}^{on}\ x_{rob}^{on}\ x_{mes}^{on}\ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ x_0\ ] \\ [\ x_1\ ] \\ [\ x_2\ ] \\ [\ x_3\ ] \\ [\ x_4\ ] \\ [\ x_5\ ] \\ [\ x_6\ ] \\ [\ x_7\ ] \\ [\ x_8\ ] \\ [\ x_9\ ] \\ [\ x_{10}\ ] \\ [\ x_{11}\ ] \\ [\ x_{12}\ ] \\ [\ x_{13}\ ] \\ [\ x_{14}\ ] \\ [\ x_{15}\ ] \end{array} \right\}$$

O conjunto de entradas condição da planta livre é formado por vetores, $U_P = U_{est} \times U_{fur} \times U_{rob} \times U_{mes}$.

$$U_P = \left\{ \begin{array}{l} [\ u_{est}^{off}\ u_{fur}^{off}\ u_{rob}^{off}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{off}\ u_{rob}^{off}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{off}\ u_{rob}^{on}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{off}\ u_{rob}^{on}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{on}\ u_{rob}^{off}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{on}\ u_{rob}^{off}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{on}\ u_{rob}^{on}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{off}\ u_{fur}^{on}\ u_{rob}^{on}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{off}\ u_{rob}^{off}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{off}\ u_{rob}^{off}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{off}\ u_{rob}^{on}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{off}\ u_{rob}^{on}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{on}\ u_{rob}^{off}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{on}\ u_{rob}^{off}\ u_{mes}^{on}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{on}\ u_{rob}^{on}\ u_{mes}^{off}\ ] \\ [\ u_{est}^{on}\ u_{fur}^{on}\ u_{rob}^{on}\ u_{mes}^{on}\ ] \end{array} \right\} = \left\{ \begin{array}{l} [\ u_1\ ] \\ [\ u_2\ ] \\ [\ u_3\ ] \\ [\ u_4\ ] \\ [\ u_5\ ] \\ [\ u_6\ ] \\ [\ u_7\ ] \\ [\ u_8\ ] \\ [\ u_9\ ] \\ [\ u_{10}\ ] \\ [\ u_{11}\ ] \\ [\ u_{12}\ ] \\ [\ u_{13}\ ] \\ [\ u_{14}\ ] \\ [\ u_{15}\ ] \\ [\ u_{16}\ ] \end{array} \right\}$$

O conjunto de saídas evento da planta livre é formado por vetores, $V_P \in V_{est} \times V_{fur} \times V_{rob} \times V_{mes}$.

$$V_P = \left\{ \begin{array}{cccc} [\ \eta & \eta & \eta & v_{mes}^{on}\ ] \\ [\ \eta & \eta & v_{rob}^{on} & \eta\ ] \\ [\ \eta & v_{fur}^{on} & \eta & \eta\ ] \\ [\ v_{est}^{on} & \eta & \eta & \eta\ ] \\ [\ \eta & \eta & \eta & v_{mes}^{off}\ ] \\ [\ \eta & \eta & v_{rob}^{off} & \eta\ ] \\ [\ \eta & v_{fur}^{off} & \eta & \eta\ ] \\ [\ v_{est}^{off} & \eta & \eta & \eta\ ] \end{array} \right\} = \left\{ \begin{array}{c} [\ v_1\ ] \\ [\ v_2\ ] \\ [\ v_3\ ] \\ [\ v_4\ ] \\ [\ v_5\ ] \\ [\ v_6\ ] \\ [\ v_7\ ] \\ [\ v_8\ ] \end{array} \right\}$$

O estado inicial do planta livre é o vetor $x_P = [x_{est}^{off}, x_{fur}^{off}, x_{rob}^{off}, x_{mes}^{off}]$.

Após obter as funções de transição e saída evento é possível encontrar um autômato que represente a linguagem do sistema. Isto é possível através do algoritmo 4.1, que também foi implementado.

Após utilizar o programa encontra-se um autômato com 65 estados para representar o comportamento da planta livre. O número de transições é 265, sendo que a ocorrência do evento nulo ($v_0 = [\eta, \eta, \eta, \eta]$) só é permitida no estado inicial, e que não há simultaneidade de eventos. A linguagem representada através do autômato é denominada L.

6.3 Aplicação do Controle Supervisório

Após obter o modelo da planta livre busca-se um supervisor C/E para controlá-la. O relacionamento da planta livre com o supervisor pode ser analisado na figura 6.2.

Para obter um supervisor é necessário definir uma especificação em termos de eventos do sistema global (a planta livre), que neste caso é formada por um conjunto de restrições. Entretanto, para facilitar a visualização, os eventos indicados nos autômatos são relativos aos subsistemas, e não os vetores pertencentes a V_P.

As restrições que devem ser tratadas para que a mesa funcione adequadamente são:

- Garantir que a mesa não vai girar sem ao menos uma peça em P_1 e/ou uma peça furada em P_2;

A figura 6.3 representa a restrição que impede a mesa de girar à toa.

- Impedir a mesa de girar enquanto a esteira, a furadeira ou o robô estiverem operando;

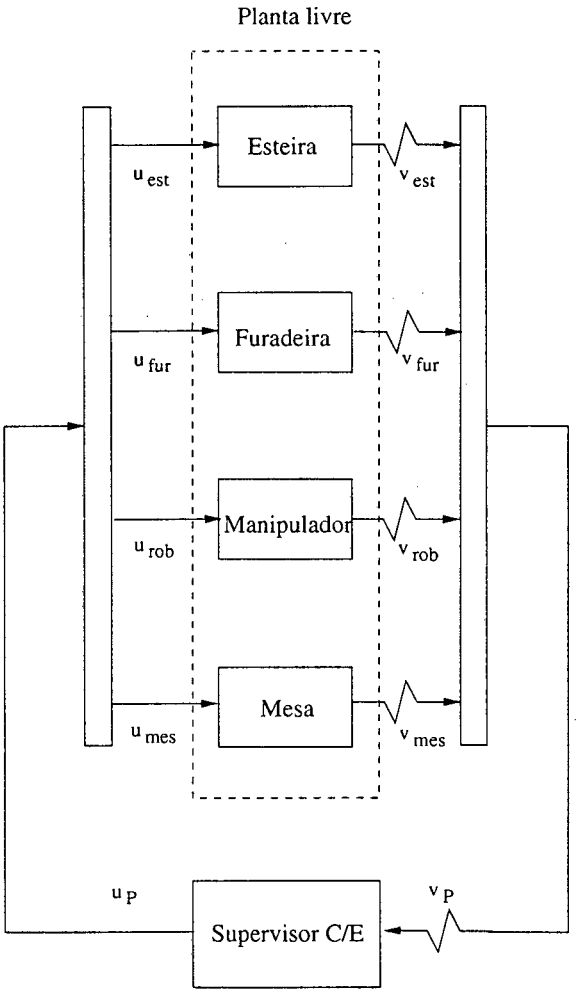

Figura 6.2: Relacionamento entre a planta livre e o supervisor

Esta restrição pode ser representada em três autômatos distintos. A restrição da figura 6.4 impede a mesa de girar enquanto a esteira está operando. A restrição da figura 6.5 impede a mesa de girar enquanto a furadeira está operando. Por fim, a restrição da figura 6.6 impede a mesa de girar enquanto o manipulador está operando.

- Evitar problemas no fluxo de peças, como:

  1. Fluxo entre P_1 e P_2: evitar sobrepor peças em P_1, furar sem peça em P_2 e girar a mesa com peça bruta em P_2;
  2. Fluxo entre P_2 e P_3: evitar furar duas vezes a mesma peça, acionar o manipulador sem peça em P_3 e girar a mesa com peça em P_3.

A restrição que controla o fluxo de peças entre P_1 e P_2 está modelada na figura 6.7, enquanto que a figura 6.8 ilustra a restrição que controla o fluxo de peças entre P_2 e P_3.

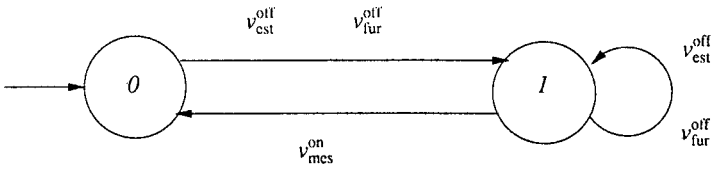

Figura 6.3: Restrição que impede a mesa de girar à toa

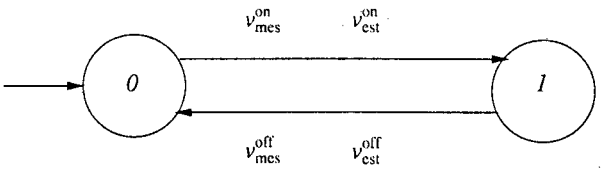

Figura 6.4: Restrição que impede a mesa de girar com a esteira operando

Após modelar as restrições é possível encontrar uma especificação E, realizando o produto síncrono de todas elas. Possuindo uma especificação, obtém-se a linguagem alvo $K = p_v^{-1}(E) \cap L$.

O próximo passo é encontrar o supervisor SED, através do algoritmo 5.1, que foi implementado em trabalhos anteriores (ver Apêndice A). Segundo o teorema 5.3 existe um supervisor C/E equivalente ao supervisor SED encontrado. Para obtê-lo basta aplicar o algoritmo 5.2.

O resultado obtido para o supervisor SED foi um autômato de 159 estados e 290 transições. Algumas transições deste autômato estão listadas a seguir:

```
0 [v0,u9] 1
1 [v4,u1] 2
2 [v8,u2] 3
3 [v1,u1] 4
4 [v5,u5] 5
4 [v5,u9] 6
```

O funcionamento do supervisor pode ser ilustrado através destas poucas transições. O evento $v_0 = [\eta, \eta, \eta, \eta]$ é o evento nulo, e aparece apenas no estado inicial 0, quando é necessário aplicar uma condição inicial à planta, e neste caso é a condição u_9 que indica o comando de ligar a esteira, transitando para o estado 1. No estado 1 o supervisor lê o evento v_4 (a esteira ligou) e a condição que

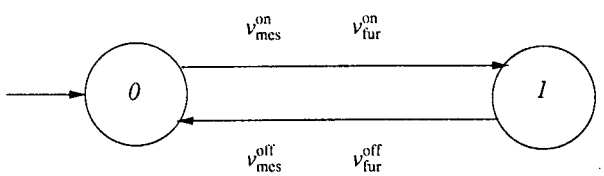

Figura 6.5: Restrição que impede a mesa de girar com a furadeira operando

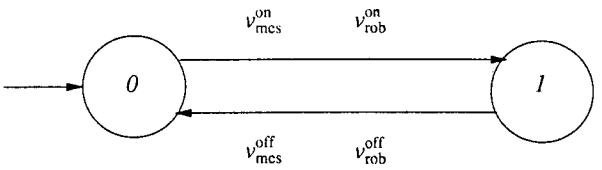

Figura 6.6: Restrição que impede a mesa de girar com o manipulador operando

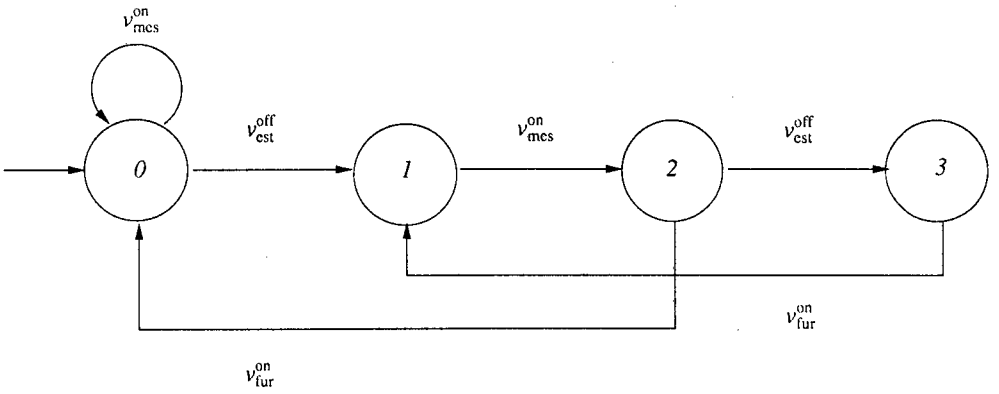

Figura 6.7: Restrição que controla o fluxo de peças entre P_1 e P_2

aplica à planta é u_1, ou seja, troca o valor da condição relacionada à esteira, resultando no comando de desligar a esteira. No estado 2 o supervisor lê o evento v_8 (esteira desligou) e aplica a condição u_2, resultando no comando para girar a mesa. No estado 3 o supervisor lê o evento v_1 (a mesa girou) e aplica a condição u_1, comando para parar a mesa. No estado 4 o supervisor lê o evento v_5 (a mesa parou) e deve aplicar uma de duas condições (u_5 ou u_9), ou seja, neste momento a planta está apta a receber dois comandos (furar a peça ou ligar a esteira). O comportamento da planta é definido dependendo da condição aplicada pelo supervisor.

O supervisor C/E pode ser obtido através do algoritmo 5.2, o qual não foi implementado.

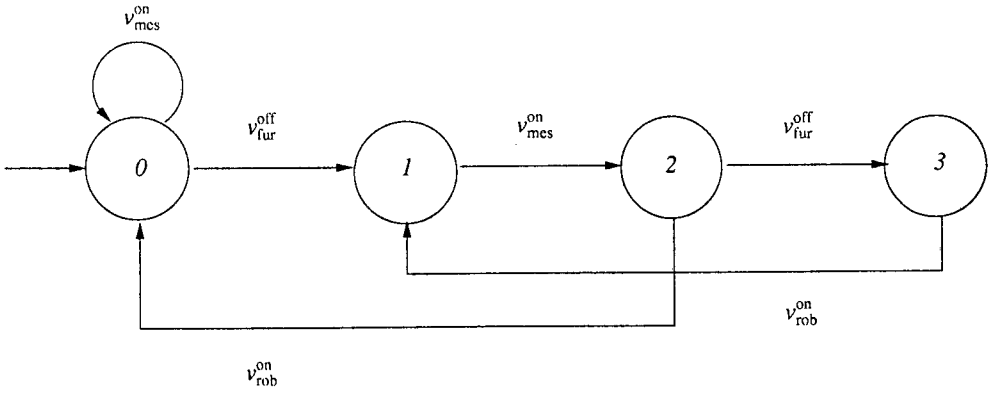

Figura 6.8: Restrição que controla o fluxo de peças entre P_2 e P_3

6.4 Conclusão

O presente capítulo apresentou a modelagem de uma mesa giratória sob o ponto de vista dos sistemas C/E. O intuito era aplicar os conceitos definidos até o presente momento. A metodologia de modelagem se mostrou bastante fácil de ser aplicada.

Além de encontrar o modelo da planta livre do sistema, o caminho para encontrar um supervisor C/E foi apresentado. Por fim, os resultados obtidos implementando computacionalmente o exemplo foram apresentados.

Capítulo 7

Conclusão e Perspectivas

O presente trabalho teve como objetivo utilizar os sistemas condição/evento para resolver um problema de síntese de supervisores para sistemas a eventos discretos controlados por CLPs.

O ponto de partida do trabalho foi um estudo dos sistemas C/E apresentado em [14]. Além disso, o trabalho proposto por [7, 8] também mereceu atenção.

Os sistemas C/E se mostraram próprios para modelar sistemas a eventos discretos controlados por CLPs, e para tal este trabalho propõe uma metodologia de modelagem. Os sistemas são analisados sob o ponto de vista dos sistemas C/E, onde cada subsistema é modelado separadamente.

A metodologia proposta modela os subsistemas isoladamente, obtendo para cada um uma realização de estado discreto. O modelo da planta livre é obtido através da aplicação das operações empilhar e conectar. A proposta da metodologia é analisar os componentes da realização de estados discretos da planta vetorialmente, onde cada componente do vetor está relacionado a um subsistema.

Após obter um modelo para planta livre, foi resolvido um problema de síntese de supervisores. O problema de síntese de supervisores para sistemas C/E é resolvido através de um problema equivalente, quando o sistema é analisado sob o ponto de vista do controle supervisório para SEDs. Um supervisor C/E foi definido, e ele deve ler as saídas eventos da planta para escolher uma entrada condição a ser aplicada na planta. Além de definir um supervisor C/E foi definido um supervisor SED, e a equivalência entre eles foi apresentada.

Ao longo do trabalho foram desenvolvidos diversos algoritmos, sendo que alguns foram implementados.

As principais contribuições deste trabalho são:

- A proposta de uma metodologia de modelagem, analisando o sistema sob o ponto de vista de um sistema C/E;

- O desenvolvimento de algoritmos para auxiliar a utilização da metodologia de modelagem. Os algoritmos da operação empilhar e obtenção de um autômato para sistemas C/E foram implementados;
- A solução do problema da síntese de supervisores para sistemas C/E;
- O desenvolvimento de algoritmos que permitem a obtenção de um supervisor SED e um supervisor C/E.

Quanto a perspectivas futuras, pretende-se estudar aspectos de controle modular e controle hierárquico nos sistemas C/E. Um ponto a ser pesquisado é o comportamento do sistema quando nem todos os estados são finais. Além disso, pretende-se aperfeiçoar os programas desenvolvidos, visando agrupar as funções à ferramenta Grail.

Apêndice A

Ferramentas Computacionais

O exemplo apresentado neste trabalho foi desenvolvido com o auxílio de uma ferramenta computacional chamada **Grail**. Além desta ferramenta foram implementados os algoritmos para encontrar um autômato de um sistema C/E (algoritmo 4.1) e executar a operação empilhar (algoritmo 4.2).

A.1 Ferramenta Grail

A ferramenta Grail é uma biblioteca de funções desenvolvida em C++ e permite a computação simbólica sobre máquinas de estado finitas, expressões regulares e linguagens finitas. Neste trabalho trabalhou-se com as funções para máquinas de estado finitas (FM). No Grail o formato de especificação de uma FM consiste de uma lista de instruções armazenada em um arquivo ASCII. A FM da figura A.1 possui a seguinte especificação no Grail:

```
(START) |- 0
0  a  1
1  b  2
1  -| (FINAL)
2  -| (FINAL)
```

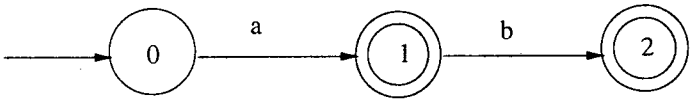

Figura A.1: Máquina de estados finitos

Cada instrução é uma tripla que consiste de um estado fonte, um símbolo (ou uma palavra) e o estado destino. Os estados iniciais devem ser indicados através da pseudo-instrução `(START) |-` e os estados finais devem ser indicados por `-| (FINAL)`.

A biblioteca oferece o conjunto clássico de operações sobre autômatos finitos, tais como minimização e determinação. Além disso também possui funções utilizadas para controle supervisório, como projeção e síntese de supervisores. As funções que o Grail oferece são apresentadas na tabela A.1

Cada função do Grail é também acessível via programas individuais que funcionam como filtros sobre *streams* de especificações de FM. Por exemplo os filtros `fmsync` (produto síncrono) e `fmdeterm` podem ser invocados de forma que o FM de saída do primeiro filtro seja a entrada do segundo:

```
fmsync A B | fmdeterm > AB
```

A FM descrita no arquivo `AB` é o resultado do produto síncrono de `A` e `B` determinizado.

Para visualizar os resultados é possível utilizar a ferramenta **Graphviz**, já que o Grail possui uma função que converte o arquivo ASCII com a FM para o formato da ferramenta (.dot). Para converter um arquivo basta usar o seguinte comando:

```
fmtodot nomearq > nomearq.dot
```

Uma observação deve ser feita sobre a numeração dos estados nas FM geradas pelo Grail. Ao executar a função `m3 = fmsync m1 m2`, n_3 é um estado de `m3` e deseja-se poder determinar estados equivalentes para `m1` e `m2`, isto é, o par (n_1, n_2) equivalente aos estados de `m1` e `m2` correspondentes aos estados de `m3`.

O primeiro passo para encontrar o par (n_1, n_2) é determinar max_1, o máximo número de estado para `m1` e depois realizar a seguinte divisão:

$$\frac{n_3}{max_1 + 1} = n_1$$

e o resto da divisão indica n_2.

A ferramenta Grail foi desenvolvida com a intenção de ser usada em ensino, pesquisa e extensão, e pode ser obtida no seguinte endereço:

`ftp://ftp.lcmi.ufsc.br/pub/Windows/programacao/sed/`

A ferramenta Graphviz pode ser obtida no seguinte endereço:

`http://www.research.att.com/sw/tools/graphviz/`

iscomp	testa se FM é completa
isdeterm	testa se FM é determinística
isomorph	testa se FM é isomorfa
isuniv	testa se FM é universal
fmalpha	tira o alfabeto de uma FM
fmcat	concatena duas FMs
fmcment	complementa uma FM
fmcomp	completa uma FM
fmcondat	informa dados de controle sobre FM
fmcross	intersecciona duas FMs
fmdeterm	torna FM determinística
fmenum	enumera palavras reconhecidas pela FM
fmexec	dada uma cadeia executa a FM
fmloop	faz o self-loop de eventos da primeira FM na segunda FM
fmmark	marca todos os estados da FM
fmmin	minimiza a FM
fmminrev	minimiza a FM (outro método)
fmplus	faz o plus de uma FM
fmproj	faz a projeção de uma FM
fmreach	retira a componente acessível de uma FM
fmremove	elimina estados de uma FM
fmrenum	renumera os estados de uma FM
fmreverse	encontra o reverso de uma FM
fmsort	sorteia as instruções para os estados
fmstar	faz o fechamento Kleene de uma FM
fmstats	obtêm informações sobre a FM
fmsupc	encontra a máxima linguagem controlável
fmsync	faz o produto síncrono de duas FMs
fmtodot	converte uma FM para o formato .dot
fmtovcg	converte uma FM para o formato .vcg
fmtrim	encontra a componente trim de uma FM
fmunion	encontra a união de duas FMs

Tabela A.1: Funções do Grail

A.2 Funções para Sistemas C/E

Ao longo do trabalho foram apresentados alguns algoritmos relacionados com a solução do problema de modelagem e síntese de supervisores de sistemas C/E.

Neste trabalho foram implementados os algoritmos referentes a modelagem dos sistemas C/E, algoritmo 4.2 e algoritmo 4.1. Os programas foram implementados na linguagem C e a sua utilização será descrita a seguir.

O programa que executa a operação empilhar foi denominado `stack`, e é invocado na linha de comando da seguinte maneira:

```
stack <arq-contr> <arq-trans> <arq-saída> [<tab-trad>]
```

Onde:

- `arq-contr` contém a lista de arquivos que descrevem os subsistemas;
- `arq-trans` é usado para armazenar a função de transição do sistema C/E;
- `arq-saída` é usado para armazenar a função saída evento do sistema C/E;
- `tab-trad` é um parâmetro opcional usado para armazenar uma tabela de tradução dos eventos e condições do sistema C/E.

O `arq-contr` é um arquivo de controle, onde cada linha possui informações sobre um subsistema. Em cada linha estão os nomes dos arquivos correpondentes a função de transição f e saída transição h de cada subsistema. O programa `stack` trabalha com *strings*, logo os elementos dos subsistemas podem ser um conjunto de números, letras ou símbolos.

Suponha um sistema formado pelos subsistemas A, B, C, o arquivo de controle teria o seguinte *layout*:

```
arq-f-A arq-h-A
arq-f-B arq-h-B
arq-f-C arq-h-C
```

Os arquivos que contém a função de transição devem conter na primeira linha o estado inicial, e nas demais a tabela de transição de estados. O arquivo `arq-f-A` poderia conter as seguintes transições:

```
x_1
x_1 u_1 x_1
x_1 u_2 x_2
x_2 u_2 x_2
x_2 u_1 x_1
```

Onde x_i indicam os estados e u_i as entradas condição.

O arquivo `arq-h-A` contém a função saída evento do subsistema `A` e poderia conter as seguintes linhas:

```
x_1 x_1 eta
x_1 x_2 v_2
x_2 x_2 eta
x_2 x_1 v_1
```

Onde x_i indicam os estados, v_i as saídas evento e o evento nulo deve ser sempre indicado com `eta`.

Os arquivos de saída `arq-trans` e `arq-saída` armazenam a função de transição e a função saída evento, respectivamente, da planta livre do sistema. O arquivo é semelhante aos apresentados para `arq-f-A` e `arq-h-A`, porém os elementos são vetores. O arquivo de tradução é importante porque o programa `stack` nomeia os vetores com as entradas condição e saídas evento.

Após encontrar a planta livre do sistema é possível encontrar um autômato que a represente. Isto é possível utilizando o programa `iotofm`.

O programa que encontra um autômato para um sistema C/E foi denominado `iotofm`, e é invocado na linha de comando da seguinte maneira:

```
iotofm <arq-trans> <arq-saída-h> [<tab-trad>] > <arq-solução>
```

Onde:

- `arq-trans` contém a função de transição do sistema C/E;
- `arq-saída-h` contém a função saída evento do sistema C/E;
- `tab-trad` é uma saída opcional usada para armazenar uma tabela de tradução dos estados do autômato;

- `arq-solução` é usado para armazenar o autômato encontrado.

Os parâmetros de entrada do programa são `arq-trans` e `arq-saída`, e possuem o mesmo formato dos arquivos `arq-f-A` e `arq-h-A`. As saídas do programa `stack` podem ser entradas do programa `iotofm`.

A tabela de tradução é importante porque no autômato de saída os estados são valores inteiros, e na realidade representam um par (estado, condição). A equivalência dos valores inteiros com os pares encontra-se nesta tabela.

A saída do programa é um arquivo ASCII contendo as transições de estado do autômato. A especificação do autômato é compatível com as FM do Grail, logo pode-se utilizar as funções do Grail sem problemas. No entanto, as etiquetas das transições são pares [v,u].

Em trabalho anterior foi implementado o algoritmo para encontrar o supervisor SED de um sistema C/E, algoritmo 5.1. Este programa foi denominado `iosupc` e deve ser invocado da seguinte maneira:

```
iosupc <arq-planta> <arq-espec>  > <arq-supervisor>
```

Onde:

- `arq-planta` é o autômato que representa a planta a ser controlada;
- `arq-espec` é a especificação que a planta deve seguir;
- `arq-supervisor` é a saída do programa, um autômato que representa a máxima linguagem controlável.

A especificação é obtida através do produto síncrono da planta com as restrições de controle, através da função `fmsync`. Todos os autômatos devem ser definidos segundo a especificação de FM do pacote Grail.

Resumidamente, os passos para resolver um problema de controle supervisório no Grail são os seguintes:

1. Definir os arquivos com as funções de transição e saída evento para cada subsistema;
2. Definir o arquivo de controle;
3. Utilizar a função `stack` para encontrar a planta livre;

4. Utilizar a função `iotofm` para encontrar o autômato da planta;
5. Definir os arquivos com as especificações, se existirem mais de uma utilizar a função `fmsync` para encontrar a especificação global;
6. Encontrar a linguagem alvo, realizando o produto síncrono da planta com a especificação global, através da função `fmsync`;
7. Utilizar a função `iosupc` para encontrar o supervisor SED da planta.

Referências Bibliográficas

[1] BALEMI, S. *Control of Discrete Event Systems: Theory and Application.* Tese (doutorado), Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, May 1992.

[2] CASSANDRAS, C. G., AND LAFORTUNE, S. *Introduction to Discrete Event Systems*, 2nd ed. Kluwer Academic Publishers, Massachussets, 1999.

[3] CURY, J. E. *Notas do Curso de Teoria de Controle Supervório de Sistemas a Eventos Discretos.* V Simpósio Brasileiro de Automação Inteligente, 2001.

[4] DE QUEIROZ, M. H. Controle Modular Local para Sistemas de Grande Porte. Dissertação (mestrado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, 2000.

[5] DE QUEIROZ, M. H., SANTOS, E. A. P., AND CURY, J. E. R. Síntese Modular do Controle Supervisório em Diagrama Escada para uma Célula de Manufatura. In *Anais do V Simpósio Brasileiro de Automação Inteligente* (2001).

[6] FABIAN, M., AND HELLGREN, A. PLC-based Implementation of Supervisory Control for Discrete Event Systems. In *Proceedings of the 37th IEEE Conference on Decision & Control* (Tampa, Florida, December 1998), pp. 3305–3310.

[7] GONZÁLEZ, J. M. E. *Aspectos de Síntese de Supervisores para Sistemas a Eventos Discretos e Sistemas Híbridos.* Tese (doutorado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Brasil, Abril 2000.

[8] GONZÁLEZ, J. M. E., DA CUNHA, A. E. C., CURY, J. R., AND KROGH, B. H. Supervision of Event-Driven Hybrid Systems: Modeling and Synthesis. In *Hybrid Systems: Computation and Control* (Rome, Italy, March 2001), pp. 247–260.

[9] HOPCROFT, J. E., AND ULLMANN, J. D. *Introduction to Automata Theory, Languages and Computation*, 1st ed. Addison Wesley Publishing Company, Reading, 1979.

[10] KROGH, B. H., AND KOWALEWSKI, S. Boolean Condition/Event Systems: Computational Representation and Algorithms. In *12Th World Congress International Federation of Automatic Control* (1993), vol. 3, pp. 327–330.

[11] LEDUC, R. J. *PLC Implementation of DES Supervisor for a Manufacturing Testbed: an Implementation Perpective*. Dissertação (mestrado), Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada, 1996.

[12] RAMADGE, P. J., AND WONHAM, W. M. Supervisory Control of a Class of Discrete Event Processes. *SIAM J. Control Optim. 25*, 1 (Jan. 1987), 206–230.

[13] RAMADGE, P. J., AND WONHAM, W. M. The Control of Discrete Event Systems. *Proceedings of the IEEE 77*, 1 (Jan. 1989), 81–98.

[14] SREENIVAS, R. S., AND KROGH, B. H. On Condition/Event Systems with Discrete State Realizations. *Discrete Event Dynamic Systems: Theory and Applications 1* (1991), 209–236.

[15] TORRICO, C. R. C. Implementação de Controle Supervisório de Sistemas a Eventos Discretos Aplicado a Processos de Manufatura. Dissertação (mestrado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, 1999.

[16] WONHAM, W. M. *Notes on Control of Discrete-Event Systems*. Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, Toronto, Canada, 1999.

[17] ZILLER, R. M. A Abordagem Ramadge-Wonham no Controle de Sistemas a Eventos Discretos: Contribuições à Teoria. Dissertação (mestrado), Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, 1993.