FRANCISCO YASTAMI NAKAMOTO

SISTEMATIZAÇÃO DO PROJETO DO CONTROLE DE SISTEMAS PRODUTIVOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

São Paulo

10/2002

OK

Nakamoto, Francisco Yastami
Sistematização do Projeto do Controle de Sistemas Produtivos, 2002, 147p.

Dissertação de Mestrado – Escola Politécnica da Universidade de São Paulo - Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.
Orientador: Prof. Dr. Diolino José dos Santos Filho.

1. Sistemas Produtivos Flexíveis (SPFs) 2. 'Deadlock' 3. Controle de SPFs I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t

FRANCISCO YASTAMI NAKAMOTO

SISTEMATIZAÇÃO DO PROJETO DO CONTROLE DE SISTEMAS PRODUTIVOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de Concentração: Engenharia Mecânica (Eng. Mecatrônica).

Orientador: Prof. Dr. Diolino José dos Santos Filho.

São Paulo

10/2002

Aos meu pais, Gunji Nakamoto e Mariko Nakamoto.

Aos meus irmãos, Nori, Cláudio, Kanji e Mako.

À minha avó, Konoe Nishijima.

À minha querida, muito querida, Haruko.

" O maior tesouro do ser humano é comparável a beleza e mistério do Universo que dificilmente será desvendado na sua totalidade, mas que a humildade e o amor pela vida o levarão a uma busca incessante ao ato de transformar aquilo que possui num instrumento que auxilie o bem estar da humanidade. Isso o torna uma estrela que irradia fortes luzes pelo imenso Universo.

E ao mais valioso tesouro, o não material mas sentimental, muito claro e ao mesmo tempo subjetivo e inexplicável, que preenche este ser como um todo e não apenas a soma de suas partes, jamais será levado ao esquecimento. Enquanto estes sentimentos de amor e humildade permanecerem, será sempre possuidor de apoio nos momentos difíceis que o possibilitarão a lutar pela humanidade ".

L. K. Yamashita

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Diolino José dos Santos Filho, pelo voto de confiança, incentivo, paciência, amizade e apoio, minha eterna gratidão.

Agradeço ao Prof. Dr. Paulo Eigi Miyagi, pelas contribuições para o desenvolvimento deste trabalho, pelo voto de confiança, incentivo e paciência.

Ao Prof. Dr. Newton Maruyama, pelas contribuições para o desenvolvimento deste trabalho, pelo voto de confiança, incentivo e paciência.

Ao Prof. Dr. José Reinaldo Silva pelo voto de confiança e incentivo.

Agradeço a todos do Grupo de Modelagem Controle e Decisão do Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos da EPUSP, pela amizade, apoio e incentivo.

Ao Prof. Tadao Murata que gentilmente atendeu à minha solicitação enviando-me artigos de sua autoria.

Ao Sr. Engº. Nobuo Shinohara pelo incentivo.

A todas as pessoas do Departamento de Engenharia Mecatrônica da EPUSP que direta e indiretamente contribuíram de alguma forma para a realização deste trabalho, muito obrigado.

SUMÁRIO

ABREVIATURAS

AGV	Automated Guided Vehicle
AGVS	Automated Guided Vehicle Systems
CFE	Ciclo Fechado de Espera
CROPN	Colored Resource Oriented Petri Net
DAA	Deadlock Avoidance Algorithm
E-MFG	Enhanced Mark Flow Graph
FIFO	First In First Out
GAR	Grafo de Alocação de Recursos
LIFO	Last In First Out
MFG	Mark Flow Graph
PFS	Production Flow Schema
PPC	Production Process Circuit
PPN	Production Petri Net
SED	Sistema a Eventos Discretos

SCMA	Sistema de Controle de Malha Aberta
SCMF	Sistema de Controle de Malha Fechada
SLD	Second Level Deadlock
SM	Sistema de Manufatura
SP	Sistema Produtivo
SPF	Sistema Produtivo Flexível
SVC	Sistema de Variáveis Contínuas
TLD	Third Level Deadlock

RESUMO

O objeto de estudo deste trabalho é o sistema de controle de Sistemas Produtivos Flexíveis (SPFs). Os SPFs são sistemas que executam múltiplos processos de forma simultânea compartilhando um mesmo conjunto finito de recursos. O processo é um conjunto de ações de transformação, ou etapas, podendo resultar em um produto ou uma prestação de serviço. A etapa é executada utilizando-se uma única instância de recurso de forma exclusiva, isto é, dois ou mais processos não poderão utilizar a mesma instância de recurso ao mesmo tempo.

O primeiro aspecto a ser considerado é referente à complexidade do sistema. O sistema de controle de SPFs é um sistema complexo. O sistema torna-se complexo devido a imposição de um comportamento dinâmico desejado ao SPFs. A natureza complexa advém do fato da coexistência de dois níveis de indeterminismo.

O primeiro nível de indeterminismo em relação ao tempo é devido à característica dos SPFs pertencerem à classe de Sistemas a Eventos Discretos (SED). Os SED são caracterizados pela existência de paralelismo, conflito e assincronismo, ou seja, a evolução dos estados ocorre de forma assíncrona baseada em eventos que causam uma transição de estados.

O segundo nível de indeterminismo é devido à característica dos SPFs executarem múltiplos processos simultâneos. O controle de cada processo é garantir o seqüenciamento das etapas porém quando é controlado a execução de todos os

processos, ou controle global, perde-se a seqüência de eventos, isto é, não é mais possível determinar quando um evento antecede outro.

Um outro aspecto é que dentro do contexto de SPFs, isto é, a execução múltipla e simultânea de processos com intenso compartilhamento de recursos de um mesmo conjunto finito de recursos, o sistema pode evoluir para um auto-travamento ou "deadlock". O "deadlock" é um fenômeno que ocorre quando o fluxo das etapas dos processos são permanentemente impedidas devido à falta de recursos e/ou informações.

Neste trabalho é adotado um método de "deadlock avoidance" aplicado em uma arquitetura hierárquica do sistema de controle. Esta arquitetura permite abordar o problema da complexidade utilizando o conceito de modelo de restrições, isto é, não é possível determinar todos os estados alcançáveis do sistema porém, é possível evitar que o sistema evolua para determinados estados indesejáveis.

O objetivo deste trabalho é apresentar uma sistematização do método, ou seja, apresentar uma abordagem algorítmica do método. Esta abordagem permite a implementação de uma ferramenta computacional que gera automaticamente o algoritmo de controle de recursos e as regras adicionais de controle com o objetivo de evitar o "deadlock".

ABSTRACT

The study object of this work is the control system of Flexible Production Systems (FPS). FPSs are systems which execute multiple processes simultaneously sharing a same finite set of resources. The first considered aspect is with the reference to the complexity of the system. The control system of FPSs is a complex system. The system becomes complex due to imposition of a dynamic behavior wished to FPSs. The complex nature is due to the fact of the coexistence of two Indetermination levels.

The first indetermination level is regarding time, that is due to the characteristic of FPSs belong to class of Discrete Events Systems (DES). DESs are characterized by the parallelism existence, conflict and desynchronism, in other words, the evolution of the state occurs of desynchronously based on events which cause a state transition.

The second indetermination level is due to the characteristic of FPSs execute multiple processes simultaneously. The control of each process is to guarantee the sequence of the stages however when it is necessary to control the execution of all the processes, or global control, the sequence of events is lost, that is, it is not possible to determine when an event precedes another.

Another aspect is that inside context of FPSs, that is, the multiple and simultaneous execution of processes, sharing the same set of resources, can lead the system to evolve for a deadlock situation. Deadlock is a phenomenon that occurs when

the flow of the stages of the processes are permanently impeded due to the resources lack and/or information.

In this work is adopted a deadlock avoidance method applied in a hierarchical architecture of the control system. This architecture allows to board the problem of the complexity using the restrictions model concept, that is, it is not possible to determine all the reachable state of the system however it is possible to avoid that the system evolves for certain undesirable state.

The goal of this work is to introduce the systematization of the method, in other words, introduce an algorithm approach of the method. This approach allows the implementation of a computational tool which, given some information about FPSs, generates automatically the resources control algorithm and the additional rules of control with the goal of avoiding deadlock.

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1
INTRODUÇÃO

O dinamismo das necessidades do mercado atual reflete-se tanto nos diversos setores do comércio quanto nos vários setores industriais. Observa-se um ambiente extremamente competitivo, em que o objetivo maior consiste em atender a essas necessidades com rapidez, eficiência e eficácia. Dependendo da forma como esta tarefa é conduzida, pode culminar com o sucesso ou fracasso da empresa. Devido a esse dinamismo, o ciclo de vida de um produto tende a ser cada vez menor impactando na abreviação do tempo que compreende entre o projeto, lançamento, produção e encerramento de um produto em si [Santos Filho, 2000a]. Consequentemente, a produção simultânea de uma variedade de produtos torna-se cada vez mais comum nos mais diversos setores da indústria.

Outro aspecto importante diz respeito ao estado tecnológico atual das máquinas, equipamentos e dispositivos de chão de fábrica que compõem os Sistemas Produtivos (SPs). Alicerçados pelo avanço dos recursos computacionais, a custos cada vez menores, viabiliza-se a utilização desses recursos nas variadas etapas de produção nos ambientes industriais [Santos Filho, 2000a].

Neste contexto, surge o conceito de Sistemas Produtivos Flexíveis (SPFs) [Groover, 2000; Kalpakjian e Schmid, 2000; Santos Filho, 2000a]. A proposta fundamental dos SPFs é a integração dos equipamentos, recursos computacionais e recursos humanos presentes nesses ambientes.

Os SPFs caracterizam-se por serem capazes de produzirem uma variedade de produtos de forma concomitante, havendo a capacidade de modificação rápida das configurações do comportamento funcional de acordo com o planejamento da produção. Esta flexibilidade permite um melhor aproveitamento dos recursos incrementando a eficiência. Porém, amplia em demasia a complexidade do controle do sistema.

1.1 MOTIVAÇÕES

O sistema de controle de SPFs é complexo. Esta complexidade advém do fato de poderem coexistir dois níveis de indeterminismo: (i) com relação ao tempo e (ii) com relação ao seqüenciamento dos eventos.

O primeiro nível de indeterminismo é com relação ao tempo. Os SPFs pertencem à classe de Sistemas a Eventos Discretos (SEDs) [Cassandras, 1993]. Portanto, são caracterizados pela existência de paralelismo, conflito e assincronismo. A evolução dos estados do sistema ocorre de forma assíncrona, isto é, baseada em eventos que causam uma transição de estados. Desta forma, não é possível prever o instante em que um evento poderá ocorrer.

O segundo nível de indeterminismo está presente quando se considera o processo global no SPF. Entende-se por processo global, o processo resultante da composição dos vários processos simultâneos e independentes que estão em atividade em um SPF. Quando vários processos são executados concomitantemente não é mais possível pré-determinar a seqüência dos eventos.

A presença deste nível de complexidade é uma motivação para que neste trabalho sejam abordados aspectos pertinentes a aplicação de metodologias para orientar o projeto do controle desses sistemas. Outra motivação para o desenvolvimento deste trabalho é verificada na situação em que vários processos são executados simultaneamente competindo entre si a alocação do mesmo conjunto de recursos. Esta competição entre os processos pode levar o sistema a um auto-travamento ou "deadlock", isto é, o fluxo dos processos são impedidos permanentemente e/ou as operações dos processos não podem mais serem executadas devido à falta de informações, materiais ou recursos [Isloor e Marsland, 1980; Banaszak e Krogh, 1990; Viswanadham et al., 1990; Cho, 1993; Lawley et al., 1997; Fanti et al., 1997; Tanenbaum, 1995; Santos Filho, 2000a].

1.2 Objetivos

O objetivo deste trabalho é a sistematização da geração do controle de SPFs baseado no método proposto por Santos Filho [2000a]. Pretende-se apresentar uma abordagem algorítmica para o método permitindo o desenvolvimento de uma ferramenta computacional que gera de forma automática o algoritmo de controle para SPFs. As regras adicionais de controle geradas na forma de regras de produção, têm como objetivo evitar o estado de "deadlock" no sistema baseado no método de "deadlock avoidance" considerando-se uma estrutura hierárquica de controle [Santos Filho, 1998; Santos Filho, 2000a; Santos Filho, 2001; Nakamoto et al. 2001b; Nakamoto et al., 2001a].

Considerando-se que os sistemas de controle de SPFs são sistemas complexos assume-se que não é possível determinar todos os seus estados alcançáveis. Entretanto, é possível determinar os estados que o sistema não deverá atingir. Portanto, o objetivo genérico passa a ser definir um conjunto de restrições para o devido controle do comportamento do sistema, baseado no conceito de modelo de restrições.

Neste trabalho adota-se a estrutura de um sistema de controle hierárquico proposto por Santos Filho [2000a; 2001], juntamente com a metodologia de modelagem de sistemas de controle e as ferramentas matemáticas associadas:

- Production Flow Schema (PFS) [Miyagi, 1996]: Técnica desenvolvida para sistematizar e facilitar a modelagem por redes;
- Enhanced Mark Flow Graph (E-MFG) [Santos Filho, 1993; Santos Filho, 1998; Santos Filho, 2000a]: Rede interpretada de Redes de Petri [Peterson, 1981; Reisig, 1985; Murata, 1989] criada para modelar sistemas de controle;
- Grafo de Alocação de Recursos (GAR) [Santos Filho 1998; Santos Filho et al. 2000b; Santos Filho et al. 2001]: Grafo bipartido sem marcações para modelar a alocação de recursos.

No sistema de controle hierárquico, o sistema é dividido em dois níveis: controle dos processos (nível inferior) e controle de recursos (nível superior). O controle de recursos desempenha a função de controle supervisório onde as estratégias de controle de utilização dos recursos propriamente ditas são implementadas. Com relação à estratégia de controle dos processos é feita uma analogia com o desenvolvimento de algoritmos

computacionais em que o objetivo é realizar o controle. Desta forma, é utilizado o conceito de modelagem estruturada envolvendo refinamento sucessivo, modularização e o uso de estruturas de controle legítimas [Santos Filho, 2001]. Neste contexto, o nível inferior correspondente ao controle dos processos: a especificação é desenvolvida utilizando-se o E-MFG\PFS [Santos Filho, 1993]. O nível superior correspondente ao controle de recursos: a especificação é desenvolvida utilizando-se o E-MFG e o GAR.

Uma vez definido e especificado o controle dos processos e dos recursos com as ferramentas matemáticas descritas anteriormente, faz-se necessário inserir as regras adicionais de controle no supervisório. Essas regras são as restrições que podem impor um comportamento para o sistema com diferentes finalidades: otimizar a utilização de recursos, evitar o travamento do sistema, aplicar estratégias de produção, entre outros.

No contexto de evitar-se o travamento do sistema, causado por processos simultâneos com intenso compartilhamento de um mesmo conjunto de recursos, por definição, existem três métodos que podem ser aplicados [Isloor e Marsland, 1980; Tanenbaum, 1995; Lawley et al., 1997]:

- Prevenir ("Deadlock Prevent");
- Detectar e Restabelecer ("Deadlock Detection and Resolution");
- Evitar ("Deadlock Avoidance").

Uma vez que os SPFs envolvem a produção de diversos produtos de forma contínua, sem interrupções, será visto ao longo deste trabalho que o método de "deadlock

avoidance" é o mais indicado. Neste trabalho é adotado o método proposto por Santos Filho [2000a] que envolve os seguintes passos:

- Determinar os estados anteriores ao "deadlock";
- Determinar as regras que evitam que o sistema a evolua para o estado de "deadlock";
- Aplicar essas regras no sistema hierárquico de controle.

1.3 METODOLOGIA DE PESQUISA

A metodologia de pesquisa adotada segue o ciclo de desenvolvimento da metodologia de pesquisa citada em Gomes [1997]:

> *"...as técnicas dão o suporte directo à implementação, enquanto as ferramentas são utilizadas para implementar e os métodos permitem evoluir da definição do problema até à sua implementação..."*

A solução de um problema materializa-se baseado na inter-relação entre três aspectos principais (figura 1.1):

- Métodos (Teoria);
- Ferramentas;
- Técnicas (Aplicações).

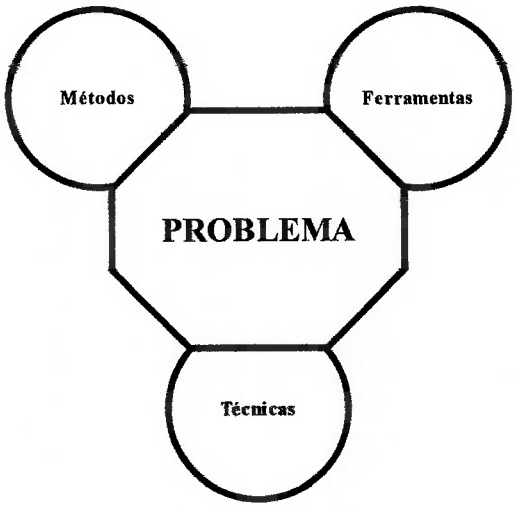


Figura 1.1 Os três aspectos principais na resolução de um problema.

Baseado na trilogia apresentada, a figura 1.2 sintetiza as principais referências do presente trabalho.

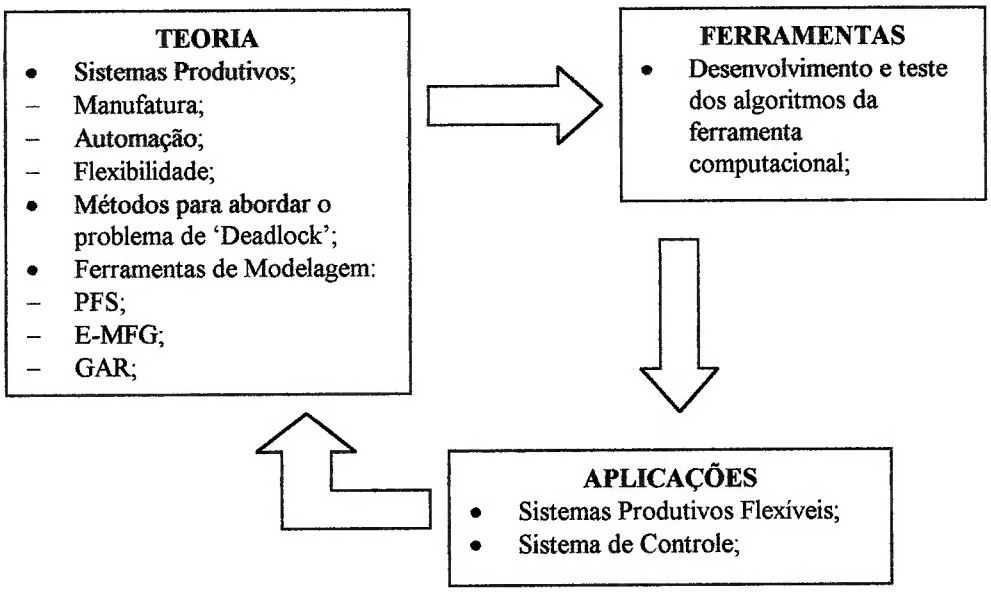


Figura 1.2 Ciclo de desenvolvimento da metodologia proposta.

1.4 Estruturação do Texto

Os primeiros capítulos (capítulos 2 a 5) apresentam uma revisão bibliográfica que caracteriza o objeto de estudo deste trabalho, ou seja, os SPFs. Por sua vez, os capítulos 6 e 7 apresentam as contribuições deste trabalho.

No capítulo 2 são definidos os conceitos de sistema, modelos, estados e eventos. São apresentadas as classificações de sistemas e as descrições de Sistemas a Eventos Discretos (SEDs), Sistemas de Variáveis Contínuas (SVCs) e sistema de manufatura, além dos conceitos de automação, flexibilidade e complexidade em sistemas de manufatura e a relação com os Sistemas Produtivos Flexíveis (SPFs).

No capítulo 3 são apresentados os conceitos de Sistema de Controle e uma análise do comportamento dinâmico dos SPFs. O capítulo 4 apresenta os conceitos, a classificação e os métodos para abordar o "deadlock" em SPFs.

No capítulo 5 é apresentado em detalhes o método adotado para este trabalho.

No capítulo 6 é proposta uma abordagem algorítmica para o método de "deadlock avoidance" proposto por Santos Filho [2000a]. É apresentado também uma proposta de algoritmo para determinação dos Ciclos Fechados de Espera (CFE) e a realização de uma comparação entre o algoritmo proposto e o algoritmo de potenciação de matrizes utilizado no passado [Santos Filho, 2000a].

Finalizando, o capítulo 7 apresenta as observações finais.

CAPÍTULO 2
SISTEMAS PRODUTIVOS FLEXÍVEIS

2.1 INTRODUÇÃO

De acordo com Groover [2000], um Sistema Produtivo (SP) é uma coleção de pessoas, equipamentos e procedimentos organizados para realizar as operações de manufatura de uma companhia sendo formados por (figura 2.1):

- Instalações: Equipamentos e layout da planta, isto é, são as máquinas, equipamentos de manuseio e transporte de materiais, e a forma como estão organizados. As máquinas e os equipamentos são usualmente agrupados de acordo com alguma lógica adotada na organização dos processos produtivos.

- Sistema de suporte à produção: Contempla o planejamento e gerenciamento da produção, isto é, um conjunto de procedimentos que são utilizados para gerenciar a produção resolvendo os problemas técnicos e logísticos que são encontrados em requisição e movimentação de materiais assegurando-se, desta forma, que os produtos estarão dentro dos padrões de qualidade. O projeto de produtos e as estratégias de negócios estão inseridos no sistema de suporte à produção.

Em síntese, os Sistemas Produtivos (SPs) podem ser interpretados como uma classe de sistemas que executam processos e que, por sua vez, são definidos a partir de um conjunto de atividades predeterminadas para realizar ações de transformação.

Neste trabalho é adotado a abordagem conforme Santos Filho [2000a], em que os SPs realizam a produção/manufatura de itens e o conceito pode ser estendido a sistemas que se caracterizam como prestadores de serviços. Outra característica importante nos SPs está relacionado ao homem: são sistemas antropocêntricos, isto é, são sistemas feitos pelo homem e para o homem ("man-made systems") [Ito, 1991; Santos Filho, 2000a]. Estes fatos serão abordados em detalhes no capítulo 3.

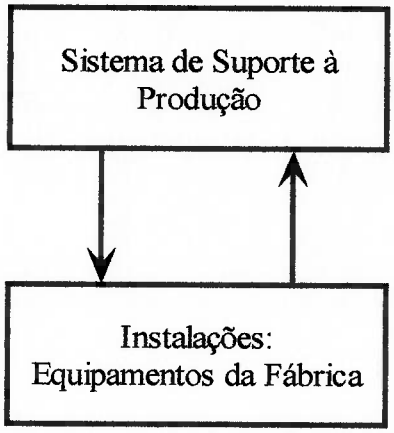


Figura 2.1 O Sistema produtivo conforme Groover [2000].

O objetivo deste capítulo é abordar os principais conceitos fundamentais que implicam na caracterização dos SPFs. Neste sentido, apresenta-se inicialmente os conceitos de sistema, estado e modelo. A seguir é apresentado uma classificação geral dos sistemas e designação da classe a qual pertence os Sistemas Produtivos Flexíveis (SPFs). Finalmente, são apresentados os conceitos de manufatura, automação e flexibilidade que fazem parte do contexto de configuração dos SPFs, objeto de estudo deste trabalho.

2.2 Sistema, Estado e Modelo

Primeiramente, é necessário apresentar algumas definições básicas [Cassandras, 1993]:

- Sistema: Sistema é um conceito primitivo. A palavra sistema é originária do grego que significa "combinar". É uma interação de componentes associados com a intenção de realizar uma determinada função;

- Estado: Estado de um sistema em um instante de tempo **t** descreve o seu comportamento naquele momento de alguma forma mensurável;

- Modelo: Modelo é uma abstração do sistema que pode ser representado por um conjunto de equações matemáticas que refletem um comportamento aproximado do sistema real.

A criação de um modelo a partir de um sistema é realizada considerando-se um conjunto de variáveis de entrada e de saída do sistema que dependem do tempo. Assim, faz-se uma aproximação simplificada do comportamento do sistema de acordo com os objetivos que se pretende atingir, originando-se um modelo [Cassandras, 1993]. Na figura 2.2 é apresentado o processo de modelagem em questão.

As ferramentas de modelagem utilizadas para representarem abstrações dos respectivos sistemas dependem diretamente da natureza desses sistemas.

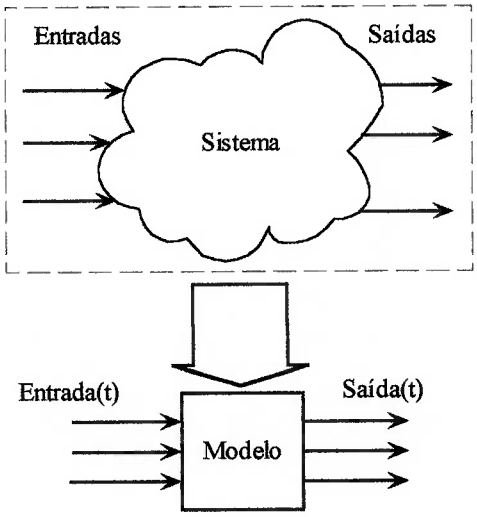


Figura 2.2 Processo de modelagem de um sistema baseado em entradas/saídas em função do tempo.

2.2.1. Classificação dos Sistemas

Para descrever os diferentes aspectos que caracterizam os sistemas, em Cassandras [1993] é apresentada a seguinte classificação:

- Sistemas Instantâneos e Dinâmicos. Nos sistemas instantâneos ou estáticos (figura 2.3) os valores das saídas são normalmente independentes dos valores anteriores das correspondentes entradas. Nos sistemas dinâmicos (figura 2.4), os valores das saídas dependem dos valores das entradas e os valores anteriores das entradas em um determinado instante;

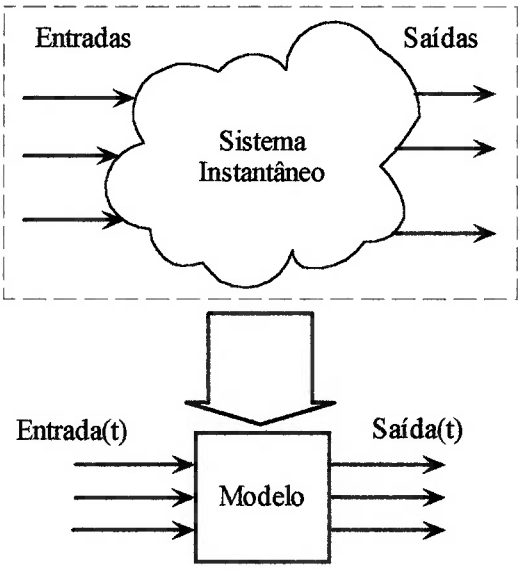


Figura 2.3 Processo de modelagem de sistemas instantâneos.

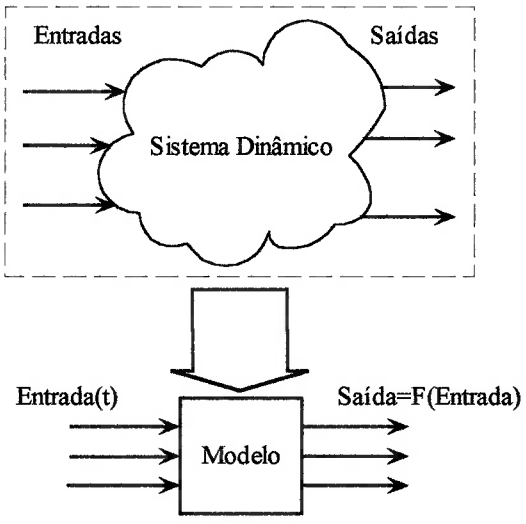


Figura 2.4 Processo de modelagem de sistemas dinâmicos.

- Sistemas Variantes no Tempo e Invariantes no Tempo. O comportamento de um sistema Invariante no Tempo não varia ao longo do tempo, isto é, a forma como é

gerada a resposta será sempre a mesma para a mesma entrada. Se não houver esta dependência será um sistema Variante no Tempo;

- Sistemas Lineares e Não-Lineares. Um sistema linear satisfaz a condição $g(a_1u_1+a_2u_2)=a_1g(u_1)+a_2g(u_2)$, onde u_1 e u_2 são dois vetores de entrada, a_1 e a_2 são números reais e **g(.)** é a função de saída resultante, isto é, o sinal de saída mediante um valor da entrada do sistema. Caso esta condição seja falsa, o sistema será não-linear. Na realidade, todos os sistemas são não-lineares, contudo, nos casos em que a faixa de variação dos sinais for suficientemente pequena, pode-se considerar esse sistema como sendo linear por aproximação;

- Sistemas de Estado Contínuo e Estado Discreto. Em sistemas de estado contínuo, as variáveis de estado podem assumir valores reais. Em sistema de estado discreto as variáveis pertencem ao domínio dos inteiros não negativos;

- Sistemas Dirigidos pelo Tempo e Dirigido por Eventos. Em sistemas dirigidos pelo tempo os estados do sistema mudam continuamente ao longo do tempo. Em sistemas dirigido por eventos, somente a ocorrência assíncrona de eventos discretos causam uma mudança de estado instantaneamente. O estado do sistema entre eventos consecutivos permanece inalterado;

- Sistemas Estocásticos e Determinísticos. Um sistema é estocástico quando a saída é uma variável aleatória, isto é, o comportamento do sistema é baseado em estatísticas. Se não houver uma saída aleatória, o sistema será determinístico;

- Sistemas de Tempo Discreto e Tempo Contínuo. Um sistema de tempo contínuo é onde todas as variáveis de entrada, variáveis de saída e as variáveis de estados são definidas para todos os possíveis valores do tempo. No sistema de tempo discreto, um ou mais dessas variáveis são definitas em pontos discretos no tempo, isto é, amostragem de variáveis no tempo.

De acordo com a classificação apresentada e esquematizada na figura 2.5, os SPFs podem ser caracterizados genericamente como **sistemas dinâmicos, invariantes no tempo, não lineares**, de **estado discreto** e **dirigido por eventos**. Estes aspectos evidenciam o fato desses sistemas poderem ser classificados como Sistemas a Eventos Discretos (SEDs).

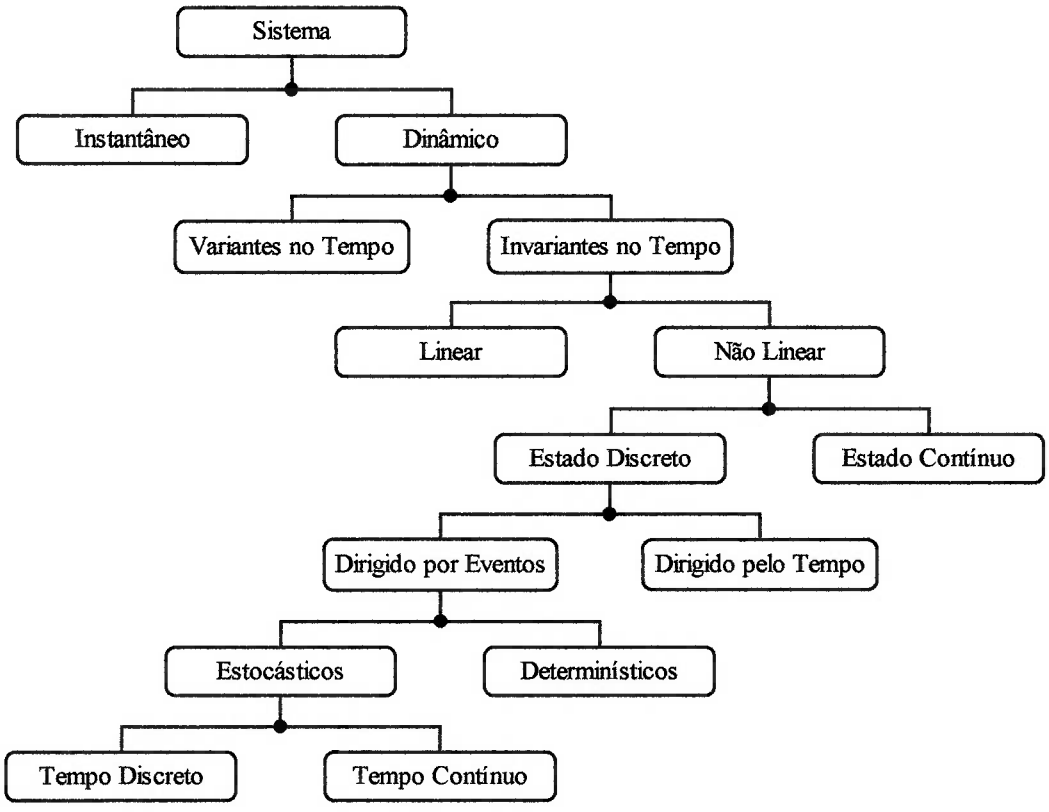


Figura 2.5 Classificação dos sistemas conforme Cassandras [1993].

2.2.2. Sistemas a Eventos Discretos

Existem sistemas cujo comportamento é governado por leis invariantes da física. Em tais sistemas, os estados variam continuamente em função do tempo, ou seja, pertencem à classe de sistemas de variáveis contínuas (SVC). Na figura 2.6 apresenta-se a evolução de uma variável de estado em função do tempo em um SVC.

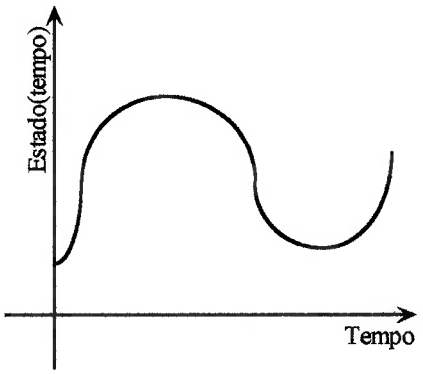


Figura 2.6 Evolução de estados em um SVC.

Ao longo da evolução tecnológica, muitas soluções aplicadas em sistemas reais foram desenvolvidas através da capacidade humana de abstrair situações, criando-se assim os sistemas feitos pelo homem e para o homem que apresentam um comportamento dinâmico que, em geral, não podem ser modelado por leis físicas como os SVCs. Com o objetivo de controlar esses sistemas foi necessário criar uma nova classe de sistemas: Sistemas a Eventos Discretos (SEDs).

Neste contexto é necessário apresentar outra definição [Cassandras, 1993]:

- Evento: Evento é um conceito primitivo. Pode ser identificado como uma ação específica, pode ser visualizado como uma ocorrência espontânea ditada pela

natureza ou pode ser resultado de várias ocorrências. Um evento ocorre instantaneamente e causa uma transição de estado.

Os SEDs são sistemas que podem ser descritos de forma discreta e a evolução dos estados ocorre de forma assíncrona baseada em eventos que causam uma transição de estados (figura 2.7).

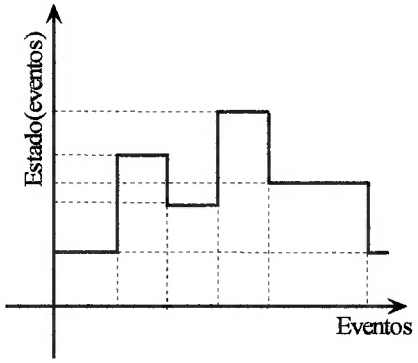


Figura 2.7 Evolução dos estados em um SED.

Através de um modelo do sistema é possível analisar o seu comportamento dinâmico. Existem várias ferramentas para modelar um SED: Cadeias de Markov [Cassandras, 1993], modelos de filas [Cassandras, 1993], autômatos e máquinas de estado [Ramadge e Wonham, 1989; Ho, Cao, 1991; Cassandras, 1993], Redes de Petri e derivadas [Peterson, 1981; Reisig, 1985; Murata, 1989; Miyagi, 1996; Cardoso, Valette, 1997], entre outros.

Os SPFs pertencem à classe de SEDs e, portanto, são caracterizados pela existência de paralelismo, conflito e assincronismo de eventos. Desta forma, não é possível predeterminar quando um certo evento ocorrerá, isto é, possui indeterminismo em relação ao tempo. Para evoluir-se na definição de SPFs é necessário ainda apresentar três conceitos: Manufatura, Automação e Flexibilidade.

2.3 MANUFATURA, AUTOMAÇÃO E FLEXIBILIDADE

2.3.1. Manufatura

A palavra manufatura é derivada do latin *manufactura* que significa "feito a mão". A história da manufatura tem início a aproximadamente 5000 a 4000 anos a.C., estando ligado à própria história do homem. Novos materiais e novos processos tem sido gradualmente desenvolvidos ao longo dos séculos de acordo com as necessidades do homem, aprimorando cada vez mais a manufatura através de operações complexas, incrementando-se taxas de produção e níveis de qualidade [Kalpakjian e Schmid, 2000].

2.3.2. Automação

Segundo Groover [2000], automação é a tecnologia que se baseia no princípio de realizar processos ou procedimentos sem intervenção humana, ou seja, retirar o elemento humano como agente direto do processo e promovê-lo à condição de supervisor do sistema. Para Kalpakjian e Schmid [2000], a automação é definida como um processo que possui máquinas que realizam uma seqüência de operações predeterminadas com pouco ou nenhum trabalho humano.

A automação é alcançada, em todo o seu potencial, utilizando uma variedade de dispositivos, sensores, atuadores, técnicas e equipamentos que são capazes de observar e controlar todos os aspectos de um processo.

Durante a revolução industrial iniciou-se a mecanização da manufatura com o objetivo de produzir-se em grande escala. A mecanização dos processos ou operações foi realizada com a instalação de dispositivos mecânicos e hidráulicos. A tecnologia de produção em massa foi desenvolvida em meados de 1920, baseada em máquinas que possuíam mecanismos automáticos fixos para a produção de um produto específico. A grande inovação na automação foi a criação do controle numérico para máquinas ferramentas em meados de 1950 [Kalpakjian e Schmid, 2000].

À partir de então, a agregação de inovações tecnológicas tem automatizado cada vez mais diversos aspectos da manufatura. Uma das inovações notórias foi a introdução de recursos computacionais que permitiram uma revisão dos principais objetivos da automação [Kalpakjian e Schmid, 2000]:

- Incrementar a Integração dos elementos que realizam os processos de manufatura;
- Melhorar a produtividade para reduzir os custos de manufatura através de um melhor controle da produção;
- Melhorar a qualidade para garantir a repetibilidade de processos;
- Reduzir o envolvimento humano com atividades repetitivas;
- Reduzir perda de matéria prima devido à manipulação manual;
- Elevar o nível de segurança principalmente em condições perigosas de trabalho;
- Economizar espaço físico organizando a planta de forma eficiente.

Um sistema de produtivo automatizado pode ser classificado em três tipos básicos [Groover, 2000; Santos Filho, 2000a]:

- Sistema de Automação Fixa: A seqüência de operações é configurada nos próprios equipamentos. Possui altas taxas de produção e relativa inflexibilidade para se adaptar a novos produtos. Constituem os chamados sistemas dedicados;

- Sistema de Automação Programável: Possui a capacidade de mudar a seqüência de operações de um sistema para adaptar-se à diferentes configurações de produto através de programas. A taxa de produção é menor do que no sistema de automação fixa pois é flexível o bastante para lidar com variações e mudanças de configuração de produtos e é voltado para produção em lotes. A cada novo lote é preciso reprogramar o sistema e realizar o "setup" das máquinas, o que pode consumir algum tempo. Este comportamento pode ocorrer em sistemas celulares de produção;

- Sistema de Automação Flexível: Possui a capacidade de produzir uma variedade de produtos. A taxa de produção é média porém produz continuamente uma variedade de produtos. O que torna a automação flexível possível é quando os processos possuem poucas diferenças. Logo, na maioria dos casos as mudanças na configuração entre os processos são mínimas. São características presentes nos sistemas "job-shop".

2.3.3. Flexibilidade

Para atender determinados objetivos da automação são necessárias tanto a flexibilidade física quanto a flexibilidade lógica. Groover [2000] define o termo flexibilidade como sendo um atributo do sistema de manufatura responsável pela capacidade de lidar com um certo nível de variação entre os processos podendo realizar comutações de atividades dos processos sem interrupções na produção. Para ser flexível, um sistema de manufatura necessita possuir as seguintes capacidades:

a) Identificação das diferentes unidades de trabalho para realizar a operação correta;

b) Rapidez na troca de instruções operacionais;

c) Rapidez no "setup" das máquinas.

Santos Filho [2000a] define o conceito de flexibilidade como sendo a capacidade de um determinado sistema adaptar-se a diferentes dinâmicas de seus processos, considerando diferentes abrangências, isto é:

a) Capacidade de processar diferentes elementos;

b) Capacidade de gerenciar fluxos alternativos para movimentação dos elementos no sistema;

c) Capacidade de alocação dinâmica das estações de trabalho para conduzir a execução de diferentes processos segundo rotas alternativas;

d) Capacidade de manutenção e atualização das funções do sistema computacional.

2.4 Sistemas Produtivos Flexíveis

Considerando as definições de manufatura, automação e flexibilidade, a quantidade a ser produzida define o nível de flexibilidade e de automação requerida para que um SPFs produzir-se de forma eficiente [Santos Filho, 2000a]. A figura 2.8 ilustra uma associação dos conceitos de flexibilidade, automação e SPs.

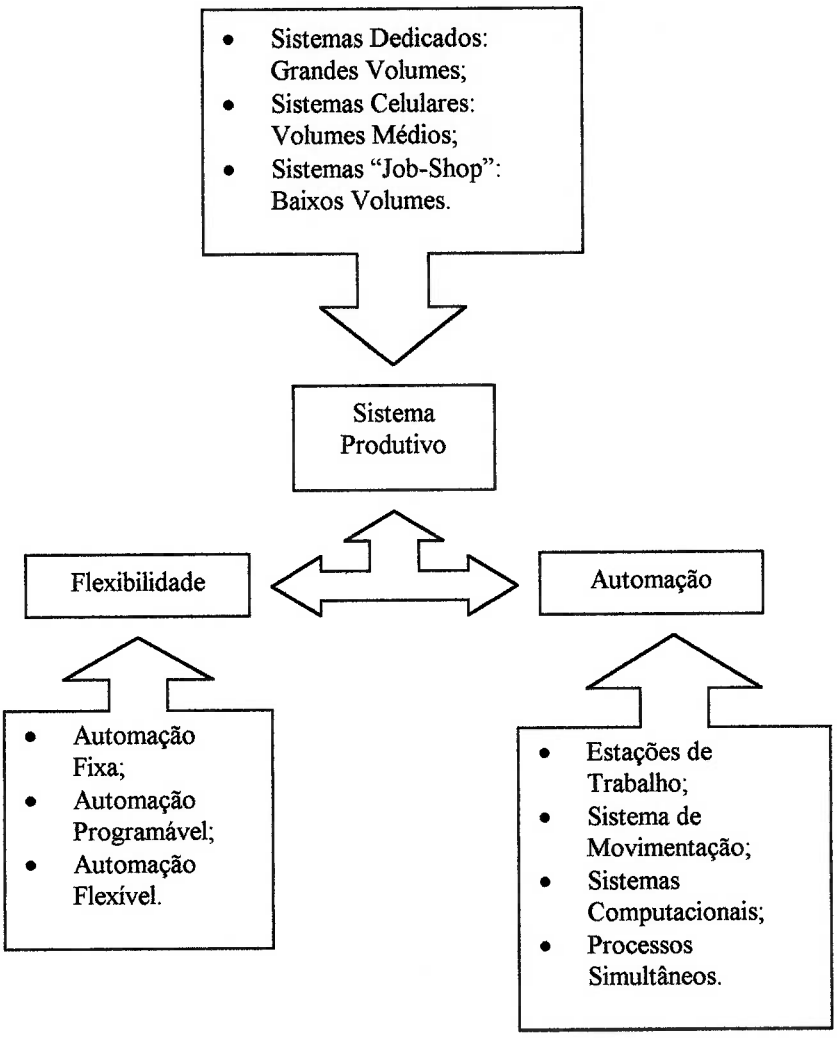


Figura 2.8 Interação entre conceitos de sistema produtivo, automação e flexibilidade [Santos Filho, 2000a].

Em um sistema produtivo onde o objetivo é a produção de grandes volumes, geralmente não é necessário uma automação flexível, sendo adequado uma automação fixa. Quanto ao tipo de recurso, estações de trabalho com algum grau de flexibilidade seria o suficiente. Um exemplo deste sistema poderia ser uma fábrica típica de autopeças.

Em um sistema onde o objetivo é a produção de volumes médios, uma automação programável com recursos de sistema de movimentação e alguns sistemas computacionais seria o indicado, como por exemplo a produção em lotes na indústria farmacêutica. Uma vez que para cada lote é necessário interromper a produção e realizar o "setup" e a manutenção das máquinas por exemplo.

E, por fim, a produção de baixos volumes requer uma automação flexível para ser capaz de atender a diferentes processos simultâneos resultando em uma organização de produção do tipo "job-shop".

Kalpakjian e Schmid [2000] define os SPFs como uma integração de elementos importantes da manufatura em um sistema altamente automatizado. Groover [2000] define da mesma forma e acrescenta:

> *"... interconectado por um sistema de manipulação e armazenamento de materiais, e controlado por um sistema distribuído de computadores ..."*

O objetivo do SPF é produzir simultaneamente um determinado conjunto de produtos de forma eficiente e eficaz. Neste contexto, pode ser necessário adaptar-se rapidamente

a situação de inclusão ou exclusão de novos produtos, e cumprir metas de maximização da utilização dos recursos e minimização dos tempos de "setup" e reprogramação [Groover, 2000].

A cada produto associa-se um processo de fabricação que pode ser descrito por um conjunto seqüencial de etapas que devem ser executadas. As etapas dos processos possuem duas interpretações possíveis [Santos Filho, 2000a]:

a) Ação ou conjunto de ações que transformam o estado do elemento;

b) Prestação de um serviço a este elemento e que implica também em alterar o estado do elemento que está recebendo este serviço.

Desta forma, um SPF pode ser definido como:

- Um agrupamento de múltiplas estações de trabalho automatizadas, interligados por um sistema automatizado de movimentação e transporte de peças e materiais capazes de realizarem diferentes percursos entre as estações, e controlado por computador, ou;

- Um agrupamento de postos de trabalho de prestação de serviços interligados por uma lógica de movimentação baseada em processos.

CAPÍTULO 3
SISTEMA DE CONTROLE PARA SPFS

3.1 INTRODUÇÃO

No Capítulo 2 foi definido que um sistema é formado por um conjunto de elementos que executam determinadas funções para atingir um determinado objetivo. Para que as funções sejam executadas a contento, o sistema necessita ser controlado, impondo-se desta forma, um comportamento desejado em respeito à ocorrência de sinais de entrada [Cassandras, 1993].

Um SPF executa múltiplos processos simultaneamente compartilhando um conjunto finito de recursos. O controle de um único processo deve garantir o seqüenciamento das atividades que compõem o processo em questão. Porém, existem outros processos que devem possuir as mesmas garantias, isto é, garantir que para cada atividade de cada um dos processos haverá a disponibilidade do recurso necessário.

Uma vez que todos os processos compartilham um mesmo conjunto finito de recursos, o sistema de controle deve considerar o seqüenciamento de cada processo e a interação entre esses processos.

Outros aspectos importantes do sistema de controle é:

A) Quanto ao tipo de controle:

- Controle de malha aberta;

- Controle de malha fechada.

B) Quando à forma de controle:

- Controle manual;
- Controle automático.

C) Quanto à classe de controle:

- Controle qualitativo;
- Controle quantitativo.

Neste capítulo é apresentado uma breve descrição do que vem a ser um sistema de controle de malha aberta e de malha fechada, as formas de controle manual e automático e as classes de controle qualitativo e quantitativo no contexto de SPFs. Em seguida é apresentado os aspectos relacionados ao comportamento dinâmico do sistema de controle que torna os SPFs difíceis de serem controlados.

3.2 Tipo, Forma e Classe de Controle

Primeiramente é necessário apresentar as definições fundamentais que se perpetuam ao longo do texto para que haja uma padronização de conceitos:

- Set-point: Representa o valor desejado ou valor de referência. É aquele valor que se deseja para a saída de um sistema;

- Variável Controlada: É o sinal de saída do sistema, isto é, uma grandeza ou condição que é medida e controlada em um sistema [Ogata, 2000];

- Variável Manipulada: É o sinal de entrada do sistema, isto é, uma grandeza ou condição que é variada pelo controlador de modo a afetar o valor da variável controlada [Ogata, 2000];

- Distúrbio ou Perturbação: São sinais que tendem a afetar de modo adverso o valor da variável de saída. Os distúrbios ou perturbações podem ser (i) internos, quando o sinal é gerado internamente no sistema e, (ii) externos quando o sinal é gerado externamente ao sistema e se comportam como um sinal de entrada no sistema [Ogata, 2000];

- Feedback: É um conjunto de informações disponíveis sobre o comportamento do sistema que é realimentado para um contínuo ajuste na entrada do controle [Cassandras, 1993];

- Feedforward: Consiste em injetar na entrada do processo um sinal proporcional a alguma perturbação externa relevante com o objetivo de reduzir os efeitos da perturbação [Moraes e Castrucci, 2001];

- Objeto de Controle: É qualquer objeto físico a ser controlado cuja finalidade é desempenhar uma determinada operação [Ogata, 2000].

3.2.1. Sistema de Controle de Malha Aberta (SCMA)

Em um sistema de controle de malha aberta os sinais percorrem somente um sentido, isto é, da entrada do sistema em direção à saída. Nesta arquitetura de sistema, não se mede o sinal de saída nem tampouco este sinal é enviado de volta para comparação com o sinal de entrada, ou seja, o sinal de saída é determinado unicamente em função do sinal de entrada [Groover, 2000]. Os elementos básicos de um sistema de controle de malha aberta são: parâmetro de entrada, controlador, atuador, objeto de controle e sinal de saída (figura 3.1).

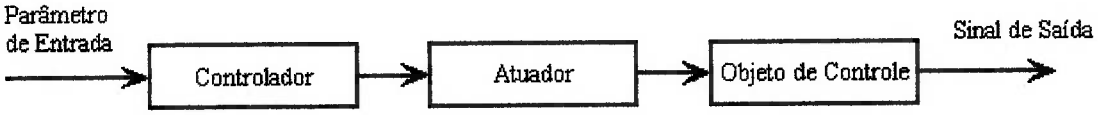


Figura 3.1 Sistema de controle de malha aberta.

Um exemplo de sistema de controle de malha aberta é o controle temporizado de diversas atividades de uma máquina de lavar roupas tradicional. Por exemplo, neste sistema não existe uma medição da saída para que o controle do grau de limpeza das

roupas seja feito com maior precisão, ou seja, a cada sinal de referência na entrada corresponde a uma condição de operação fixa de acordo com seqüenciadores e temporizadores que associa cada ciclo um conjunto de tempos pré-definidos pelos projetistas.

3.2.2. Sistema de Controle de Malha Fechada (SCMF)

O sistema de controle de malha fechada ou sistema de controle Feedback ou por realimentação mantém uma relação preestabelecida entre o sinal de saída e um determinado sinal de referência. O objetivo é comparar os sinais e utilizar a diferença como meio de controle [Groover, 2000], isto é, o sinal de saída é realimentado para a entrada do sistema para modificar a ação do controlador de modo que a saída do sistema se mantenha sempre próxima de um valor desejado. Os elementos básicos de um sistema de controle de malha fechada são: parâmetro de entrada, controlador, atuador, objeto de controle, sensores e sinal de saída (figura 3.2).

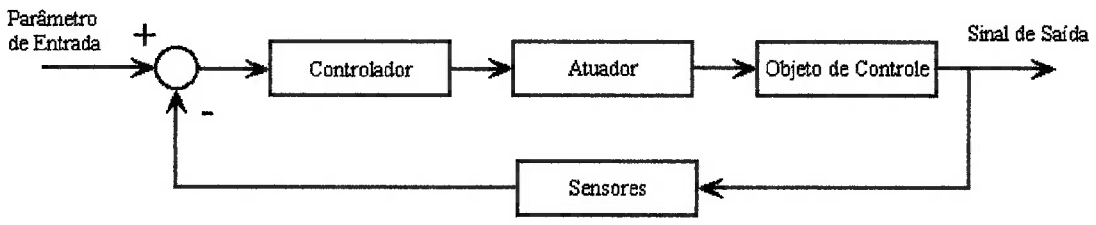


Figura 3.2 Sistema de controle de malha fechada com realimentação.

O sinal realimentado da saída é conseguido mediante a utilização de sensores. Desta forma, qualquer perturbação externa que cause alguma alteração na saída do sistema,

faz com que o controlador receba essa informação dos sensores e corrija o problema [Cassandras, 1993].

Um exemplo de sistema de controle de malha fechada é o sistema de ar-condicionado. Nesse sistema, o valor desejado de temperatura do ambiente é escolhido por meio de um seletor do aparelho que funcionará de modo a alcançar e manter essa temperatura. Existe uma constante monitoração da temperatura no ambiente. Atingido-se a temperatura selecionada, automaticamente o aparelho é desligado. Se, por acaso, o ambiente for aquecido, o sensor informará esse efeito e o aparelho retorna ao funcionamento de modo que a variável controlada (temperatura) fique sempre próxima do valor desejado ("Set-point").

3.2.3. Comparação entre SCMA e SCMF

Os SCMAs são indicados em sistemas onde as entradas são conhecidas antecipadamente no tempo e não há perturbações externos e/ou internos. Os SCMAs são mais fáceis e de menor custo para serem implementadas porém, necessitam de uma calibração inicial para exercerem a função de controle uma vez que os sinais de saída não são medidos e muito menos comparado com os sinais de entrada [Ogata, 2000].

Os SCMFs são indicados em sistemas onde há perturbações. Cassandras [1993] apresenta as seguintes vantagens na utilização dos SCMFs:

a) O comportamento desejado do sistema torna-se menos sensível às perturbações inesperadas;

b) O comportamento desejado do sistema torna-se menos sensível aos possíveis erros no valor dos parâmetros assumidos no modelo;

c) A saída pode automaticamente seguir ou localizar o sinal de referência desejado, buscando continuamente minimizar a diferença ou erro;

As desvantagens da utilização do SCMF são [Cassandras, 1993]:

a) São necessários sensores ou outros tipos de equipamentos para monitorar a saída e alimentar o controlador com essas informações;

b) A realimentação (Feedback) requer esforços (medições no sistema) que podem afetar adversamente o desempenho global do sistema;

c) A realimentação (Feedback) pode criar outros problemas indesejáveis no comportamento do sistema enquanto procura corrigir outros erros.

3.2.4. Sistemas de Controle Manual

Considerando por exemplo o sistema de controle de nível de água de um tanque, conforme apresentado na figura 3.3, o sinal de saída é o nível de água do tanque.

O controlador é um elemento humano que abre ou fecha a válvula para que o tanque mantenha um nível constante. Este tipo de sistema é chamado de sistema de controle de malha fechada manual.

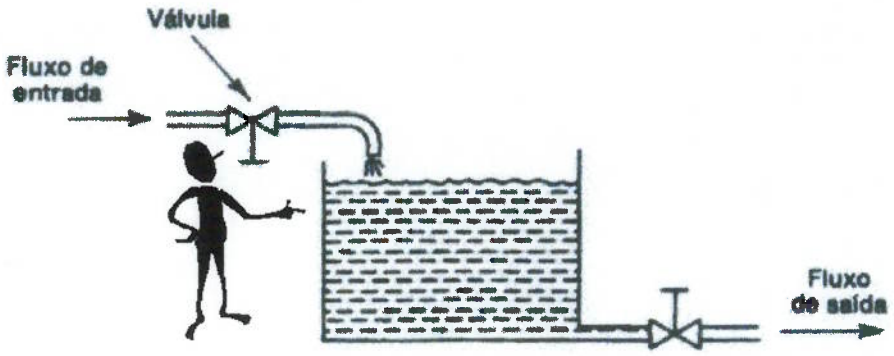


Figura 3.3 Controle de nível de água.

3.2.5. Sistema de Controle Automático

Um sistema de controle torna-se automático quando o operador humano é substituído por um controlador automático, conforme apresentado na figura 3.4.

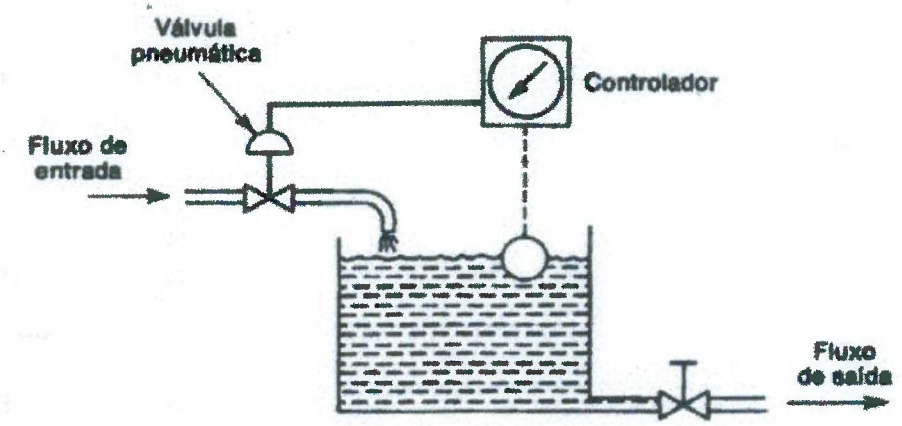


Figura 3.4 Controle automático de nível.

O controle automático é a base tecnológica para a realização da automação. Pode ser dividido em duas classes [Miyagi, 1996]:

- Sistema de Controle Quantitativo;
- Sistema de Controle Qualitativo.

3.2.5.1. Sistema de Controle Quantitativo

Utiliza técnicas para controlar sistemas que são governados através de valores mensuráveis. Possuem uma quantidade infinita de informações analógicas e/ou contínuas, manipulando informações quantitativas (tabela 3.1). Uma das técnicas de implementação é o controle de Sistema de Variáveis Contínuas (SVC). A figura 3.5 apresenta uma ilustração de um sistema de controle SVC de malha fechada.

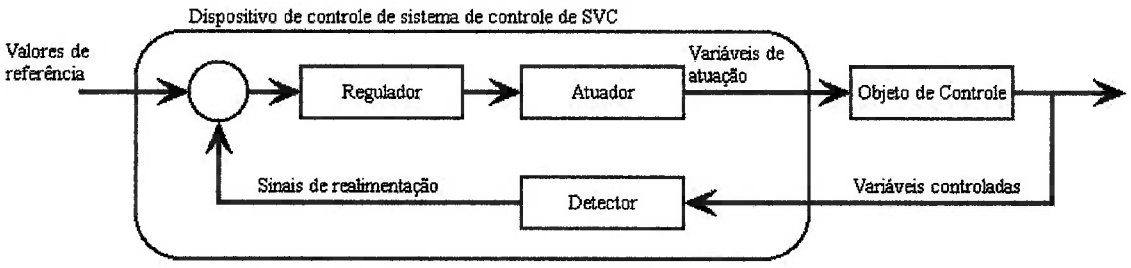


Figura 3.5 Sistema de controle SVS de malha fechada.

3.2.5.2. Sistema de Controle Qualitativo

Utiliza técnicas para controlar sistemas que possuem estados discretos, ou seja, possuem um número finito de estados que podem ser assumidos manipulando informações qualitativas (tabela 3.1). Uma das técnicas de implementação é o controle de Sistemas a Eventos Discretos (SED). A figura 3.6 apresenta a ilustração de um sistema de controle SED de malha fechada. Nos sistemas de controle de SED, o valor de referência é substituído pelo comando de tarefas.

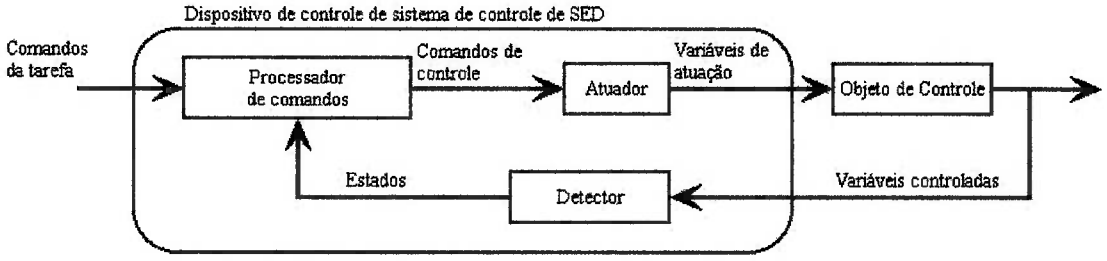


Figura 3.6 Sistema de controle SED de malha fechada.

Tabela 3.1 Controle Automático (SVC x SED)

Controle Quantitativo Controle SVC	Em geral, o objeto de controle manipula informações contínuas;
	É um controle efetivo para o controle de variáveis físicas como fluidos na indústria de processos;
	Envolvem conceitos de controle com realimentação negativa, controle de malha fechada;
	A estrutura de controle é geralmetne em malha fechada.
Controle Qualitativo Controle SED	Em geral o objeto de controle manipula informações discretas;
	É um controle imprescindível para o controle de processos que ocorrem na indústria de manufatura;
	Este controle envolve o cotnrole qualitativo e o processamento do comando de controle;
	A estrutura de controle não é necessariamente em malha fechada.

3.3 Modelo do Sistema de Controle

Dos itens anteriores, classifica-se o **sistema de controle para SPFs** como sendo um **controle de malha fechada**, de forma **automática** e **qualitativa**.

Segundo Miyagi [1996] o termo controle pode ser interpretado como a aplicação de uma ação pré-planejada para que aquilo que se considera como objeto de controle atinja certos objetivos, resultando na composição de um comportamento dinâmico desejado ao sistema. Portanto, o sistema de controle realiza funções preestabelecidas para atingir objetivos predeterminados.

No contexto de SPFs, tais funções são basicamente a execução de atividades e/ou operações através dos vários níveis de controle.

A figura 3.7 apresenta uma estrutura tradicional de sistema de controle conforme Miyagi [1996]. O sistema de controle interage com o objeto de controle e o operador através dos dispositivos de atuação, detecção, comando e monitoração.

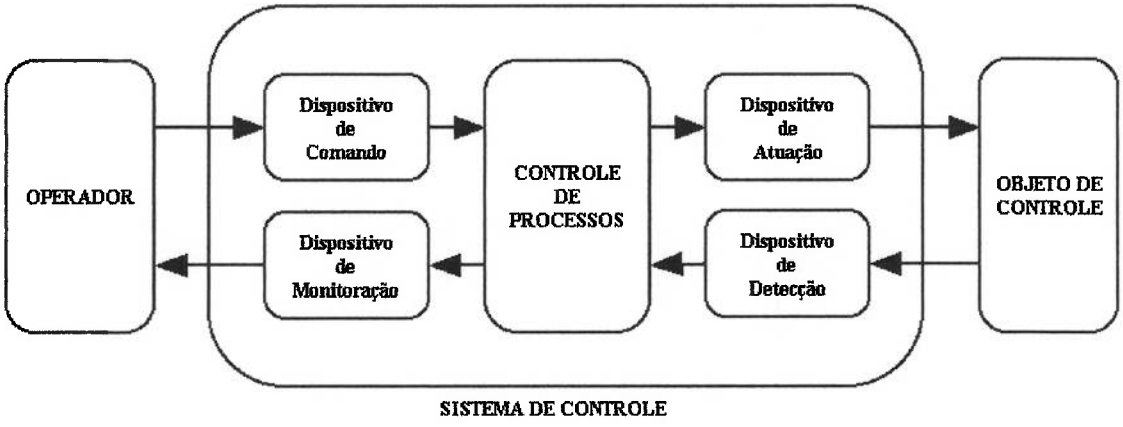


Figura 3.7 Estrutura do sistema de controle.

Neste trabalho é adotado o modelo do sistema de controle antropocêntrico proposto por Santos Filho [1998; 2000a], ou seja, há a necessidade de tornar o elemento humano como parte integrante do sistema.

Os sistemas antropocêntricos são sistemas feitos pelo homem e para o homem ("man-made systems") [Ito, 1991; Santos Filho, 1998; Santos Filho, 2000a]. São sistemas que incorporam novas tecnologias em que o elemento humano, possuidor do conhecimento, participa na especificação de requisitos, desenvolvimento, implementação e monitoramento e, quando necessário, executando ações que interferem na dinâmica do sistema. Na figura 3.8 é apresentada a arquitetura básica de uma sistema de controle antropocêntrico.

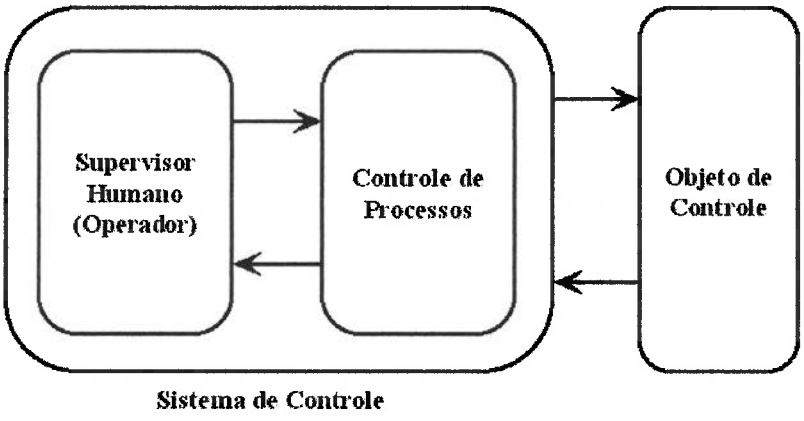


Figura 3.8 Estrutura do sistema de controle antropocêntrico.

A partir desta arquitetura são destacadas as seguintes observações [Miyagi, 2000; Santos Filho, 1998]:

a) O elemento humano é responsável pelo monitoramento do processo global;

b) Uma vez detectado uma restrição que pode levar o sistema a um estado indesejável, o elemento humano possui a capacidade de iniciar uma análise dos efeitos que esta mudança pode causar ao sistema;

c) Realizada a análise, o elemento humano pode reprogramar o sistema de controle adequadamente incorporando as mudanças necessárias para que o sistema evolua dinamicamente de forma desejada.

3.4 Complexidade do Sistema de Controle para SPFs

O conceito de complexidade é definida de várias formas em diferentes contextos porém, não há uma definição geral o suficiente que seja válido para todas as situações onde existe a complexidade [Calinescu et al., 2000].

O termo "complexidade" vem do latim, *complexus*, que significa *entrelaçado* ou *torcido junto*, isto é, são necessários duas ou mais diferentes partes ou componentes que estão de algum modo interligados formando uma estrutura estável [Palazzo e Castilho, 1998].

Em Edmonds [1995] faz-se a seguinte definição:

> *" A complexidade é a propriedade da representação em que o comportamento global é difícil de ser formulado mesmo possuindo-se todas as informações dos componentes e suas inter-relações ".*

Calinescu et. al. [1997] distingue a complexidade em:

- Complexidade estrutural: A causa da complexidade pode ser devido à variabilidade inerente ao sistema, isto é, a superposição dos processos (tipos de produtos, seqüências das etapas de fabricação para produzir cada produto, entre outros) sobre os recursos necessários para a produção dos mesmos (instâncias e tipos de recursos, tempo de setup, "lead times", tempo ocioso, medida de performance, entre outros);

- Complexidade operacional: A causa da complexidade pode ser devido à incerteza que a variabilidade pode gerar, isto é, os problemas que podem surgir na planta ao iniciar o processo de produção. Por exemplo, desarranjos da planta, escassez, absentismo e desequilíbrio de fluxos. Estes tipos de efeitos podem gerar filas de processos.

Dentro do contexto de SPFs, Santos Filho [1998; 2000a] apresenta que a complexidade é uma propriedade que o modelo de um sistema real pode possuir, isto é, nos SPFs a complexidade pode surgir quando é imposto um determinado comportamento dinâmico a estes sistemas.

Controlar um SPF é impor um comportamento desejado preestabelecido ao sistema. Impor um comportamento dinâmico para um único processo é relativamente simples, pois envolve apenas a garantia do seqüenciamento das etapas predeterminadas do processo. Em um SPF, onde diversos processos podem ser executados simultaneamente com um alto grau de compartilhamento de recursos, não é mais possível determinar o comportamento do sistema de modo puramente seqüencial, isto é, no contexto global o

sistema torna-se complexo porque não é possível predefinir o comportamento dinâmico desejado.

A complexidade do processo global deve-se a dois fatores:

- Devido ao fato dos SPFs pertencerem à classe de SED, isto é, são dirigidos por eventos e, portanto, não é possível predeterminar o instante em que ocorrerá um determinado evento;

- A execução de processos simultâneos com compartilhamento de um conjunto finito de recursos.

Desta forma passam a coexistir dois graus de indeterminismo [Santos Filho, 2000a]:

- Indeterminismo em relação ao tempo: Não é possível predeterminar quando é que um certo evento irá ocorrer;

- Indeterminismo em relação à seqüência de ocorrência de eventos: Não é possível determinar qual evento precede outro.

Desta forma, apesar de conhecido o comportamento individual de cada processo, isto é, um conjunto de etapas seqüenciais predeterminadas, não é possível descrever qual é o comportamento do sistema como um todo a partir do momento em que vários processos são executados simultaneamente. Portanto, a somatória das partes não é igual ao todo, ou seja, o todo é a somatória das partes mais a interação entre outras partes [Santos Filho, 2000a]. Em síntese, esta é uma das características presentes em sistemas complexos.

CAPÍTULO 4
"DEADLOCK" EM SISTEMAS PRODUTIVOS FLEXÍVEIS

4.1 INTRODUÇÃO

O auto-travamento ou "deadlock" é caracterizado quando o fluxo das atividades são permanentemente impedidos devido à indisponibilidade de materiais, recursos e/ou informações. Este fenômeno pode ocorrer em sistemas que executam atividades paralelas, isto é, a execução das atividades é simultânea e há compartilhamento de um mesmo conjunto de materiais, recursos e/ou informações. Desta forma, o "deadlock" pode ocorrer em diferentes contextos. Tradicionalmente, este problema é estudado em sistemas operacionais e banco de dados, envolvendo, portanto, a área de ciências da computação.

A figura 4.1 apresenta um exemplo em que o fenômeno ocorre em sistemas operacionais. Admitindo-se que cada processo tem o uso exclusivo do recurso e somente irá liberá-lo quando alocar o próximo recurso, pode-se observar que:

- O processo *A* solicita e consegue permissão para utilizar o recurso "scanner", enquanto o processo *B* solicita e consegue permissão para utilizar o recurso "dispositivo de armazenamento";

- A seguir, o processo *A* solicita a utilização do recurso "dispositivo de armazenamento" mas a solicitação não será atendida até que o processo *B* o libere;

- Porém, o processo *B* ao invés de liberar o recurso "dispositivo de armazenamento", solicita o recurso "scanner";

- A partir deste ponto, ambos os processos estão aguardando indefinidamente por um recurso utilizado por outro processo. Isto é uma situação de "deadlock".

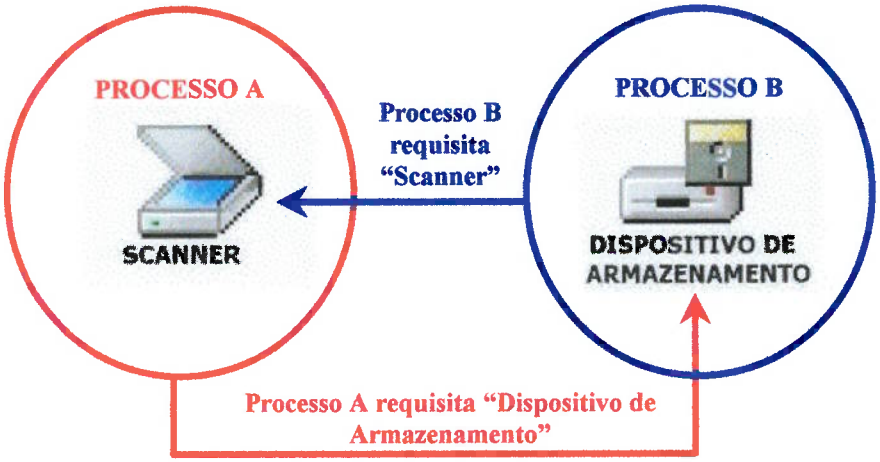


Figura 4.1 Exemplo de "deadlock" em sistemas operacionais.

Um outro exemplo comum é encontrado no tráfego de automóveis [Isloor e Marsland, 1980]. Considere a situação da figura 4.2, onde quatro automóveis (*A*, *B*, *C* e *D*) chegam aproximadamente no mesmo instante em um cruzamento.

- Cada veículo pode virar à direita, à esquerda ou manter a sua trajetória. Por exemplo, para o veículo **A** virar à direita deverá ocupar a posição *1*, para virar à esquerda deverá ocupar as posições *1*, *2* e *3*, e para seguir em frente deverá

ocupar as posições *1* e *2*. Esta regra vale para os demais veículos nas respectivas posições;

- Uma das situações de "deadlock" seria se os quatro veículos seguissem a sua trajetória assumindo a sua respectiva posição à sua frente;

- **A** ocupando a posição **1** não poderia assumir a posição **2** pois está sendo ocupada por **B**; por sua vez **B** ocupando a posição **2** não poderia assumir a posição **3** pois está sendo ocupada por **C** e assim por diante.

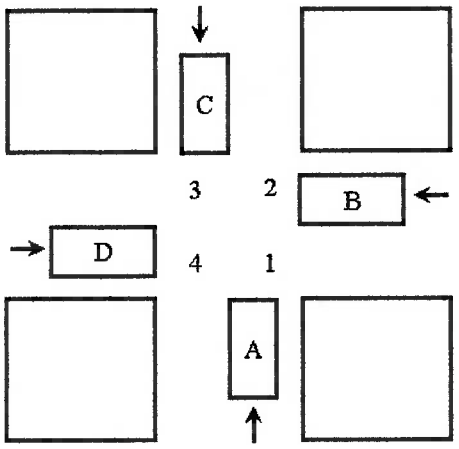


Figura 4.2 Exemplo de "deadlock" em tráfego.

Este capítulo prossegue apresentando uma revisão bibliográfica dos conceitos de alocação de recursos e "deadlock", seguida de uma discussão sobre as condições suficientes e necessárias para a ocorrência de "deadlock" e quais os métodos para abordar o problema.

4.2 Alocação de Recursos e "Deadlock"

Os SPFs são passíveis de apresentar o problema de "deadlock". No capítulo 2 foi definido que os SPFs são caracterizados por produzirem uma variedade de produtos simultaneamente, modificando rapidamente as configurações de acordo com o planejamento da produção e compartilhando um conjunto finito de recursos. A execução de processos paralelos com acentuado compartilhamento de recursos, podem gerar um auto-travamento do sistema, ou "deadlock", conforme visto anteriormente. Para entender o problema de "deadlock" em SPFs deve-se examinar a alocação de recursos nesses sistemas.

A alocação de recursos em SPFs é uma função primária de controle que depende do cenário industrial em questão. Desta forma, os SPFs podem ser classificados de acordo com o tipo de alocação de recursos necessários no sistema [Lawley et al., 1997]:

- "Single Resource Allocation". Um processo é formado por várias etapas e em cada etapa é requerido apenas um único recurso para a sua execução;

- "Conjunctive Resource Allocation". Um processo requer um conjunto de recursos antes da sua execução. Por exemplo, para que um processo possa executar a próxima etapa é necessário estar disponível um AGV para o transporte além do próximo recurso disponível;

- "Disjunctive Resource Allocation". Um processo possui várias rotas alternativas de acordo com a aquisição de determinado recurso de um conjunto de recursos possíveis. Por exemplo, em uma determinada etapa do processo estaria disponível

um conjunto de máquinas. Dependendo da máquina que for alocada, o processo prosseguirá por uma determinada rota;

- "Conjunctive/Disjunctive Resource Allocation": A alocação de recursos é uma combinação de ANDs e ORs, isto é, além de ser necessário a disponibilidade de um AGV para o transporte, o processo pode executar a próxima etapa em um determinado recurso de um conjunto de recursos possíveis.

Com relação ao "deadlock", em Cho [1993] é apresentada a seguinte classificação:

- "Part Flow Deadlock". Corresponde à situação onde os processos estão em uma cadeia cíclica aguardando a liberação de recursos alocados por outros processos que pertencem à cadeia propriamente dita. A figura 4.3 apresenta um exemplo de "Part Flow Deadlock". O recurso *R1* está sendo utilizado pelo processo *A* e quando finalizar a etapa irá requisitar a utilização do recurso *R2* que está sendo utilizado pelo processo *B*. O processo *B* ao finalizar a utilização do recurso *R2* irá requisitar a utilização do recurso *R3* que está sendo utilizado pelo processo *C*. O processo *C* ao finalizar a utilização do recurso *R3* irá requisitar a utilização do recurso *R1* que está sendo utilizado pelo processo *A*. Desta forma, o exemplo se encontra em estado de "deadlock";

- "Impending Part Flow Deadlock". Corresponde à situação em que há fluxo de processos porém o sistema evolui para o estado de "deadlock" após a execução de algumas etapas. A figura 4.4 apresenta um exemplo de "Impending Part Flow Deadlock". O recurso *R1* está sendo utilizado pelo processo *A* e quando finalizar a

etapa irá requisitar a utilização do recurso *R2*. O recurso *R3* está sendo utilizado pelo processo *B* e quando finalizar a etapa irá requisitar a utilização do recurso *R2*. O recurso *R2* encontra-se disponível porém quando um dos processos alocá-lo, o sistema evoluirá para o estado de "deadlock";

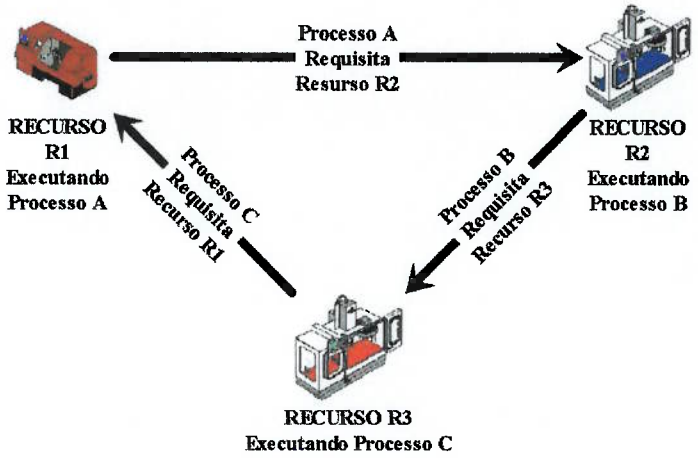


Figura 4.3 "Part Flow Deadlock".

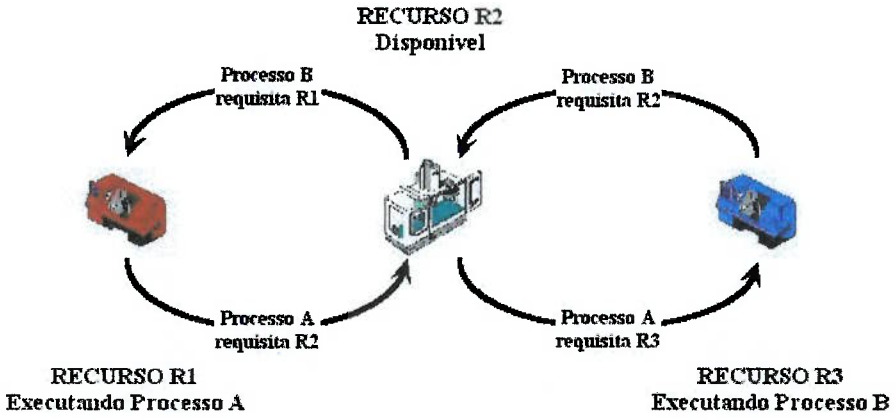


Figura 4.4 "Impending Part Flow Deadlock".

- "Processing Resource Deadlock". Corresponde à situação em que os processos estão travados devido à falta de dispositivos ou ferramentas necessários para a execução

da etapa do processo. A figura 4.5 apresenta um exemplo em que há três processos (*A*, *B* e *C*), três recursos (*R1*, *R2* e *R3*), duas ferramentas (*F1* e *F2*) e um dispositivo (*D1*). O processo A está utilizando o recurso *R1* e a ferramenta *F1* porém necessita da ferramenta *F2* que está sendo utilizada pelo processo *B*. O processo *B* está utilizando o recurso *R2* e a ferramenta *F2* porém necessita do dispositivo *D1* que está sendo utilizado pelo processo *C*. O processo *C* está utilizando o recurso *R3* e o dispositivo *D1* porém necessita da ferramenta *F1* que está sendo utilizada pelo processo *A*;

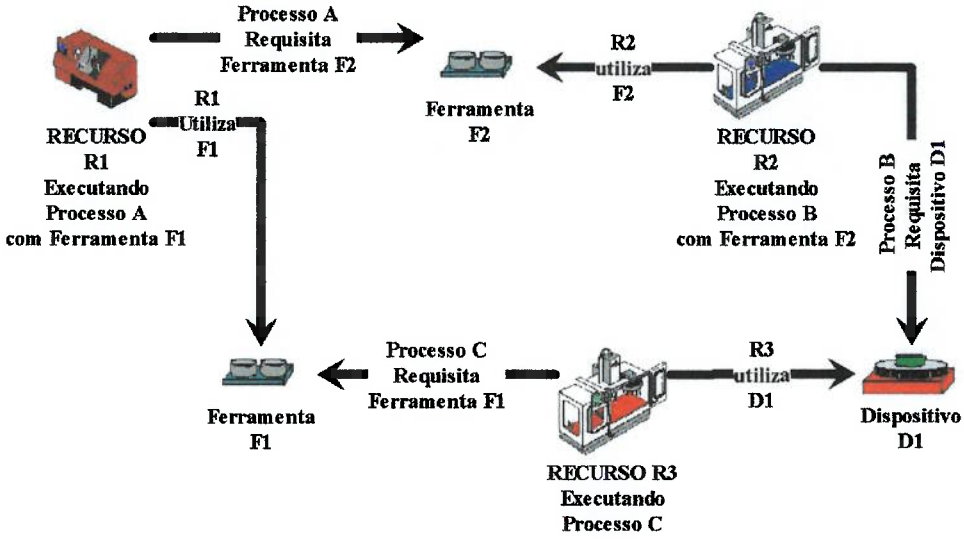


Figura 4.5 "Processing Resource Deadlock".

- "Material Handler Deadlock". Corresponde à situação em que os processos estão travados devido à falta de matéria-prima, isto é, referente à alocação, manipulação e/ou transporte da matéria-prima. A figura 4.6 apresenta um exemplo de "Material

Handler Deadlock" em um AGVS[1] (Automated Guided Vehicle System) que encontra-se em estado de "deadlock".

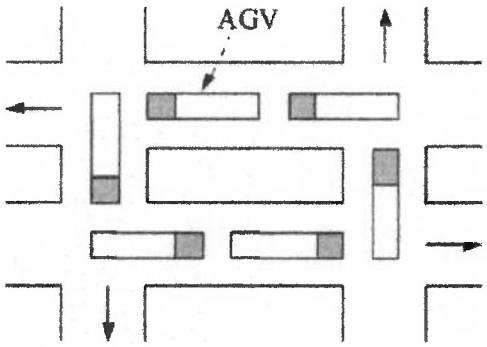


Figura 4.6 "Material Handler Deadlock".

Neste trabalho, é abordado o problema de "deadlock" em "**Part Flow Deadlock**" com "**Single Resource Allocation**", ou seja, cada etapa de processo utiliza apenas um único recurso para ser executado. Por exemplo, considerando-se sete processos *A*, *B*, *C*, *D*, *E*, *F* e *G* sendo executados simultaneamente, compartilhando entre si seis recursos *R1*, *R2*, *R3*, *R4*, *R5* e *R6*, de maneira que a seqüência de utilização dos recursos pelos processos é apresentada na tabela 4.1 e na figura 4.7.

Tabela 4.1 Seqüência de utilização dos recursos pelos processos.

	Processo	Seqüência de utilização de recursos
1	A	R1 e R2
2	B	R2 e R1
3	C	R6, R3 e R2
4	D	R3 e R5
5	E	R5 e R6
6	F	R4 e R3
7	G	R1 e R4

[1] "Automated Guided Vehicle System" (AGVS) é um sistema de manipulação de materiais que utiliza veículos auto-guiados por rotas pré-estabelecidas [Groover, 2000].

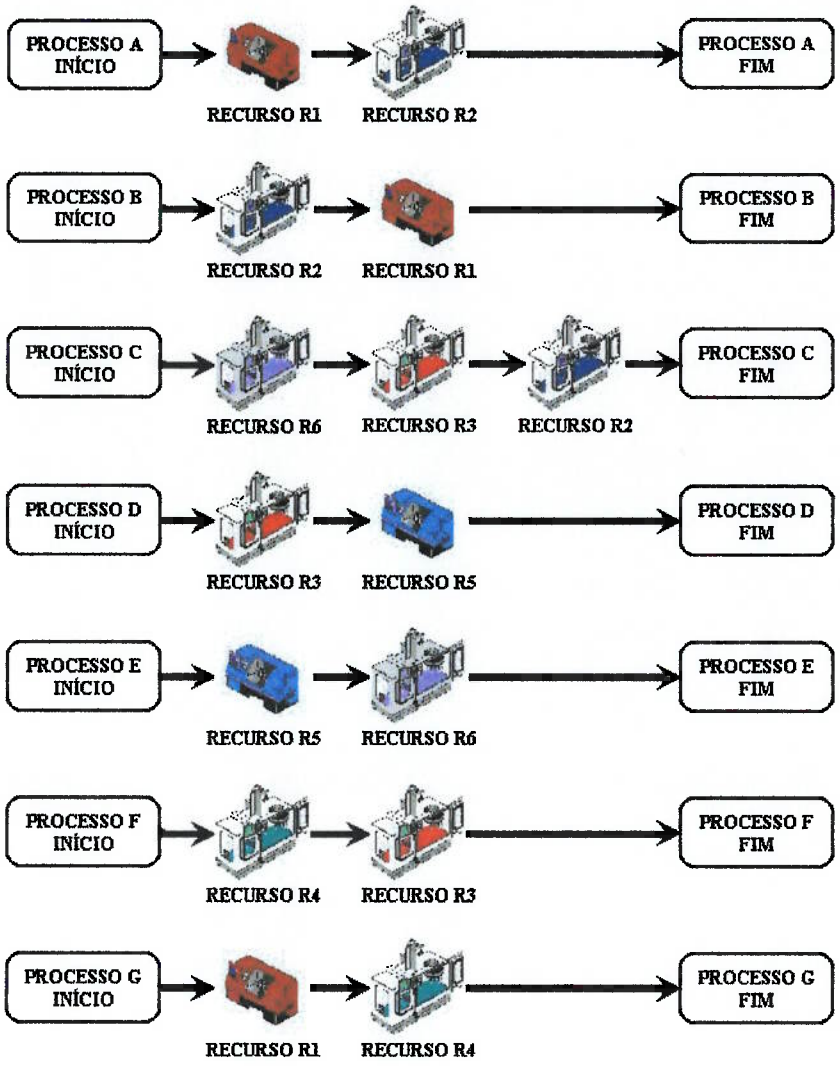


Figura 4.7 Seqüência de utilização dos recursos pelos processos individualmente.

O processo *A* utiliza seqüencialmente o recurso *R1* e em seguida o recurso *R2*. O processo *B* utiliza o recurso *R2* e recurso *R1*. O mesmo repete-se para os demais processos. A seqüência de eventos necessária para garantir a utilização de recursos é a seguinte [Tanenbaum, 1995]:

(i) Requisita o recurso;

(ii) Utiliza o recurso;

(iii) Libera o recurso.

Assim, cada processo tem o uso exclusivo do recurso de tal forma que somente irá liberar esse recurso quando alocar o próximo recurso. A figura 4.8 apresenta um modelo global da utilização dos recursos pelos processos.

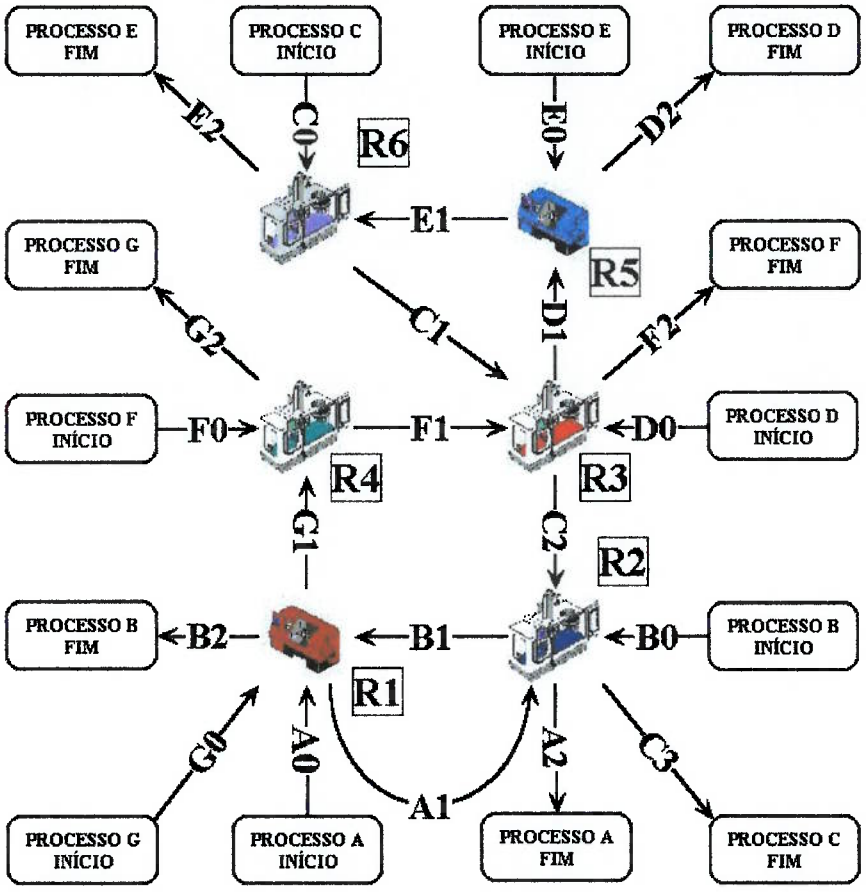


Figura 4.8 Modelo global da utilização dos recursos pelos processos.

Por exemplo, considerando-se a seguinte situação:

- O processo *D* está utilizando o recurso *R3*, o processo *E* está utilizando o recurso *R5* e o processo *C* está utilizando o recurso *R6*. Quando o processo *D* terminar de utilizar o recurso *R3* irá requisitar o recurso *R5*.

- Porém, o recurso *R5* está sendo utilizado pelo processo *E*. Quando o processo *E* terminar de utilizar o recurso *R5*, irá requisitar o recurso *R6*.

- Porém, o recurso *R6* está sendo utilizado pelo processo *C* e, por sua vez, o processo *C* ao terminar a utilização o recurso *R6* irá requisitar o recurso *R3* que, por sua vez, está sendo utilizado pelo processo *D*. Nesta situação, os processos estão em "deadlock".

A partir dos exemplos apresentados, é possível observar que os processos travados aguardam a ocorrência de eventos que devem ser disparados por outros processos, e que também estão aguardando a ocorrência de outros eventos, isto é, a liberação de recursos.

Finalizando esta seção é importante ressaltar o que caracteriza processos travados, processos bloqueados e estado seguro de um sistema. Um estado é dito seguro se não provocar "deadlock", isto é, o sistema pode garantir que todos os processos irão terminar conforme previsto. Por sua vez, para entender-se a diferença entre processos bloqueados e travados considere que enquanto os processos bloqueados aguardam por um evento que certamente irá ocorrer, os processos travados aguardam por um evento que nunca irá ocorrer.

4.3 Condições Suficientes e Necessárias para a Ocorrência de "Deadlock"

Em estudos anteriores foi identificado que existem quatro condições suficientes e necessárias para que se configure uma situação de "deadlock" [Isloor e Marsland, 1980; Banaszak e Krogh, 1990; Viswanadham et al., 1990; Cho, 1993; Tanenbaum, 1995; Santos Filho, 2000a]:

- Mútua exclusão. Cada processo requisita o uso exclusivo do recurso, isto é, cada recurso ou está alocado a um processo ou está disponível;
- Retenção enquanto aguarda. Enquanto aguarda a liberação de um recurso, o processo não libera o recurso alocado por ele;
- Não há preempção. Um recurso poderá ser liberado somente pelo processo que o alocou;
- Espera circular. É uma cadeia cíclica fechada de processos aguardando a liberação de recursos alocados por outros processos pertencentes à mesma cadeia cíclica (ciclo de espera).

Todas as quatro condições devem ser verificados para que o "deadlock" ocorra, portanto, basta que uma das condições seja falsa para que não ocorra o travamento do sistema.

Utilizando novamente o exemplo da figura 4.2, verifica-se que as quatro condições são verdadeiras.

(i) Uma posição pode ser ocupada somente por um único veículo;

(ii) O veículo irá liberar a posição que está ocupando somente quando ocupar a próxima posição;

(iii) A posição poderá ser liberado somente pelo veículo que a está ocupando;

(iv) Pode-se formar uma cadeia cíclica fechada de veículos que aguardam a posição ocupada por outros veículos.

Desta forma é possível observar que este exemplo possui as quatro condições suficientes e necessárias para ocorrência do "deadlock". De forma análoga, se considerarmos as posições como recursos e os veículos como processos no exemplo da figura 4.8, verifica-se que novamente essas quatro condições são observadas.

Por sua vez, verifica-se que nos SPFs as três primeiras condições são inerentes ao sistema, isto é:

(i) Cada recurso somente poderá ser utilizado por apenas um processo;

(ii) O processo somente irá liberar o recurso quando alocar o próximo recurso;

(iii) O recurso é somente liberado pelo processo que detém sua posse.

Desta forma, não é possível evitar a ocorrência dessas três condições, restando apenas a condição de espera circular para ser controlada. Portanto, no presente trabalho, o problema de "deadlock" é abordado controlando-se a condição de espera circular.

4.4 MÉTODOS PARA ABORDAR O PROBLEMA DE "DEADLOCK"

Basicamente, o problema de "deadlock" pode ser abordado de quatro formas diferentes, envolvendo métodos distintos [Isloor e Marsland, 1980; Tanenbaum, 1995; Lawley et al., 1997]:

- Método de Ignorar o "Deadlock";
- Método de Prevenir o "Deadlock" ("Deadlock Prevent");
- Método de Detectar o "Deadlock" e restabelecer ("Deadlock Detection and Resolution");
- Método de Evitar o "Deadlock" ("Deadlock Avoidance").

4.4.1. Método de Ignorar o "Deadlock"

É um método que ignora o problema de "deadlock" propriamente dito. Por exemplo, no sistema operacional UNIX a estratégia em relação aos 'deadlocks' é simplesmente ignorar o problema. Os desenvolvedores desse sistema operacional sabem da existência e provavelmente conhecem formas de tratar os "deadlocks". Porém, o custo computacional envolvido em estabelecer restrições aos usuários quanto ao uso dos recursos do sistemas de modo a eliminar os travamentos não são viáveis em virtude da baixa probabilidade destes tipos de problema ocorrerem [Tanenbaum, 1995].

Entretanto, considerando os SPFs, o custo envolvido em ignorar-se a ocorrência de um travamento não é admitido, uma vez que compromete o controle do sistema.

4.4.2. "Deadlock Prevent"

É uma esquematização estruturada do sistema que garante que pelo menos uma das quatro condições necessárias nunca irá ocorrer. A modelagem é realizada desde o início com o objetivo de prevenir o estado de "deadlock".

Em Viswanadham et al. [1990] é adotada uma exaustiva análise do grafo de atingibilidade do sistema baseado em redes de Petri quanto à alocação de recursos. O grafo de atingibilidade possui dois tipos de transições:

- Transição Imediata;
- Transição Temporizada.

E três classes de marcações no grafo de atingibilidade proposta pelo autor (figura 4.9):

- Marcações 'Vanishing'. As marcações 'Vanishing', representadas por um círculo, são marcações em que pelo menos uma transição imediata está habilitada;
- Marcações Tangíveis. As marcações tangíveis, representadas por um duplo círculo concêntricos, são marcações em que somente as transições temporizadas estão habilitadas;

- Marcações em "Deadlock". As marcações em "Deadlock", representadas por três círculos concêntricos, são marcações que não possuem nenhuma transição habilitada, isto é, está em "deadlock".

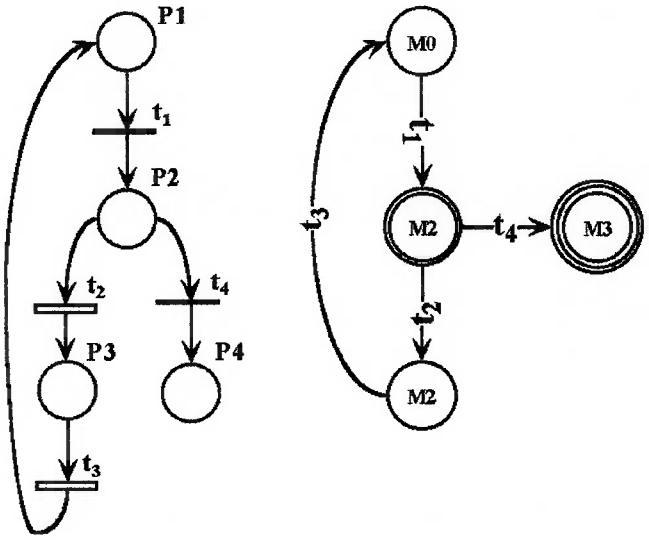


Figura 4.9 Exemplo de modelo em redes de Petri e o respectivo grafo de atingibilidade.

O método de 'deadlock prevent' proposto por Viswanadham et al. [1990] consiste em impossibilitar os disparos de transições que levam o sistema ao travamento.

4.4.3. "Deadlock Detection and Resolution"

Consiste em um método baseado em algoritmos que detectam o estado de "deadlock" e realizam ações para tirar o sistema deste estado. As ações para o restabelecimento do sistema podem ser [Tanenbaum, 1995]:

- Através da Preempção. Paralisação temporária de um ou mais processos e disponibilização do recurso para outro processo. Um exemplo considerando um SPF em estado de "deadlock" seria a transferência temporária de um ou mais processos

travados para um "buffer" (*fila de processos*) liberando desta forma os demais processos;

- Através da Volta ao Passado. Os processos devem ser periodicamente verificados, isto é, registra-se informações do estado dos processos e dos recursos para que possam ser consultadas mais tarde. Uma vez detectado o estado de "deadlock", os processos retornam ao instante anterior ao travamento através das informações contidas na verificação. Esta ação é utilizada em sistemas de gerenciamento de bancos de dados, mantendo assim a consistência dos dados;

- Através da Eliminação de Processos. Eliminação de um ou mais processos em estado de "deadlock". Caso a eliminação de um único processo não surta efeito, pode-se continuar eliminado outros processos até que o sistema saia do estado de "deadlock".

Em Tanenbaum [1995] são apresentados dois algoritmos para a detecção do travamento do sistema de acordo com a seguinte classificação:

- Sistema com uma instância de cada classe de recurso;

- Sistema com várias instâncias de cada classe de recurso.

Para sistemas com uma instância de cada classe de recurso é apresentado o seguinte exemplo, conforme descrito em Tanenbaum [1995]:

✓ O processo A está de posse de R1, e precisa de R2;

- ✓ O processo B não está de posse de nenhum recurso, mas precisa de R4;
- ✓ O processo C não está de posse de nenhum recurso, mas precisa de R2;
- ✓ O processo D está de posse de R5, e precisa de R2 e de R4;
- ✓ O processo E está de posse de R4, e precisa de R6;
- ✓ O processo F está de posse de R3, e precisa de R2;
- ✓ O processo G está de posse de R6, e precisa de R5.

Dado um conjunto de processos e a utilização seqüencial dos recursos, cria-se um grafo de recursos do sistema conforme apresentado na figura 4.10.

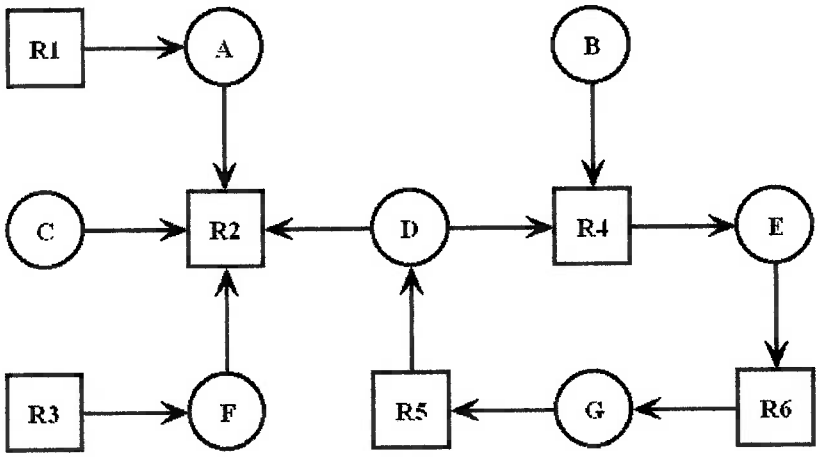


Figura 4.10 Grafo de recursos.

Os quadrados representam os recursos e os círculos representam os processos. A interpretação do grafo é realizada da seguinte forma:

- O arco orientado que sai do quadrado para o círculo significa que o recurso indicado no quadrado está sendo utilizado pelo processo que está indicado no círculo;

- O arco orientado que sai do círculo para o quadrado significa que o processo está requerendo a utilização do recurso indicado no quadrado.

Através do algoritmo (figura 4.11) apresentado por Tanembaum [1995] detecta-se um ciclo de espera ou seja, se tal grafo contiver um ciclo de espera, estará garantida pelo menos uma situação de "deadlock".

Para cada nó N do grafo, realize os seguintes cinco passos, sendo que N é o nó inicial.
1. Inicialize L como uma lista vazia e designe todos os arcos como não marcados.
2. Coloque o nó corrente no fim da lista L e verifique se ele aparece mais de uma vez em L. Se isto for verdade, o grafo contém um ciclo listado em L e o algoritmo deve terminar.
3. A partir deste nó, verificar se existe algum arco não assinalado partindo do nó. Se houver vá para o passo 4, do contrário vá para o passo 5.
4. Escolha aleatóriamente um arco não assinalado, partindo deste nó e o marque. Pegue o próximo nó e volte para o passo 2.
5. Volte para o nó anterior e reinicie no passo 2. Se este nó for o inicial, o grafo não contém ciclos e o algoritmo deve terminar.

Figura 4.11 Algoritmo de detecção para uma instância de cada classe de recurso.

Este algoritmo é uma variação do algoritmo de pesquisa por profundidade e amplitude em grafos [Netto, 2001]. O algoritmo toma cada um dos nós do grafo ordenadamente, isto é, partindo-se da raiz de uma árvore realiza-se uma pesquisa profunda. Cada nó visitado é adicionado a uma lista de nós. Se alguma vez visitar um nó encontrado anteriormente, então foi descoberto um ciclo de espera e o algoritmo é encerrado. Ao exaurir todos os arcos de determinado nó, retorna ao nó anterior. Se voltar à raiz e não

puder continuar, o subgrafo demarcado até o presente nó não possui nenhum ciclo de espera. Sendo esta propriedade verdadeira para todos os nós, o grafo estará livre de ciclos e, portanto, não apresenta condição de "deadlock".

Para sistemas com várias instâncias de cada classe de recurso, utiliza-se um algoritmo baseado em matrizes para detecção de "deadlock" entre n processos, de P_1 até P_n.:

(i) Seja m o número de classes diferentes de recursos, com E_1 recursos da classe 1, E_2 da classe 2 e, genericamente, E_i recursos da classe i, sendo que $1 \leq i \leq m$;

(ii) Seja E um vetor de recursos existentes que fornece o número total de instâncias de cada recurso existente;

(iii) Seja A o vetor de recursos disponíveis, com A_i fornecendo o número de instâncias do recurso i atualmente disponíveis;

(iv) Seja C a matriz de alocação corrente e R a matriz de requisições, onde as linhas (i) representam os processos e as colunas (j) representam os recursos.

Uma importante condição permanece invariante para estas quatro estruturas, isto é, a somatória de todas as instâncias do recurso j que estiverem alocadas, e a este valor somarmos todas as instâncias disponíveis, o resultado é o total de instâncias existentes para cada classe de recursos:

$$\sum_{i=1}^{m} C_{ij} + A_j = E_j \tag{4.1}$$

O algoritmo realiza uma comparação de vetores. Inicialmente, cada processo é considerado como desmarcado. Conforme o algoritmo progride, os processos serão marcados, indicando que estão habilitados a terminar o seu processamento. Ao término do algoritmo se houver algum processo desmarcado então o sistema está no estado de "deadlock". O algoritmo é apresentado na figura 4.12.

1. Procure por um processo demarcado Pi, para o qual a i-ésima linha de R é menor do que a correspondente de A.
2. Se um processo com tais características for encontrado, adicione a i-ésima linha de C a A, marque o processo, e retorne para o passo 1.
3. Se não houver nenhum processo nesta situação, o algoritmo termina.

Figura 4.12 Algoritmo de detecção para várias instâncias de cada classe de recurso.

Dado um sistema exemplo que possui os seguintes recursos e suas respectivas quantidades conforme apresentado na tabela 4.2:

Tabela 4.2 Exemplo de um sistema.

Recurso	Qtd
Unidade de fita	4
Plotters	2
Impressoras	3
Unidade de CD-ROM	1

Neste sistema, há três processos que estão sendo executados simultaneamente da seguinte forma:

- ✓ O processo 1 está utilizando uma impressora. Para a próxima etapa necessita de 2 unidades de fita e uma unidade de CD-ROM;
- ✓ O processo 2 está utilizando 2 unidades de fita e uma unidade de CD-ROM. Para a próxima etapa necessita de uma unidade de fita e uma unidade de impressora;

✓ O processo 3 está utilizando 2 unidades de fita e uma unidade de Plotters. Para a próxima etapa necessita de 2 unidades de fita e uma unidade de Plotters.

Desta forma, tem-se os seguintes vetores e matrizes:

$$E = [4 \quad 2 \quad 3 \quad 1] \tag{4.2}$$

$$A = [2 \quad 1 \quad 0 \quad 0] \tag{4.3}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix} \tag{4.4}$$

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix} \tag{4.5}$$

O algoritmo procura um processo cujas necessidades de recursos possam ser satisfeitas.

Na presente situação, o processo 3 pode ser satisfeito. Então, o processo 3 é marcado e realiza o seu respectivo processamento. Ao término, o processo 3 devolve os recursos tornando-os disponíveis aos demais processos e é feita novamente a verificação dos processos cujas necessidades de recursos possam ser satisfeitas. Neste exemplo não ocorre a situação de "deadlock" pois todos os processos são executados.

Em Kumaran et al. [1994] é proposta a utilização de grafos que representam a alocação de recursos do SPF. O grafo *G*, *G=(V, A, C)*, onde *V* é o conjunto de nós representado os recursos (graficamente é um quadrado), *A* o conjunto de arcos orientados que representam a etapa do processo e *C* é o conjunto de ciclos de espera. A figura 4.13

apresenta um exemplo onde os quadrados representam os recursos, os arcos são as etapas de processo e os círculos são os processos que estão sendo executados em determinado instante de tempo.

Processo	Seqüência
1	A-B-C-D
2	F-E-B-A
3	D-C-B-E-F

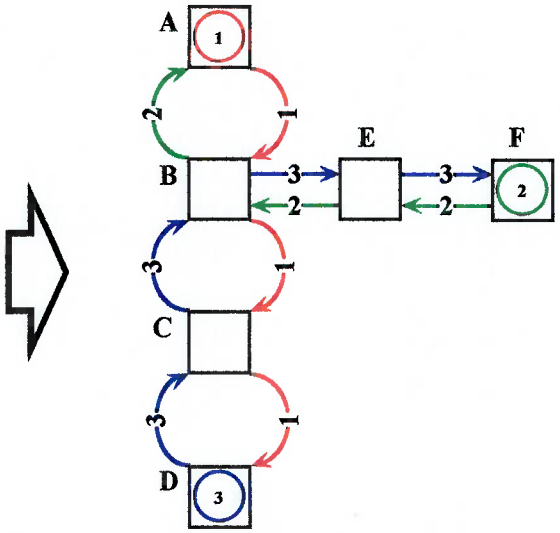


Figura 4.13 Exemplo apresentado em Kumaran et al. [1994].

A figura 4.14 apresenta o algoritmo para a determinação dos ciclos de espera. Este algoritmo é baseado na técnica de pesquisa em profundidade aplicada diretamente no grafo *G*.

Procedure Deadlock-Detection [G]
Entrada Grafo G
Saída Conjunto de processos travados
Passo 1. Determinação dos ciclos de espera (C) em G.
Passo 2. Validadar os ciclos de espera

Procedure Validar[C]
Se o número de processos do ciclo de espera for igual ao número de arcos e número de nós então o ciclo está em 'deadlock' do contrário não está em 'deadlock'.

Figura 4.14 Algoritmo para detecção do "deadlock" conforme Kumaran.

A condição necessária e suficiente para resultar em "deadlock" é quando o número de etapas de processos é igual ao número de arcos e ao número de quadrados que pertencem ao ciclo de espera. Detectado o estado de travamento, o algoritmo é encerrado e inicia-se o restabelecimento do sistema. A ação realizada é a movimentação de etapas dos processos travados para uma fila de espera. Uma vez que o "deadlock" esteja resolvido, o controlador deve esvaziar a fila de espera para disponibilizá-la na eventual reincidência do travamento.

4.4.4. "Deadlock Avoidance"

Consiste em um método baseado em algoritmos que supervisionam os processos quanto à alocação de recursos realizando em tempo real as decisões para evitar o estado de "deadlock", isto é, detecta o estado pré "deadlock" e inibe a alocação de determinados recursos que possam levar o sistema ao travamento. Como visto anteriormente, em SPFs as condições suficientes e necessárias para a ocorrência de "deadlock" podem ser intrínsecas ao sistema, exceto a condição de espera circular. Muitos trabalhos propõem soluções controlando-se essa condição através de um modelo de restrições [Banaszak e Krogh, 1990; Kumaran et al., 1994; Fanti et al., 1995; Fanti et al., 1997; Wu, 1999; Santos Filho, 2001a, Santos Filho, 2000; Santos Filho, 1998].

Em Tanenbaum [1995] é descrito o "algoritmo do banqueiro" proposto por Dijkstra em 1965. Este algoritmo baseia-se nas premissas adotadas por um banqueiro de um pequeno banco para garantir ou não crédito a seus correntistas. A partir de um número

fixo de processos, a cada processo declara-se um número máximo de recursos que serão utilizados durante a execução. A execução é permitida se a soma dos recursos requisitados for menor que os recursos disponíveis no sistema, isto é, no momento de cada solicitação, será verificada se o atendimento desta solicitação leva a um estado seguro. Em caso afirmativo, a solicitação é imediatamente atendida, caso contrário o atendimento é adiado para outra ocasião.

A verificação do estado do sistema é realizada da seguinte forma: a partir da solicitação inicial verifica-se se a quantidade de recursos disponíveis no sistema atendem a pelo menos um processo, então é assumido que a quantidade de recursos utilizadas por esse processo será devolvida, passando então a analisar os demais processos da mesma forma.

Se todos os processos puderem ser executados, o estado é seguro e a solicitação inicial pode ser atendida. Este algoritmo foi proposto para sistemas operacionais e não é eficiente em SPFs pois não leva em consideração o seqüenciamento da alocação e desalocação de recursos pelos processos.

Viswanadham et al. [1990] propõem também um algoritmo de "deadlock avoidance" baseado no mesmo processo de modelagem do sistema apresentado anteriormente na descrição do método de "deadlock prevent". O algoritmo não evita efetivamente o problema do "deadlock" e, portanto, conforme a definição de "deadlock avoidance", não pode ser classificado como tal, pois o método não captura todas as possíveis situações de travamento de maneira que o algoritmo possui um procedimento para recuperar o sistema do travamento.

Em Banaszak e Krogh [1990] é proposto a modelagem do sistema utilizando redes de Petri [Murata, 1989; Reisig, 1985; Reisig, 1992; Peterson, 1981]. Os autores consideram que cada recurso possui dois "buffers" (um de entrada e um saída) e denominam este modelo de "Production Petri Net" (PPN). Esta rede é utilizada para modelar processos concorrentes e alocação dinâmica de recursos para evitar o estado de "deadlock". Na figura 4.15 é apresentado o modelo em PPN do exemplo 2 da tabela 4.3.

Tabela 4.3 Seqüência de utilização dos recursos pelos processos

	Processo	Seqüência de utilização de recursos
1	A	R1, R2, E R3
2	B	R5 e R6
3	C	R2, R4 e R5
4	D	R6, R4 e R1
5	E	R3 e R2

Os autores propõem uma política de alocação de recursos em tempo real de forma que a condição de espera circular nunca ocorra a partir de modelos do sistema em PPN e a seqüência de utilização dos recursos. É considerado que cada recurso possui um buffer de entrada (I_n) e um de saída (O_n). Cada buffer possui dois lugares que representam a disponibilidade ($I_n a$ ou $O_n a$) e utilização ($I_n b$ ou $O_n b$) do recurso ($r \mid r \in R$).

A seqüência de produção p_q é particionada em subseqüências ou zonas z_q, isto é, cada seqüência p_q possui um conjunto de $n(p_q)$ zonas tal que:

$$p_q = p_q(0) z_q^1 z_q^2 \ldots z_q^{n(p_q)} p_q(L_q + 1) \tag{4.6}$$

onde q representa um determinado produto com a respectiva seqüência de lugares:

$$\begin{aligned} &q \in Q \\ &p_q = \{p_q(0), p_q(1), \ldots, p_q(L_q+1)\} \end{aligned} \tag{4.7}$$

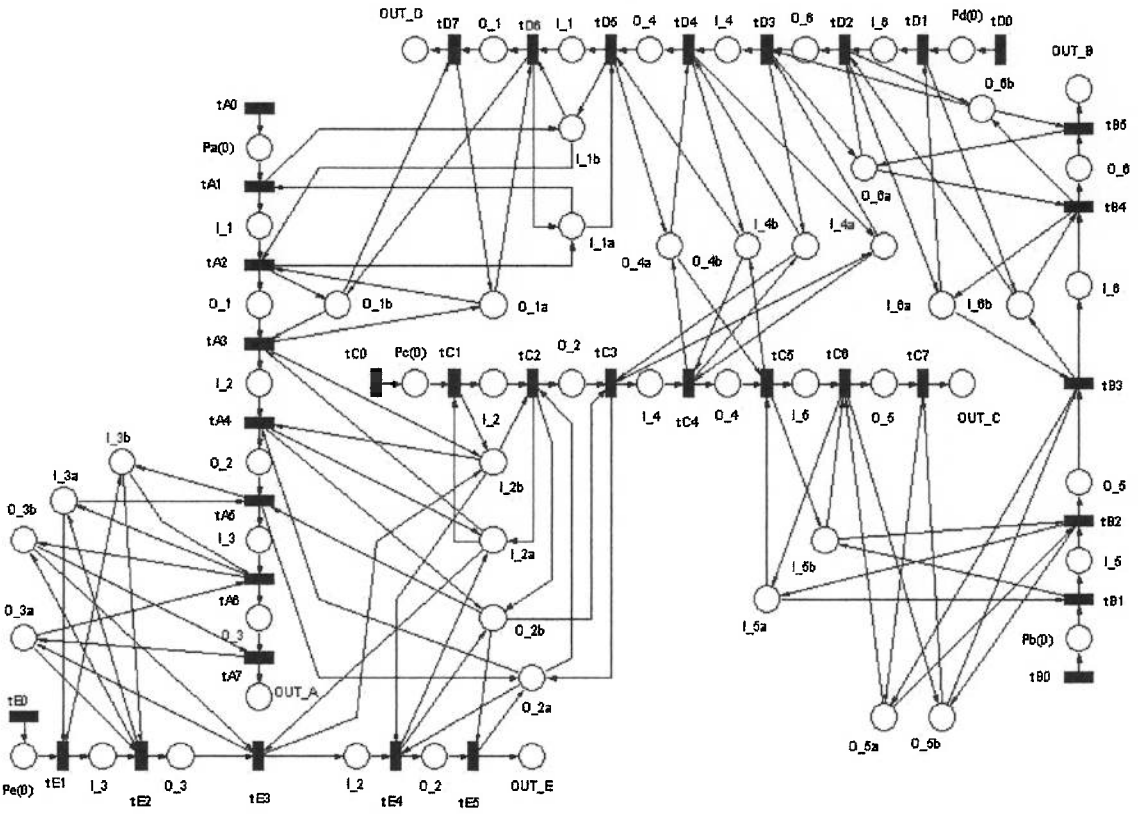


Figura 4.15 Modelo conforme Banaszak e Krough [1990].

O lugar $p_q(0)$ representa a entrada do processo no sistema e o lugar $p_q(L_q+1)$ representa a saída do processo do sistema. Os lugares intermediários representam as etapas da seqüência de produção.

Cada zona é decomposta em sub-zonas formadas por dois conjunto de recursos: compartilháveis s_q e não compartilháveis u_q tais que:

$$\begin{aligned} &z_q^k = s_q^k u_q^k \\ &k = 1,2,\ldots,n(p_q) \end{aligned} \tag{4.8}$$

A quantidade de zonas de um processo ($n(p_q)$) é determinada somando-se 1 à quantidade de recursos não compartilháveis que existem na seqüência de utilização de recursos deste processo, respeitando-se a ordem: compartilháveis e não compartilháveis.

O algoritmo de "deadlock avoidance" (Deadlock Avoidance Algorithm - DAA) atua no sistema através de uma política de restrições que determina se uma transição pode ser disparada quando um recurso está disponível de acordo com duas regras:

- Regra 1. Permite a entrada de um processo na zona somente se a quantidade de recursos não compartilhados da subzona for maior que o número de processos que estão presentes na zona corrente;

- Regra 2. Assegura que se um recurso compartilhado é requisitado por um processo, todos os demais recursos compartilhados pertencentes a zona estarão disponíveis naquele momento.

Basicamente, o objetivo dessas duas regras é evitar que a alocação de um determinado recurso possa fechar o ciclo de espera. A forma como são executadas para alcançar este objetivo baseia-se na habilitação ou inibição das transições. Para sistemas em que todos os recursos são compartilháveis, isto é, não há recursos não compartilháveis, a regra 1 nunca poderá ser atendida.

O método de "deadlock avoidance" proposto por Kumaran et al. [1994] realiza um mapeamento antecipado de todo o sistema cada vez que um processo aloca um recurso e quando se inicia a execução de um novo processo no sistema. Este mapeamento é

realizado utilizando-se grafos [Netto, 2001]. O método verifica a quantidade de recursos que estão sendo executados dentro dos ciclo de espera.

Uma vez sendo possível prever a ocorrência de "deadlock", os processos que podem causar o travamento são transferidos para um buffer (fila de espera). Os autores propõem uma técnica denominada de redução de grafos que visa simplificar o procedimento de "deadlock avoidance". A redução de grafos tem como objetivo prever o estado de "deadlock" nos ciclos de espera e a inter-relação entre os ciclos de espera.

O procedimento inicia-se com a construção do grafo de alocação de recursos conforme apresentado anteriormente na descrição do método de "deadlock detection and resolution" proposto pelos mesmos autores. A partir desse grafo cria-se o grafo de primeira ordem, isto é, os quadrados são transformados em círculos e os arcos recebem uma nova nomenclatura. A indicação de um processo utilizando um recurso é feita por um número (identificação do processo) e por um sinal '[+]'.

Uma vez criado o grafo de primeira ordem, são construídos outros grafos de ordem superior até o seu limite da seguinte maneira:

(i) Cada ciclo de espera torna-se o nó do grafo de ordem superior;

(ii) Os arcos que saem do nó da intersecção de dois ciclos de espera consecutivos tornam-se os arcos do grafo de ordem superior;

(iii) Os nós do grafo de segunda ordem e superior não representam recursos mas sim ciclos de espera de ordem inferior.

Portanto, a condição necessária para a ocorrência de "deadlock" no grafo (G) de uma dada ordem (n) são os ciclos de espera formados com os nós desta ordem. A condição suficiente para a ocorrência de "deadlock", dado um grafo G de ordem n, é que o número de arcos, o número de recursos e o número de indicações de processos ou etapas de processos (sinalizados por '[+]') devem ser iguais. Na figura 4.16 é apresentado o modelo conforme Kumaran baseado no exemplo 2 da tabela 4.3.

O método em questão pode apresentar maior complexidade em termos computacionais. Maiores detalhes podem ser encontrados em Kumaran et al. [1994].

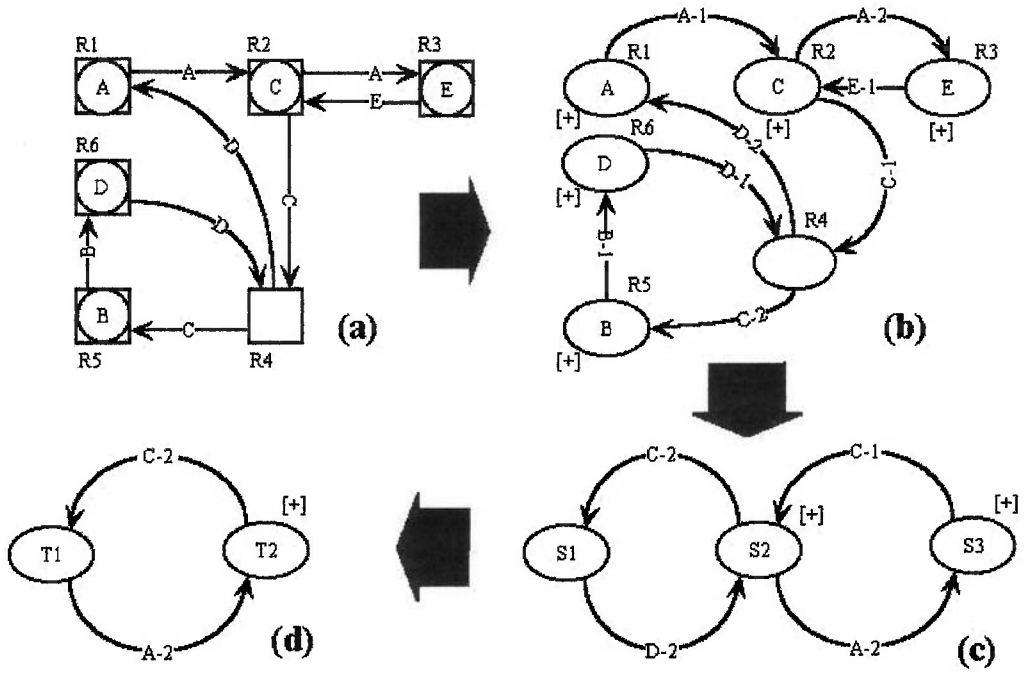


Figura 4.16 Modelo conforme Kumaran (a) e (b) Construção do grafo, (c) Grafo de segunda ordem e (d) Grafo de terceira ordem.

Fanti et al. [1995; 1997] propõem um método de "deadlock avoidance" baseado em um modelo de restrições. Sendo conhecida a seqüência de utilização de recursos de cada

processo, conhecendo-se quais os recursos que estão sendo utilizados e quais serão requisitados é então realizada uma análise desses estados, levando-se em conta a alocação de recursos entre os processos que estão sendo executados e os que entram no sistema.

O método utiliza a teoria de dígrafos [Netto, 2001] que descrevem toda a seqüência de utilização de recursos pelos processos dentro do contexto global ('Working Procedure Digraph') e o estado atual do sistema quanto à utilização de recursos e as futuras requisições de recursos ('Transition Digraph'). Tais dígrafos permitem identificar as condições suficientes e necessárias para a ocorrência de "deadlock" e a identificação de uma situação crítica denominada de Segundo Nível de "Deadlock" ou "Second Level Deadlock" (SLD) e Terceiro Nível de "Deadlock" ou "Third Level Deadlock" (TLD). O SLD e TLD são situações que ocorrem em sistemas que não se encontram em "deadlock" porém, após algumas etapas, ocorrerá o travamento do sistema ("Impending Part Flow Deadlock"), isto é, a inter-relação entre os ciclos de espera em segundo e terceiro nível. O método propõe também um conjunto de cinco regras fundamentais que restringem a entrada de processos dentro dos ciclos de espera [Fanti et al. , 1997].

Em Wu [1999] é proposta uma abordagem modelando o sistema de manufatura flexível que descreve a dinâmica da competição pela utilização de recursos. O modelo em rede de Petri colorida [Cardoso e Valette, 1997] é denominado de "Colored Resource-Oriented Petri Net" (CROPN). Na figura 4.17 é apresentado o modelo conforme Wu [1999] corresponde ao exemplo 2 da tabela 4.3.

A condição necessária e suficiente para a não ocorrência de travamento no sistema é a garantia de vivacidade do modelo [Peterson, 1981; Reisig, 1985; Reisig, 1992; Murata, 1989]. Um ciclo de espera em CROPN é chamado de Circuito do Processo de Produção (PPC – Production Process Circuit). É uma classe especial de circuito que desempenha um papel importante para garantir a vivacidade do CROPN através do controle das marcas dentro do PPC, habilitando-se ou desabilitando-se determinadas transições que evitam o travamento. O objetivo básico é controlar a entrada de processos no ciclo de espera que causem o "deadlock".

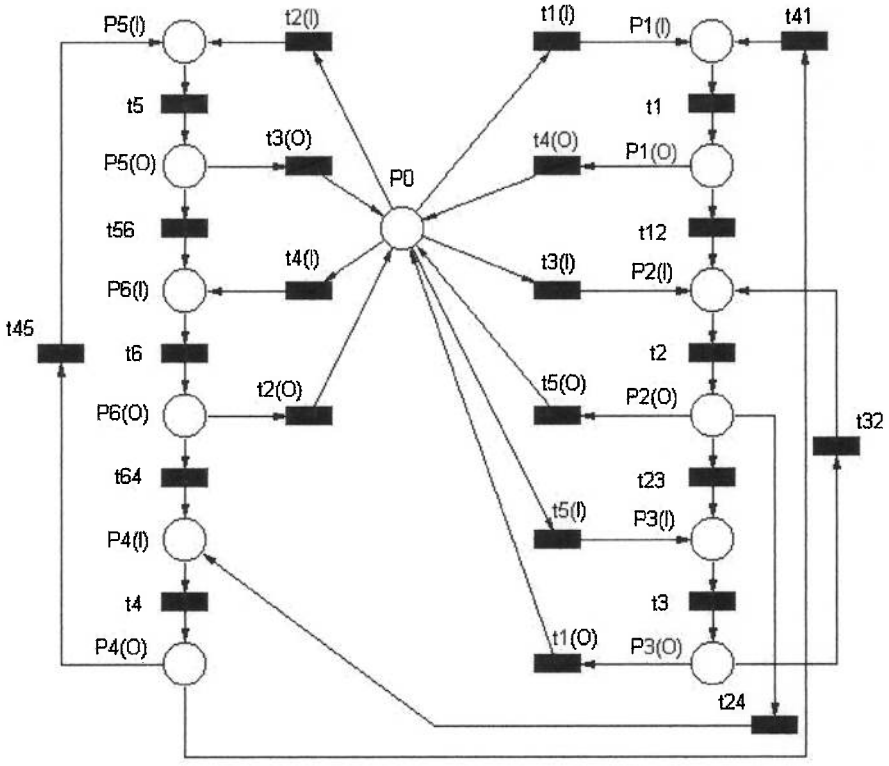


Figura 4.17 Modelo conforme Wu [1999].

Yoon e Lee [2000a, 2000b] propõem um método de "scheduling" livre de "deadlock" através do controle da alocação de recursos. Primeiramente, as informações do seqüenciamento de utilização de recursos pelos processos e o conjunto de tipos de

recursos disponíveis, e suas respectivas quantidades, são armazenados em um banco de dados. Baseado nessas informações, o conjunto de recursos é dividido em dois grupos:

- Conjunto potencial de "deadlock", que é o conjunto de recursos que podem levar o sistema ao travamento no futuro;
- Conjunto livre de "deadlock", que é o conjunto de recursos livre de travamento.

O conjunto potencial de "deadlock" é identificado através da matriz de requisição de recursos, isto é, dado um conjunto de recursos R tal que:

$$R = \{r_1, r_2, \ldots, r_{|R|}\} \tag{4.9}$$

onde $|R|$ representa a cardinalidade do conjunto R.

A matriz S^τ para o processo τ é definido da seguinte forma:

$$S^\tau = \begin{array}{c} \\ r_1 \\ r_2 \\ \vdots \\ r_{|R|} \end{array} \begin{array}{c} \begin{array}{cccc} r_1 & r_2 & \cdots & r_{|R|} \end{array} \\ \begin{bmatrix} s^\tau_{1,1} & s^\tau_{1,2} & \cdots & s^\tau_{1,|R|} \\ s^\tau_{2,1} & s^\tau_{2,2} & \cdots & s^\tau_{2,|R|} \\ \vdots & \vdots & \ddots & \vdots \\ s^\tau_{|R|,1} & s^\tau_{|R|,2} & \cdots & s^\tau_{|R|,|R|} \end{bmatrix} \end{array} \tag{4.10}$$

onde

$$s^\tau_{i,j} = \begin{cases} 1 \Rightarrow Se(r_i \in R) \wedge (r_j \in R) \\ ou \\ 0 \end{cases} \tag{4.11}$$

Então, se o sistema possui T processos, a matriz de requisição de recursos S é definida como sendo:

$$S = \sum_{\tau=1}^{T} S^{\tau} \tag{4.12}$$

A partir da matriz S identifica-se os elementos que compõem o indicador do conjunto potencial de "deadlock" L, isto é:

$$L = \{s_{i,j} \mid (s_{i,j} > 0) \wedge (i > j)\} \tag{4.13}$$

para

$$\begin{aligned} i &= \{1,2,...,|R|\} \\ j &= \{1,2,...,|R|\} \end{aligned} \tag{4.14}$$

Para cada elemento do conjunto L, isto é,

$$s_{i,j} \in L \tag{4.15}$$

São calculados os ciclos de espera (condição de espera circular) que formam um conjunto potencial de "deadlock" D, onde:

$$\begin{aligned} &r \in D(i,j) \\ &D(i,j) = \{r_k \mid (r_k \in R) \wedge (j \le k \le i)\} \\ &D = \bigcup D(i,j) para\_todo(i,j) \in \{(i,j) \mid s_{i,j} \in L\} \end{aligned} \tag{4.16}$$

As condições de espera circular são calculados utilizando-se o algoritmo baseado no algoritmo de busca em profundidade na versão recursiva [Netto, 2001]. Os ciclos de espera são então armazenados no banco de dados.

A proposta dos autores é realizar um "scheduling" livre de "deadlock", isto é, através das informações contidas em banco de dados gera-se dinamicamente um "scheduling" utilizando-se gráficos de Gantt [Rocha, 1995] de forma on-line. A figura 4.18 apresenta o exemplo de uma célula de manufatura executando três processos (*J1*, *J2* e *J3*) em três recursos (*M4*, *M5* e *M6*) em um determinado instante de tempo. O processo *J2* está utilizando o recurso *M4* e irá requisitar o recurso *M5* na seqüência. O processo *J1* está utilizando o recurso *M5* e ao finalizar a etapa irá requisitar o recurso *M4*. Neste momento ocorreu a entrada do processo *J3* no sistema que está requisitando o recurso *M5* e na seqüência irá requisitar o recurso *M4*, formando um ciclo de espera com o processo *J2*.

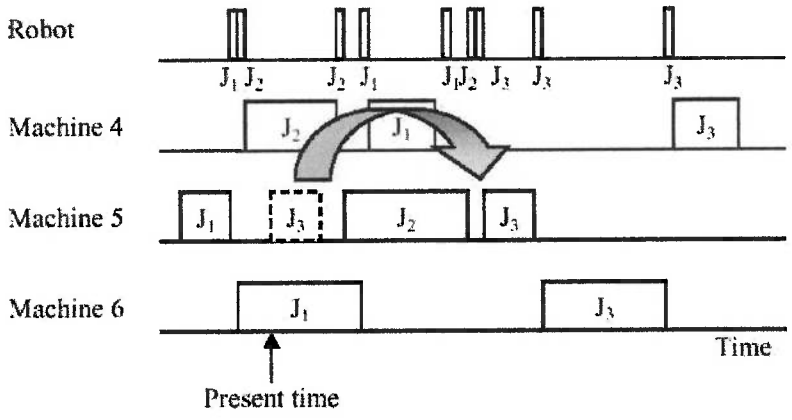


Figura 4.18 Exemplo de geração do gráfico de Gantt de forma on-line.

A ação para evitar o estado de "deadlock" é baseada nas informações armazenadas no banco de dados. Basicamente são criados todos possíveis seqüenciamentos de utilização de recursos pelos processos evitando que se formem os ciclos de espera. Após computar todas as seqüências possíveis, é então fornecida a melhor alternativa.

O método proposto por Yoon e Lee [2000a, 2000b] considera a disponibilidade e a indisponibilidade de recursos, isto é, uma vez que um determinado recurso torna-se indisponível (quebra de máquina, por exemplo) são calculadas rotas alternativas. A desvantagem deste método é que torna-se ineficiente se o conjunto livre de "deadlock" for consideravelmente pequeno, o que incrementa a complexidade computacional do sistema conforme apresentada pelos autores.

Santos Filho [2000a] propõem um modelo de restrições em uma estrutura hierárquica de controle [Santos Filho, 2000a], isto é, divide-se o controle em dois níveis hierárquicos (superior e inferior), aplicando-se o conceito de "deadlock avoidance" no nível superior. Os SPFs são sistemas classificados como sistemas complexos pois possuem dois níveis de indeterminismo (em relação ao tempo e em relação à seqüência de eventos). A complexidade surge quando é imposto um comportamento pré-estabelecido no sistema. Uma vez que, devido à natureza complexa dos SPFs, não é possível determinar todos os estados alcançáveis do sistema, identifica-se e evita-se os estados indesejáveis. Neste contexto, entende-se por estados indesejáveis os estados de "deadlock" em que há um travamento do sistema devido à interrupção permanente do fluxo de itens inerente aos processos envolvendo compartilhamento de recursos. Basicamente o método identifica os estados anteriores ao travamento e utiliza esses dados gerando um modelo de restrições para mapear as ações necessárias para impedir que o sistema evolua para o travamento de forma dinâmica.

Essas restrições são denominadas regras adicionais de controle[2] [Santos Filho, 2001; 2000a; 1998]. Essas regras são aplicadas no controle supervisório dos SPFs, ou seja, são aplicadas no controle de utilização dos recursos que permite a alocação de recursos no controle do processo.

4.5 Observações Complementares

Apresentado uma revisão dos métodos fundamentais que abordam o "deadlock" em SPFs, faz-se necessário descrever uma síntese dos aspectos conclusivos deste capítulo no sentido de destacar a adequabilidade de cada técnica em relação à proposta deste trabalho em si. Portanto, verificou-se que:

- método de ignorar a ocorrência de "deadlock" em SPFs é inadequado pois causará uma paralisação do sistema e consequentemente um prejuízo financeiro, além de não atingir os objetivos dos SPFs.

- Por sua vez o método de "deadlock prevent" impõe restrições estruturais estáticas ao sistema que impedem a ocorrência de "deadlock" mas reduz consideravelmente a flexibilidade do sistema, principalmente se for realçado o fato de que em SPFs deve-se levar em consideração como diferentes processos podem estar se combinando para criar um processo global distinto a cada instante. Logo, dependendo do número de elementos (recursos, processos, etc.) do sistema, essa

[2] É importante observar que as regras adicionais de controle podem agregar diversas funcionalidades além de evitar o "deadlock", como por exemplo otimização da produção, prioridades entre processos, etc...

combinação pode ser exponencial, isto é, o grau de complexidade de implementação do método de "deadlock prevent" é proporcional à dimensão[3] do sistema. Entretanto, para sistemas de pequeno porte, envolvendo poucos elementos, o método de "deadlock prevent" é o mais indicado.

- O método de "Deadlock Detection and Resolution" possui algumas desvantagens. Os processos paralisados e/ou cancelados podem perder parte ou todas as atividades que estavam sendo realizadas durante a ocorrência do travamento, além do fato de que o tempo de restabelecimento pode ser relativamente longo. Em uma análise custo-benefício pode ser inviável o sistema retornar ao estado anterior ao "deadlock". Porém, da mesma forma que o método de "Deadlock Prevent", é possível obter resultados satisfatórios para sistemas de pequeno porte que envolvem poucos elementos.

- Os métodos de "deadlock avoidance" apresentados propõem evitar o "deadlock" controlando-se a quantidade de processos que entram nos ciclos de espera. Em alguns trabalhos [Kumaran et al., 1994; Fanti et al., 1995; Fanti et al., 1997] são abordados o caso de "Impending Part Flow Deadlock", isto é, a inter-relação entre ciclos de espera consecutivos. Uma outra característica comum é que todos possuem uma complexidade computacional proporcional ao tamanho do sistema [Fanti et al., 1997] em questão, podendo inclusive inviabilizar a sua implementação

[3] Entenda-se por dimensão como sendo o número de processos , recursos e demais dispositivos que compõem o sistema.

devido ao grande esforço computacional. O esforço computacional implica em distinguir os custos dos algoritmos off-line e on-line [Fanti et al., 1997]. Os algoritmos off-line são aplicados antes da execução dos algoritmos on-line. As aplicações dos métodos de "deadlock avoidance" necessitam de um controle on-line do sistema, isto é, a partir do modelo do sistema são implementados algoritmos de controle que monitoram o sistema em tempo real. Portanto, a viabilidade depende da possibilidade de implementação de algoritmos on-line e off-line.

- Por fim, outro aspecto relaciona-se ao controle de alocação de recursos. Nos trabalhos de Banaszak e Krogh [1990], Viswanadham et al. [1990], e Wu [1999] o controle dos processos e o controle dos recursos são tratados em um mesmo nível, enquanto que em Yoon e Lee [2000a, 2000b], Kumaran et al.[1994], Fanti et al.[1995; 1997] e Santos Filho [2000a] as estratégias são aplicadas no Sistema de Alocação de Recursos.

CAPÍTULO 5
ARQUITETURA HIERÁRQUICA DE CONTROLE

5.1 INTRODUÇÃO

Os SPFs são sistemas pertencentes à classe de SEDs (Sistemas a Eventos Discretos) e, portanto, são caracterizados pela existência de paralelismo, conflito e assincronismo de eventos. Desta forma, os SPFs possuem um primeiro nível de indeterminismo: o indeterminismo em relação ao tempo. Há ainda um possível segundo nível de indeterminismo que está relacionado à seqüência de eventos devido à execução de atividades simultâneas (processos) e, conforme discutido anteriormente, este pode ser um fato que caracteriza um sistema complexo.

Um outro aspecto a ser considerado é que a execução de atividades simultâneas com intenso compartilhamento de um mesmo conjunto de materiais, recursos e/ou informações pode levar o sistema a um auto-travamento ou "deadlock", isto é, quando o fluxo dos processos são permanentemente impedidos.

Para abordar o problema de "deadlock" em SPFs é adotado neste trabalho o método proposto por Santos Filho [2000a]. Este método propõe uma solução para o controle de SPFs, considerando-se uma arquitetura hierárquica para o sistema de controle. Baseado nesta arquitetura, é aplicada o conceito de "deadlock avoidance" que evita o problema de "deadlock" em SPFs.

O objetivo deste capítulo é apresentar a arquitetura hierárquica de controle, as ferramentas de modelagem, a descrição do método adotado e a aplicação do método utilizando um exemplo.

5.2 HIERARQUIA DO SISTEMA

Nos SPFs a combinação de estados alcançáveis pode ser exponencial, dificultando o controle sobre o mesmo. Em alguns trabalhos referentes ao "deadlock" verifica-se uma relativa dificuldade na modelagem do controle de SPFs considerando o controle de processos e a alocação de recursos simultaneamente no mesmo nível de abstração [Banaszak e Krogh, 1990; Viswanadham et al., 1990; Kumaran et al., 1994; Wu, 1999].

A solução adotado neste trabalho contempla dois aspectos fundamentais. Em primeiro lugar é baseado em modelos de restrições [Santos Filho, 2000a]. A proposta baseia-se no fato de que em alguns casos, não é possível determinar todos os estados alcançáveis do sistema porém, através de regras adicionais de controle é possível evitar que o sistema atinja determinados estados indesejáveis. O outro aspecto refere-se à divisão do sistema de controle em dois níveis hierárquicos [Santos Filho, 2000a] (figura 5.1):

- Controle de Processos: O nível inferior, responsável por garantir o seqüenciamento das etapas dos processos, alocando recursos disponíveis;

- Controle de Recursos: O nível superior, responsável por gerenciar a utilização dos recursos pelos processos. De fato, é um controle supervisório.

O termo controle supervisório é usualmente associado aos processos industriais, mas o conceito pode ser aplicado igualmente para automação da manufatura. Cada processo possui um controlador e o controle supervisório atua sobre esses controladores. Em processos industriais, o controle supervisório gerencia as atividades da integração de unidades de operações para alcançar determinados objetivo. Em algumas aplicações,

não é mais do que um controle regulador ou controle de alimentação avante. Em outras aplicações, o sistema de controle supervisório é designado para a busca da otimização de uma função objetivo, normalmente baseado em medidas de taxas de desempenho como rendimentos, taxas de produção, custos, qualidade ou outros critérios que estão relacionados à performance do processo.

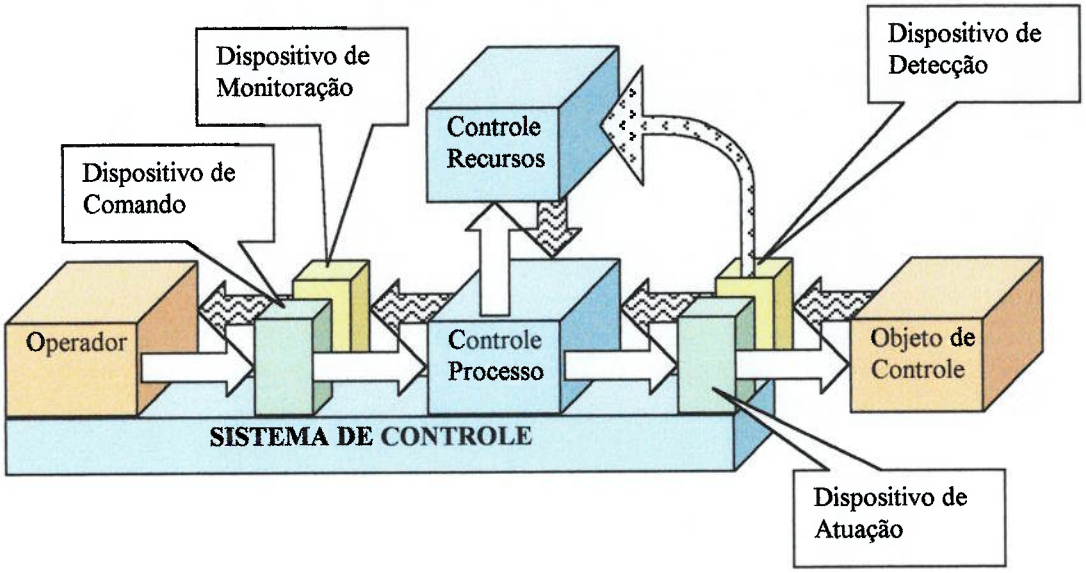


Figura 5.1 Estrutura hierárquica do sistema de controle.

No contexto de automação da manufatura, o controle supervisório pode ser definido como um sistema de controle que direciona e coordena as diversas atividades e demais dispositivos de controle. Os objetivos deste tipo de controle podem incluir: definições das melhores condições operacionais, minimização do custo da produção, ou ainda maximização da utilização de recursos, por exemplo, através de propostas de "scheduling" mais eficientes.

Desta forma, o controle de recursos irá monitorar as etapas dos processos quanto à utilização de recursos de acordo com um conjunto de regras adicionais de controle. Os dois níveis hierárquicos possuem semânticas distintas, isto é, quando um processo

requisita um determinado recurso no nível inferior, esta solicitação é feita ao controle de recursos (nível superior). Ao ser autorizado pelo controle de recursos, tem-se no nível superior uma indicação que o recurso está sendo utilizado e consequentemente que pode ser alocado no nível inferior. Desta forma, a utilização do recurso no nível superior determina a alocação do mesmo no nível inferior.

Portanto, o controle de um SPF pode ser realizado em um modelo hierárquico de controle através de um modelo de restrições em que aplicam-se regras adicionais de controle no nível superior (controle de recursos) que estabelece uma lógica de alocação de recursos no nível inferior (controle de processos).

A modelagem do controle de recursos, conforme proposto por Santos Filho [2000a], é realizada utilizando-se as ferramentas de modelagem gráficas e matemáticas: (i) Grafo de Alocação de Recursos (GAR) e (ii) Enhanced Mark Flow Graph (E-MFG).

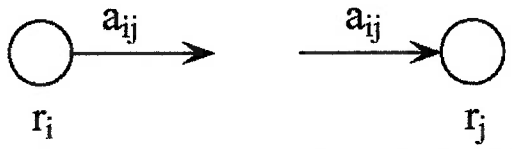
5.3 Ferramentas de Modelagem

5.3.1. Grafo de Alocação de Recursos (GAR)

Um Grafo de Alocação de Recursos (GAR) [Santos Filho 2000a; Santos Filho et al. 2000b; 2001] é um grafo bipartido, não marcado e que representa a utilização de recursos pelos processos. O GAR é utilizado para determinar a condição de espera circular que, neste trabalho, corresponde aos Ciclos Fechados de Espera (CFE). O grafo GAR $G=(R, A)$ é um conjunto não vazio tal que R é um conjunto de nós, que representam os recursos $R = \{r_1, r_2, r_3, ..., r_n\}$, e A o conjunto de arestas ou arcos

orientados, que representam a alocação e requisição de recursos. Um arco orientado é um par ordenado $a_{ij} = (r_i, r_j)$, onde r_i e r_j são elementos de R.

Um par contendo um nó e o respectivo arco orientado de saída é **denominado par de alocação** enquanto que um par contendo um nó e o respectivo arco de entrada é denominado **par de requisição**. Em um **par de alocação**, o nó representa um determinado recurso r_i e o arco orientado de saída possui uma inscrição fixa que representa a etapa do processo a_{ij} que aloca r_i para ser executada. Por sua vez em **um par de requisição** o nó também representa um determinado recurso r_j e o arco orientado de entrada possui uma inscrição fixa que representa a etapa do processo a_{ij} que o requisita para ser utilizado na próxima etapa do processo. A figura 5.2 ilustra graficamente a representação destes pares para o caso de um mesmo produto em uma determinada etapa de sua seqüência de produção.



(a) Par de Alocação (b) Par de Requisição

Figura 5.2 Elementos fundamentais de um GAR.

A representação de um GAR é feita através de sua Matriz de Adjacência [Netto, 2001]. Seja G um GAR de ordem n, isto é, n recursos: R. A matriz de adjacência é uma matriz $n \times n$, onde o valor de cada elemento $e_{ij} = a_{ij} = (r_i, r_j)$ da matriz é determinado da seguinte maneira: Se $a_i \neq \varnothing$ e $a_j \neq \varnothing$ então $e_{ij} = a_{ij}$ do contrário $e_{ij} = 0$;

Por exemplo, um processo q que possui a seguinte seqüência de utilização de recursos: r_1, r_3 e r_2. Admitindo-se uma entrada e uma saída para o processo (q_{in} e q_{out}) a sua respectiva representação através da Matriz de Adjacência é:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ q_{IN} \\ q_{OUT} \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & q_{IN} & q_{OUT} \end{array} \\ \begin{pmatrix} 0 & 0 & q_{1,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{2,OUT} \\ 0 & q_{3,2} & 0 & 0 & 0 \\ q_{IN,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{5.1}$$

5.3.2. Ciclo Fechado de Espera (CFE)

A condição de espera circular pode ocorrer em SPFs onde há compartilhamento de recursos. Essa condição de espera circular no modelo GAR é denominada Ciclo Fechado de Espera (CFE).

A figura 5.3 apresenta os elementos que compõem um CFE em um modelo GAR. O nó que corresponde à interseção é denominado nó de interseção. O nó de interseção é formado pelo cruzamento de um ou mais segmentos[1]. Cada seguimento possui:

- Um Par de Alocação de Interseção (PAI);
- Um Par de Alocação Terminal (PAT).

[1] Conexão contínua de um determinado par de alocação de Interseção até o próximo par de alocação de intersecção.

O PAT de um seguimento aponta o nó de intersecção que, por sua vez, é o PAI do próximo seguimento.

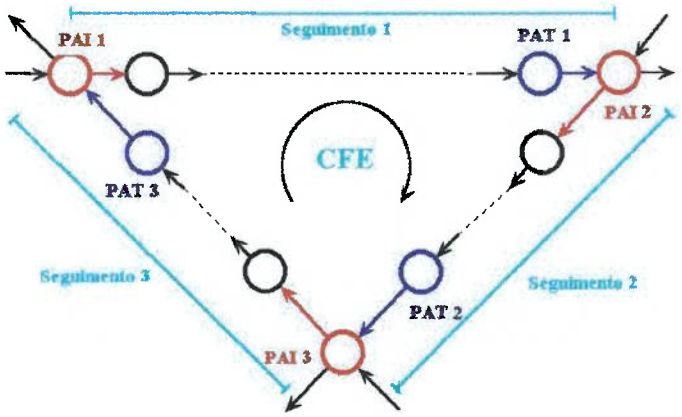


Figura 5.3 Elementos de um CFE.

Existem dois tipos de CFE [Santos Filho, 1998; 2000a; 2001]:

a) Quando as rotas de processos diferentes cruzam-se antes de atingirem as etapas finais de cada seqüência de produção, isto é, forma-se pela intersecção de várias rotas diferentes (figura 5.4 a).

b) Quando a rota de um determinado processo cruza-se sobre si mesma antes de atingir a etapa final da seqüência de produção, isto é, forma-se pela intersecção de uma única rota sobre si mesma (figura 5.4 b);

Além dos dois tipos citados anteriormente, os CFEs podem ser ainda classificados em mais dois tipos:

- CFE Simples: Os CFEs simples são formados por um único ciclo (figura 5.5a);

- CFE Compostos: Os CFEs compostos são formados por uma seqüência de dois ou mais ciclos de espera sendo que na intersecção entre os CFEs há um recurso em comum além de nenhuma saída ou entrada de processos que pertencem aos demais ciclos (figura 5.5b).

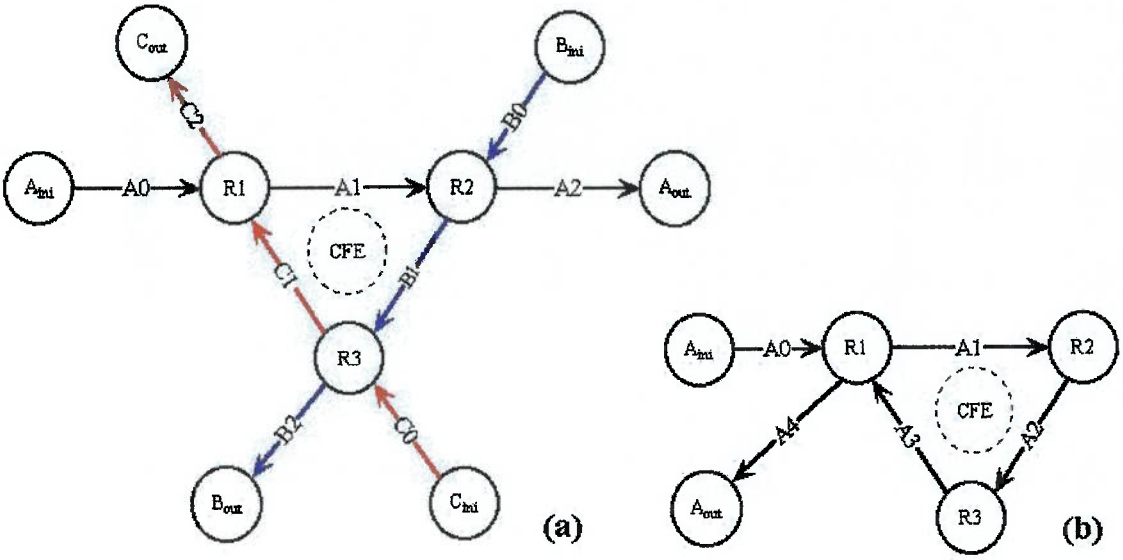


Figura 5.4 Tipos de CFE.

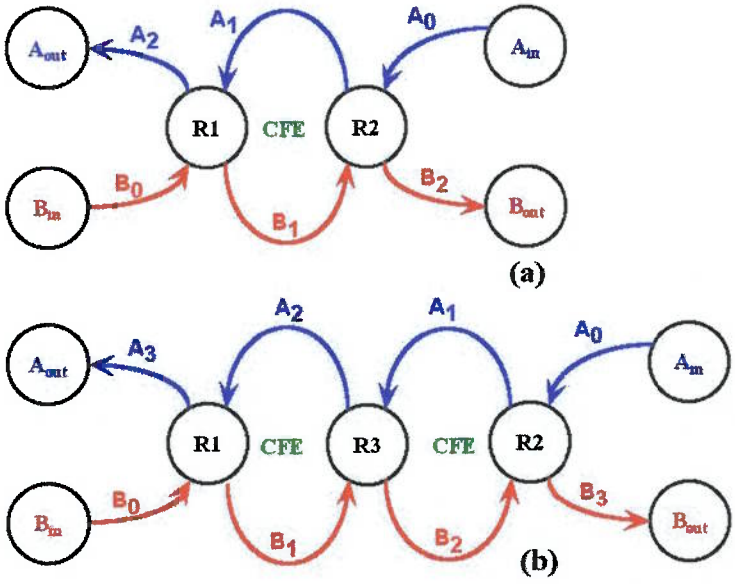


Figura 5.5 Outros tipos de CFE (a) Simples (b) Compostos.

5.3.3. Mark Flow Graph (MFG)

O MFG (Mark Flow Graph) é um grafo bipartido que representa as características fundamentais necessárias para a representação estruturada de sistemas seqüenciais complexos considerando o comportamento funcional do sistema e a realização do seu controle [Miyagi, 1996; Santos Filho, 1993; Santos Filho 1998]. É uma subclasse de redes deduzida a partir de Redes de Petri [Peterson, 1981; Reisig, 1985; Murata, 1989].

5.3.3.1. Elementos Estruturais do Grafo MFG

O MFG é composto pelos seguintes elementos estruturais (figura 5.6):

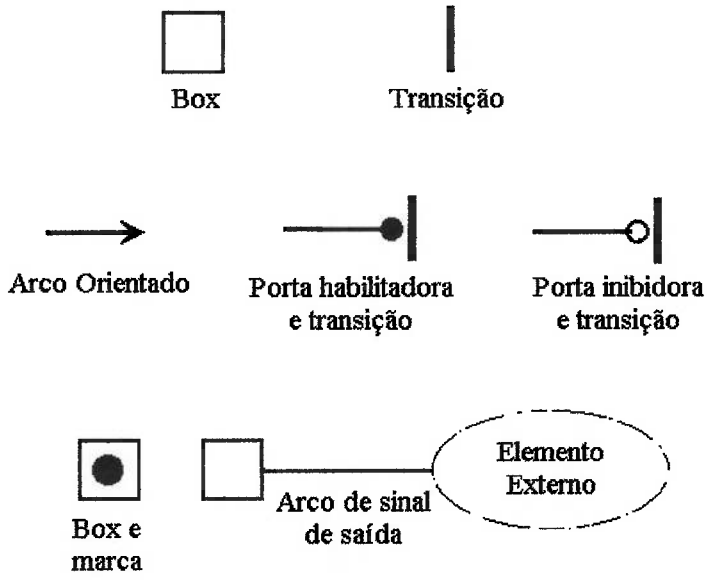


Figura 5.6 Elementos estruturais básicos do MFG.

a) "Box". O "box" corresponde a uma condição que pode estar associada ao modo de operação ou à disponibilidade de um elemento do sistema. É representada por um bloco quadrado no grafo;

b) Transição. A transição corresponde a um evento que causa uma mudança de estado do sistema. É representada por uma barra vertical no grafo;

c) Arco orientado. A relação entre um "box" e uma transição é representada por um arco orientado que conecta estes dois elementos. Os arcos são representados por setas;

d) Marca. A manutenção de uma condição é representada pela existência de uma marca no "box" correspondente. A alocação de marcas nos "boxes" do grafo constitui a sua marcação;

e) Porta ("Gate"). São elementos que desempenham a função de habilitar ou inibir a ocorrência dos eventos correspondentes às transições a que estão conectadas. A porta habilitadora é representada por um arco com um círculo negro na extremidade que a conecta à transição. Se o sinal de origem desta porta for verdadeiro, a transição a ela conectada é habilitada. A porta inibidora é representada por um arco com um círculo branco na extremidade que a conecta à transição. Se o sinal de origem desta porta for verdadeiro, a transição a ela conectada é inibida. As portas podem ainda ser classificadas em externas e internas. Uma porta interna possui como sinal de origem um "box" do próprio grafo (o sinal de origem faz parte do

modelo). Uma porta externa representa um dispositivo que não faz parte do modelo, isto é, o sinal de origem associado a esta porta é gerado por um dispositivo externo;

f) Arco de sinal de saída. Corresponde a um arco que se origina de um "box" e envia a informação deste "box" para um elemento externo. Se houver marca no "box", envia verdadeiro como sinal, caso contrário, envia falso;

g) Elemento externo. Corresponde a um elemento que não faz parte do modelo.

O MFG possui uma poderosa capacidade de representação dos vários níveis de abstração do modelo, porém, à medida que se eleva o grau de complexidade do sistema, a representação torna-se uma tarefa trabalhosa devido à necessidade de considerar os diversos detalhes pertinentes ao nível de chão de fábrica. Com o objetivo de simplificar a representação do modelo foi adicionado os seguintes elementos ao MFG original criando-se o F-MFG (MFG Funcional) [Santos Filho, 1993]:

a) Elementos temporizados. São elementos para representar o tempo de duração de processos (figura 5.7a). O MFG possui dois tipos: o "box" temporizado e a transição temporizada. O "box" temporizado retém a marca durante um intervalo de tempo pré-definido. Enquanto que a transição temporizada atrasa o disparo de um intervalo de tempo pré-definido;

b) "Box" capacidade. Representa um dispositivo armazenador do tipo FIFO ("First-in First-out") com capacidade de armazenar no máximo **N** marcas (figura 5.7b);

c) "Box" agrupador. Representa uma estrutura formado por um número específico de "boxes" que conectados de forma serial permite o disparo da transição de saída somente após a chegada de todas as *n* marcas (figura 5.7c);

d) "Box" dispersor. Representa a operação inversa de um "box" agrupador (figura 5.7d).

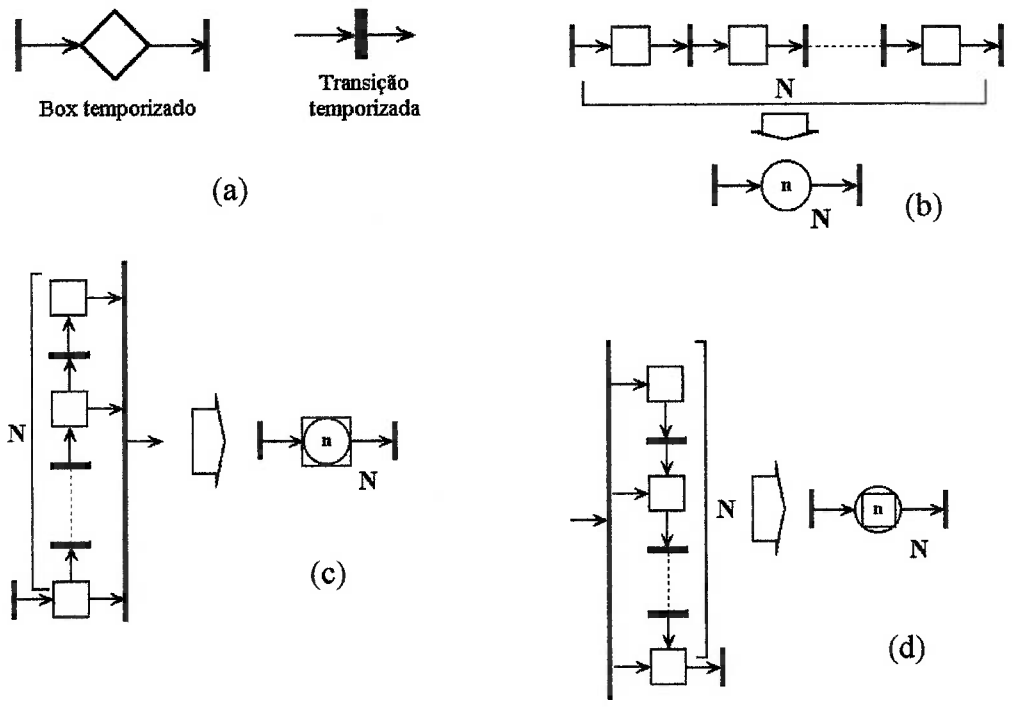


Figura 5.7 Elementos do grafo MFG.

5.3.3.2. Dinâmica das Marcas no Grafo MFG

O comportamento dinâmico do sistema é representado pela evolução das marcas no grafo devido à ocorrência de eventos. A evolução das marcas é regida pelas regras de disparo das transições correspondente à ocorrência de eventos.

Uma transição é denominada habilitada se satisfazer as seguintes condições (figura 5.8):

a) Todos os "boxes" de entrada desta transição encontram-se marcados;

b) Todos os "boxes" de saída desta transição não estão marcados;

c) Todas as portas habilitadoras internas apresentam sinal de origem igual a verdadeiro;

d) Todas as portas inibidoras internas apresentam sinal de origem igual a falso.

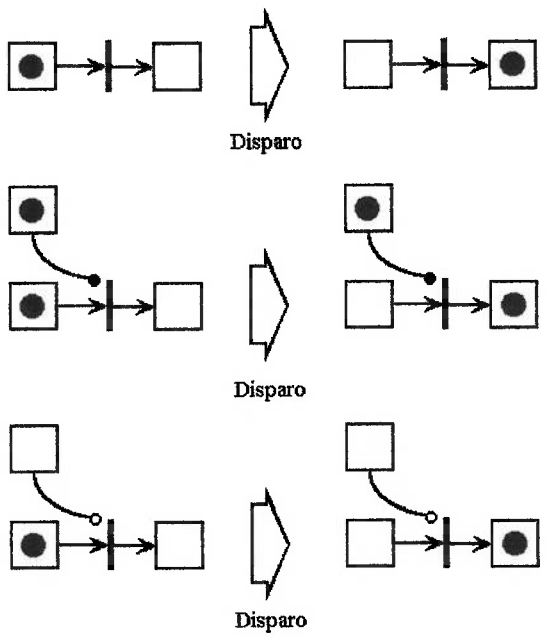


Figura 5.8 Disparo das transições.

Caso uma dessas condições seja falsa, a transição é denominada desabilitada. Para que ocorra o disparo desta transição ainda é necessário que obedeça a outras duas condições:

a) Todas as portas habilitadoras externas no estado habilitado;

b) Todas as portas inibidoras externas no estado desabilitado.

Satisfazendo as duas condições, a transição é denominada transição disparável. O disparo ocorre de forma imediata e instantânea, isto significa que o disparo ocorre em um intervalo de tempo infinitamente pequeno. A figura 5.9 apresenta um outro exemplo da dinâmica do MFG.

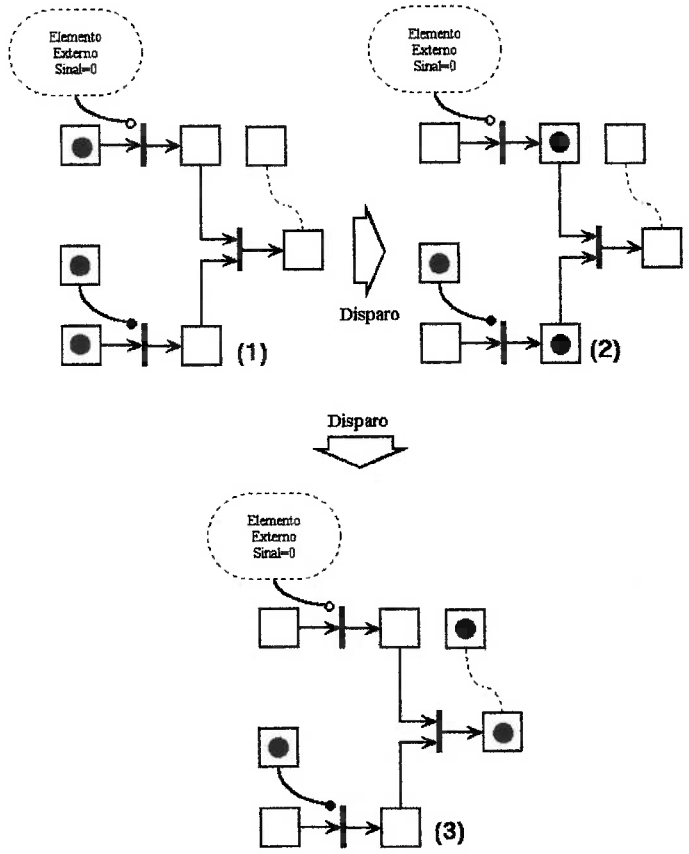


Figura 5.9 Disparo das transições.

5.3.4. Enhanced Mark Flow Graph (E-MFG)

A grande característica do MFG é a facilidade de interpretação do sistema a partir do modelo, porém essa capacidade torna-se limitada quando o MFG é utilizada para modelar sistemas de manufatura de maior porte que exigem a representação de estratégias de controle mais elaboradas [Santos Filho, 1993].

Para superar esta limitação, Santos Filho [1993; 1998; 2000a] propôs uma extensão ao MFG no sentido de aperfeiçoar o poder de modelagem da ferramenta. Neste sentido, criou-se o MFG Estendido ou E-MFG. O E-MFG possui os mesmos elementos e as mesmas regras de disparo do MFG padrão com algumas extensões.

5.3.4.1. Elementos Estruturais do Grafo E-MFG

Os grandes diferenciais em relação ao MFG são: a individualidade e composição das marcas e as regras adicionais de controle para o disparo das transições. Esta técnica tem como objetivo agregar as vantagens disponíveis de uma ferramenta de alto nível, possibilitando uma descrição mais adequada e consistente das características dinâmicas específicas dos SPs dirigidos por eventos.

A seguir são apresentados as extensões nos elementos do E-MFG:

a) Marca. As marcas no E-MFG possuem um conjunto de atributos que lhes garantem individualidade (figura 5.10). Estes atributos são associados às diversas informações do produto, processo e controle. Os atributos são

definidos como um vetor de dimensão *n* e cada posição do vetor representa um determinado atributo. A composição de marcas define uma marca individual composta a partir de marcas individuais simples, incorporando vetores de atributos. Desta forma, é possível representar um agrupamento ou desempacotamento de informações de um determinado processo produtivo. A figura 5.10 apresenta um por exemplo de marcas individuais e a composição de marcas.

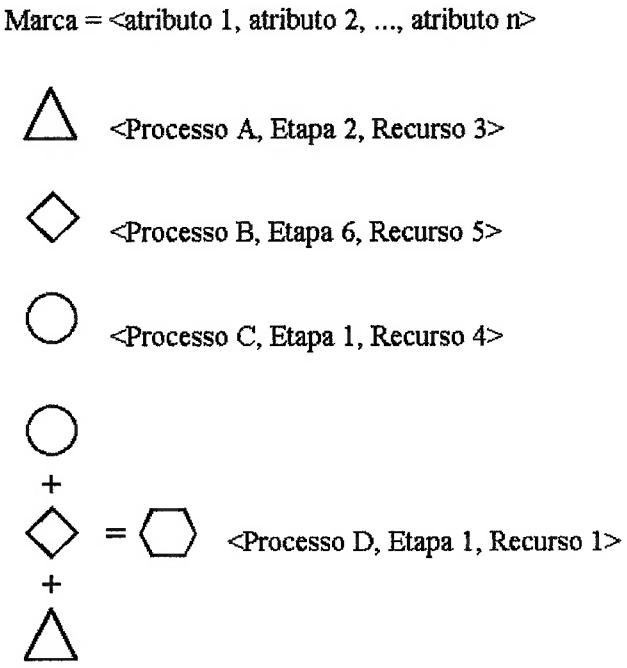


Figura 5.10 Marcas individuais.

b) "Box" capacidade. É adicionado ao "box" especificar, além do número máximo de marcas, a regra para retirada de marcas armazenadas neste "box" (figura 5.11). Uma vez que as marcas são individuais permite

selecionar a forma de retirada das marcas: FIFO ("First-in First-out") ou LIFO ("Last-in First-out");

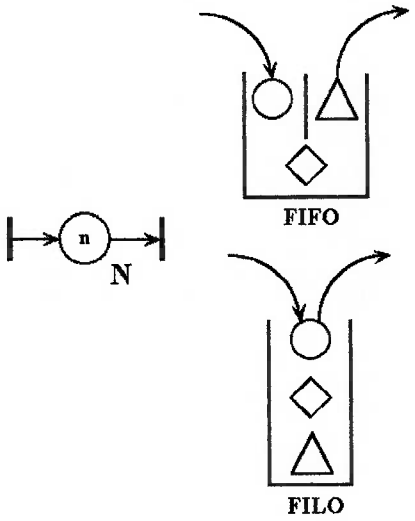


Figura 5.11 "Box" capacidade FIFO e LIFO.

c) "Box" agrupador. Além de preservar as informações das marcas que estão sendo agrupados, quando ocorre o disparo da transição, todas as marcas contidas no "box" desaparecem formando uma marca composta das demais (figura 5.12). Um "box" agrupador representa uma composição de marcas;

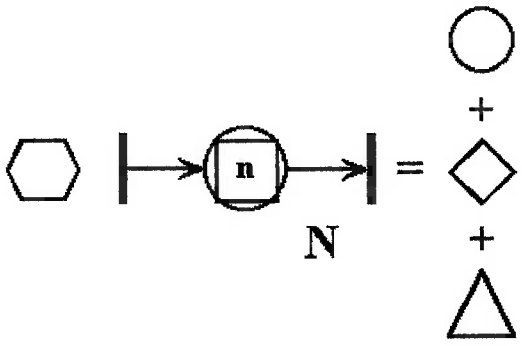


Figura 5.12 "Box" agrupador.

d) "Box" dispersor. Recebe uma marca individual composta na entrada e, na saída, obtêm-se as marcas ordenadamente, uma por uma, mantendo as especificações de seus atributos correspondentes ao estado anterior à composição (figura 5.13);

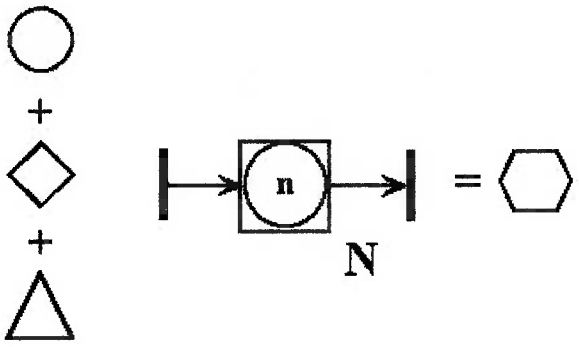


Figura 5.13 "Box" dispersor.

e) "Box" controlador. É um "box" que associa um conjunto de regras de controle para atualizar os atributos das marcas (figura 5.14). Estas regras são regras de produção do tipo "se... então..." que são definidas durante o processo de modelagem e constituem uma forma de especificar estratégias de controle para controle das informações que acompanham as marcas;

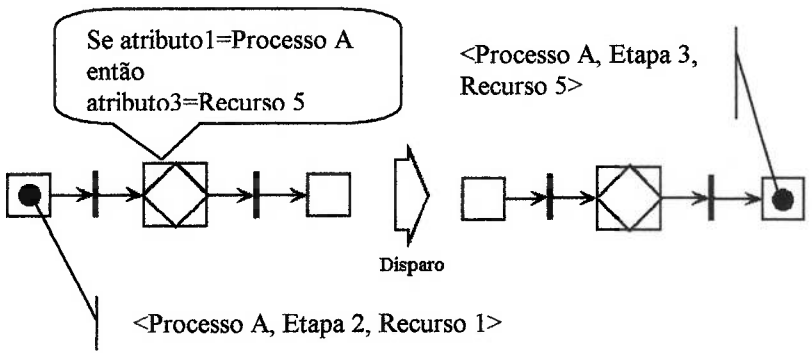


Figura 5.14 "Box" controlador.

f) "Box" temporizador. É um "box" de capacidade unitária que retém a marca em seu interior durante um intervalo de tempo. O intervalo de tempo pode ser definido previamente ou através de um atributo da marca.

5.3.4.2. Dinâmica das Marcas no Grafo E-MFG

A dinâmica de disparo de uma transição é estabelecida por regras de decisão segundo uma determinada hierarquia. Esta hierarquia corresponde a três níveis de decisão:

- Primeiro Nível. Regras de restrições adicionais de disparo, isto é, corresponde às regras de restrições adicionais de disparo que são funções lógicas agregadas às transições;

- Segundo Nível. Regras de habilitação de disparo, isto é, corresponde em satisfazer as mesmas condições apresentadas no MFG;

- Terceiro Nível. Regras de realização do disparo, isto é, correspondem à verificação das regras de arbitragem em situações que envolvem conflito e a verificação das regras de filtragem seletiva dos atributos de acordo com as inscrições nos arcos orientados.

Uma transição habilitada é denominada transição disparável quando possui as possíveis situações de conflito arbitradas e as condições de filtragem seletiva averiguadas. É importante observar que a única forma do disparo de uma transição alterar os atributos

de uma marca é através de filtragem seletiva (figura 5.15). Os atributos não transmitidos são considerados ausentes.

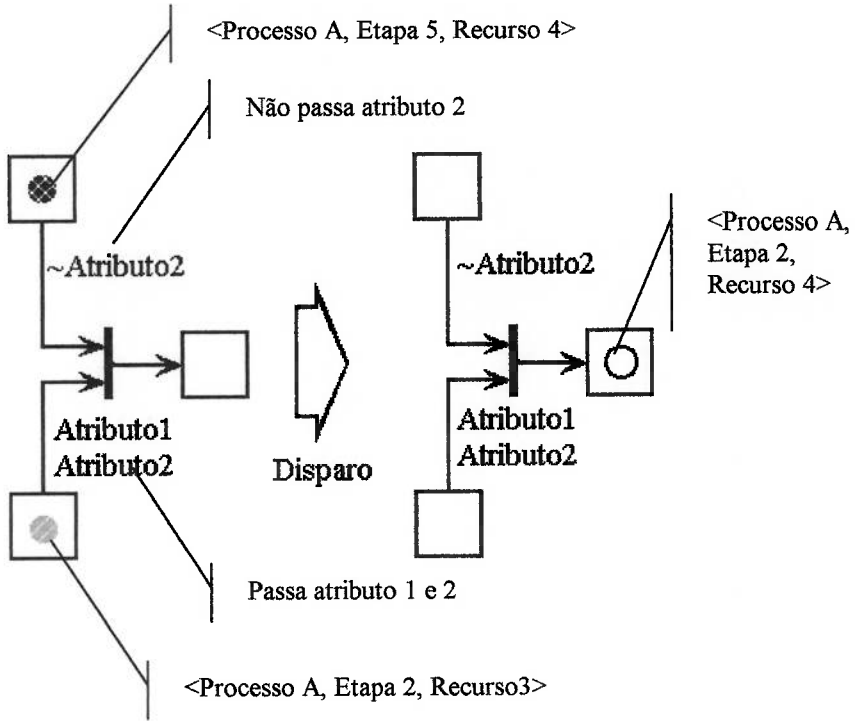


Figura 5.15 Filtragem seletiva.

5.3.4.3. Matriz que Representa o Grafo E-MFG

Para um grafo E-MFG com n etapas (transições) e m recursos ('boxes'), a matriz de incidência $E=[a_{ij}]$ é uma matriz n x m, de inteiros, em que a entrada é dada por:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-} \tag{5.2}$$

Onde $a_{ij}^{+} = w(i,j)$ é o peso do arco da transição i para o box de saída j e $a_{ij}^{-} = w(i,j)$ é o peso do arco de box de entrada j para a transição i.

Por exemplo, um grafo E-MFG pode ser representado pela matriz conforme apresentado na figura 5.16.

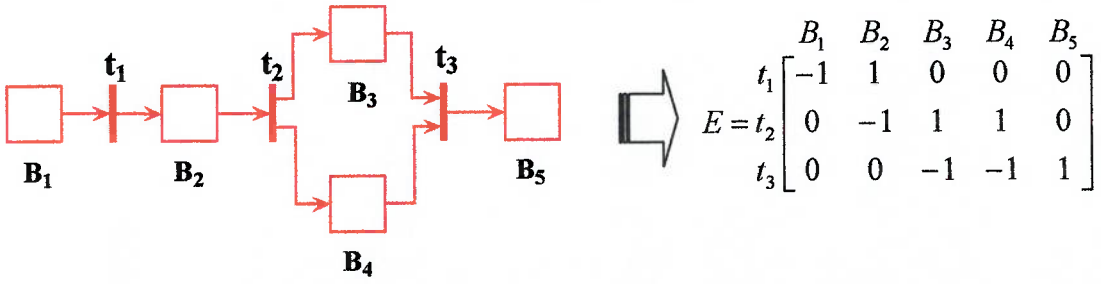


Figura 5.16 Exemplo de matriz que representa um grafo E-MFG.

5.3.5. Production Flow Schema (PFS)

O PFS [Miyagi, 1996] é utilizado para descrever, graficamente e conceitualmente, os processos relacionados com a produção de itens (peças, produtos, informações, etc.) sob a forma de seqüências de etapas de atividades e de distribuição. Como indicado na figura 5.17, o PFS consiste de nós de elementos-atividade, nós de elementos distribuidores e arcos de fluxo, que conectam seqüencialmente um tipo de nó ao outro.

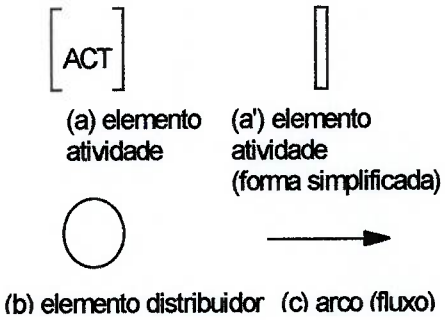


Figura 5.17 Elementos do PFS.

- Elemento-atividade. Representa um componente ativo do sistema que é responsável pela produção, transporte ou modificação dos itens. Corresponde a um macro-evento que representa a realização de certas unidades de operações;

- Elemento-distribuidor. Representa um componente passivo do sistema capaz de armazenar, permanecer em certos estados e tornar visíveis os itens. Corresponde a um lugar onde os itens são temporariamente alojados entre a entrada e a saída no elemento;

- Arcos de fluxo. Representam uma conexão lógica, proximidade física, direitos de acessos ou conexões diretas que indicam a direção do fluxo. Os arcos conectados na parte externa da atividade indicam o fluxo principal, e os arcos conectados na parte interna da atividade indicam o fluxo secundário.

A figura 5.18 ilustra um exemplo de uma linha de processamento e a figura 5.19 ilustra o correspondente em PFS.

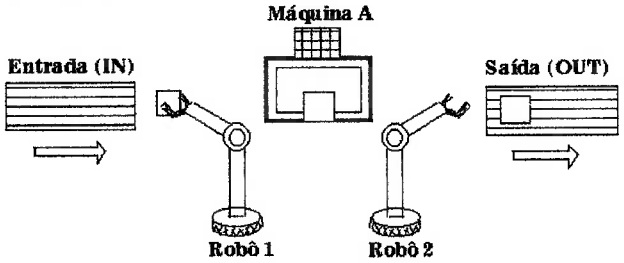


Figura 5.18 Exemplo de uma linha de processamento.

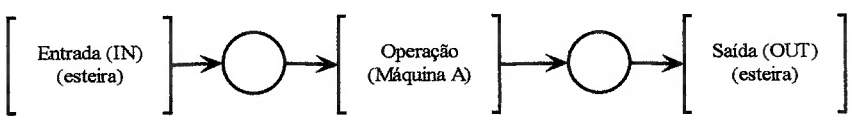


Figura 5.19 Modelo PFS que representa uma linha de processamento.

5.4 Geração das Regras Adicionais de Controle

O método de geração das regras adicionais de controle baseia-se nos seguintes fatos:

- A condição a ser controlada corresponde a condição de espera circular;
- O método para abordar o problema de "deadlock" corresponde ao de "deadlock avoidance");
- A arquitetura do sistema de controle a ser considerada é de um Sistema de Controle Hierárquico;
- As ferramentas de modelagem utilizadas para atingir o objetivo deste trabalho são o GAR e o E-MFG.

Considerando-se os aspectos levantados, a figura 5.20 apresenta o algoritmo do método adotado para a obtenção das regras adicionais em questão.

1 - Modelagem de cada processo através do GAR
2 - Obtenção do GAR Global
3 - Determinação dos Ciclos Fechados de Espera
4 - Geração das regras adicionais de controle para evitar o 'deadlock'
5 - Mapeamento do GAR Global para o E-MFG com as regras adicionais de controle: Síntese do Supervisório

Figura 5.20 Algoritmo do método adotado.

De forma a facilitar o entendimento, será utilizada o exemplo apresentado na tabela 5.1.

Tabela 5.1 Seqüência de utilização dos recursos pelos processos.

	Processo	Seqüência de utilização de recursos
1	A	R1 e R2
2	B	R2 e R1
3	C	R6, R3 e R2
4	D	R3 e R5
5	E	R5 e R6
6	F	R4 e R3
7	G	R1 e R4

5.4.1. Modelagem de Cada Processo Através do GAR

O GAR é criado de acordo com a seqüência pré-determinada de utilização dos recursos em cada processo. Os nós representam os recursos e os arcos orientados representam as etapas de cada processo.

Para a modelagem de cada processo através do GAR, é importante salientar que é fundamental inserir neste modelo dois elementos:

- Um elemento que representam a origem dos itens que serão processados (mapeamento da entrada de itens do processo) e;
- Um elemento que representa a saída dos itens processados.

Estes elementos de entrada e saída podem ser representados, por exemplo, através de "buffers", caso haja um armazenamento temporário desses itens no processo.

Considerando o estudo de caso em questão, a modelagem do processo *A* é apresentada na figura 5.21 onde é possível observar os "buffers" de entrada e saída. Desta forma, os modelos GAR de cada processo que compõe o sistema produtivo em estudo estão apresentados na figura 5.22.

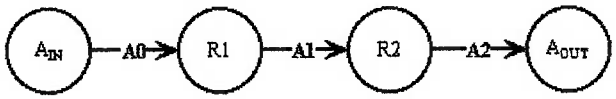


Figura 5.21 Modelo GAR do processo *A*.

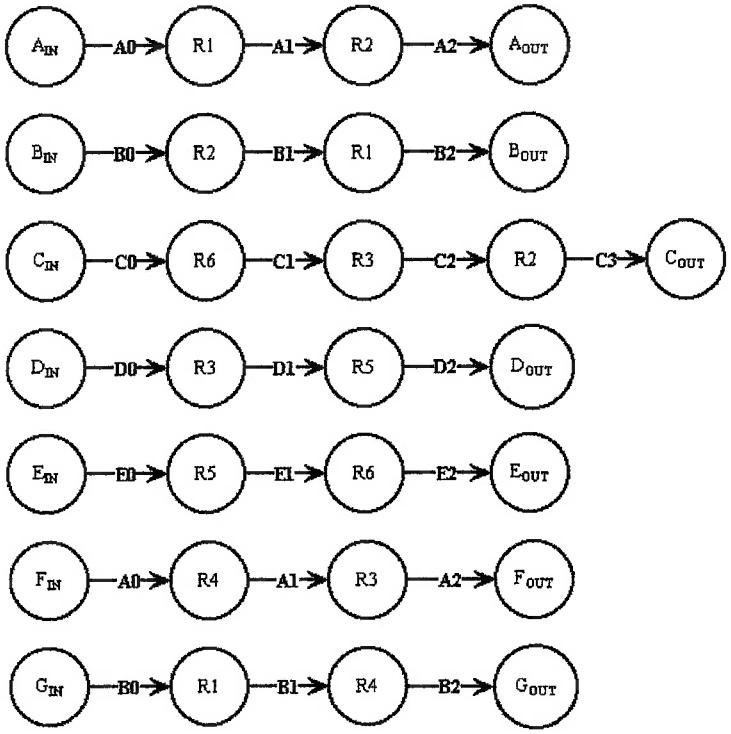


Figura 5.22 Modelo GAR de todos os processos.

O GAR pode ser representado na forma de matriz conforme a teoria de grafos [Netto, 2001]. Assim, para o GAR do processo *A*, temos:

$$G_A = \begin{array}{c} \\ R1 \\ R2 \\ A_{IN} \\ A_{OUT} \end{array} \begin{array}{c} \begin{array}{cccc} R1 & R2 & A_{IN} & A_{OUT} \end{array} \\ \begin{bmatrix} 0 & A1 & 0 & 0 \\ 0 & 0 & 0 & A2 \\ A0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{5.3}$$

5.4.2. Obtenção do GAR Global

A obtenção do GAR global é realizada aplicando-se o operador união da teoria dos conjuntos [Peterson, 81]. A técnica consiste em unir os nós comuns de todos os GAR individuais formando um único GAR que representa toda a seqüência de utilização dos recursos considerando a dimensão do processo global (figura 5.23).

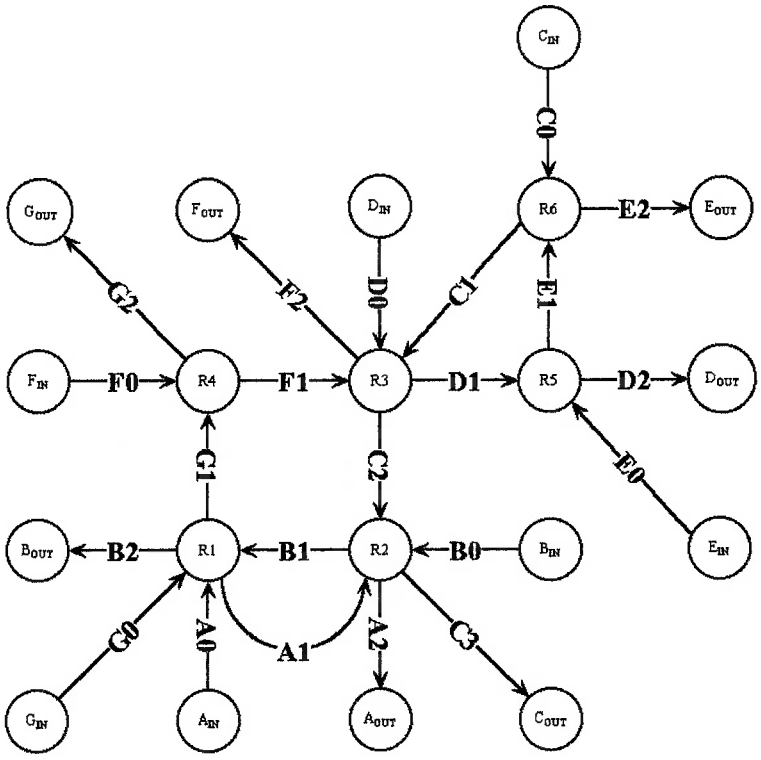


Figura 5.23 Modelo GAR Global.

Desta forma, temos a seguinte matriz equivalente para o processo global:

$$G = \begin{array}{c} \\ R1 \\ R2 \\ R3 \\ R4 \\ R5 \\ R6 \\ A_{IN} \\ A_{OUT} \\ B_{IN} \\ B_{OUT} \\ C_{IN} \\ C_{OUT} \\ D_{IN} \\ D_{OUT} \\ E_{IN} \\ E_{OUT} \\ F_{IN} \\ F_{OUT} \\ G_{IN} \\ G_{OUT} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccccc} R1 & R2 & R3 & R4 & R5 & R6 & A_{IN} & A_{OUT} & B_{IN} & B_{OUT} & C_{IN} & C_{OUT} & D_{IN} & D_{OUT} & E_{IN} & E_{OUT} & F_{IN} & F_{OUT} & G_{IN} & G_{OUT} \end{array} \\ \begin{bmatrix} 0 & A1 & 0 & G1 & 0 & 0 & 0 & 0 & 0 & B2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ B1 & 0 & 0 & 0 & 0 & 0 & 0 & A2 & 0 & 0 & 0 & C3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C2 & 0 & 0 & D1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & F2 & 0 & 0 \\ 0 & 0 & F1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G2 \\ 0 & 0 & 0 & 0 & 0 & E1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & E2 & 0 & 0 & 0 & 0 \\ A0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & B0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & E0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & F0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad (5.4)$$

5.4.3. Determinação dos Ciclos Fechados de Espera

Os CFEs são determinados utilizando-se o algoritmo (figura 5.24) descrito no trabalho de Santos Filho [2001].

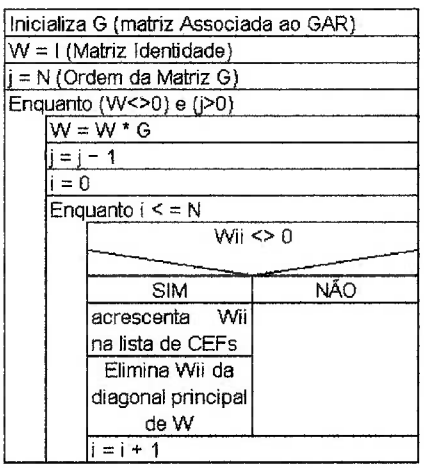


Figura 5.24 Algoritmo para obtenção dos CFEs de um GAR.

O algoritmo baseia-se na potenciação da matriz GAR até que a mesma se torne nula ou que o número de interações torne-se maior que a ordem da matriz. A cada interação formam-se seqüências concatenadas de etapas de processos e realiza-se uma verificação da existência de elementos na diagonal principal da matriz resultante. Os elementos que se encontrarem na diagonal principal da matriz são os CFE, isto é, a seqüência retornou à etapa inicial, ou seja, o ciclo fechou-se. Esses elementos são retirados da matriz e armazenados em uma lista de CFE. Aplicando-se este algoritmo no exemplo adotado, obtém-se os seguintes CFEs (figura 5.25):

- **E.1/C.1/D.1** ;
- **F.1/C.2/B.1/G.1** e,
- **B.1/A.1** .

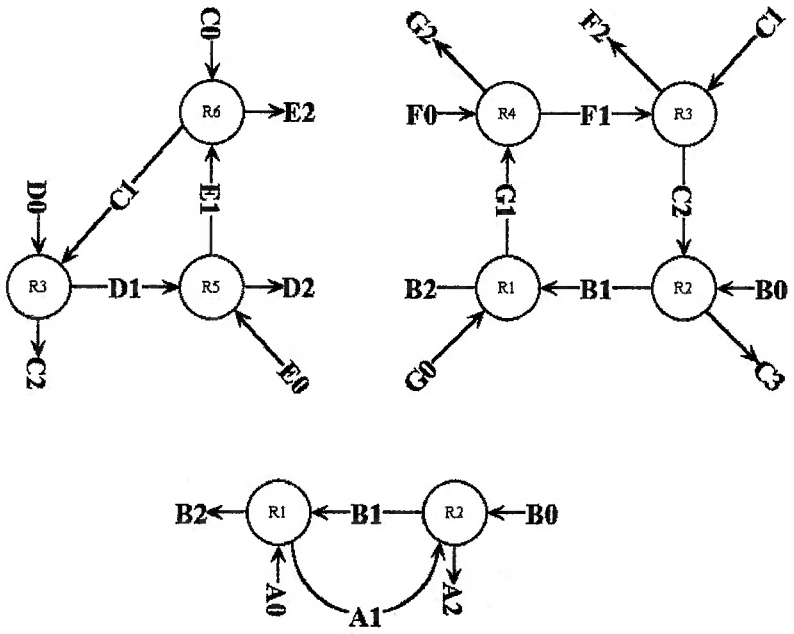


Figura 5.25 Os ciclos fechados de espera.

5.4.4. Dedução das Regras para Evitar o "Deadlock"

As regras adicionais de controle com o objetivo de evitar o "deadlock" devem:

- Determinar o estado anterior ao "deadlock";
- Controlar a quantidade de processos dentro de um CFE, inibindo as transições adequadas para que o sistema não evolua para esse estado

O estado anterior ao "deadlock" é definido como sendo a situação em que um CFE possui apenas um recurso disponível em todos os seus segmentos estando os demais recursos ocupados realizando algum tipo de processamento [Santos Filho, 2001].

A figura 5.26 apresenta um exemplo em que o nó de intersecção que representa o recurso 6 (*R6*) possui uma entrada de processo que pertence ao CFE (no caso é o processo *C*). Caso o recurso 5 (*R5*) estiver sendo utilizado pelo processo *E* e o recurso 3 (*R3*) estiver sendo utilizado pelo processo *D*, então a entrada do processo *C* deve ser inibida (figura 5.27), ou seja:

$$IF \quad (R3 = D) \quad AND \quad (R5 = E) \quad THEN \quad C0 = FALSE \tag{5.5}$$

Analogamente, são criadas as demais regras instanciadas a cada nó de intersecção que possui entrada de processo que pertencem ao CFE, conforme apresentado na tabela 5.2.

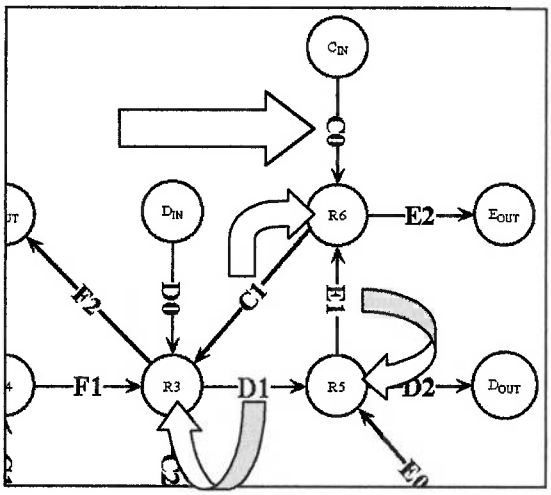


Figura 5.26 Exemplo de criação das regras adicionais de controle.

Tabela 5.2 Regras adicionais de controle para evitar o "deadlock".

CFE	Regras Adicionais de Controle
E1/C1/D1	IF (R3=D) AND (R5=E) THEN C0=FALSE
	IF (R5=E) AND (R6=C) THEN D0=FALSE
	IF (R3=D) AND (R6=C) THEN E0=FALSE
F1/C2/B1/G1	IF (R1=G) AND (R2=B) AND (R3=C) THEN F0=FALSE
	IF (R1=G) AND (R2=B) AND (R4=F) THEN C1=FALSE
	IF (R1=G) AND (R3=C) AND (R4=F) THEN B0=FALSE
	IF (R2=B) AND (R3=C) AND (R4=F) THEN G0=FALSE
B1/A1	IF (R1=A) THEN B0=FALSE
	IF (R2=B) THEN A0=FALSE

5.4.5. Síntese do Supervisório

O mapeamento é relativamente simples, uma vez que o modelo complementar de utilização dos recursos é um tipo de representação isomórfica do GAR em termos de descrição de estados alcançáveis [Santos Filho, 2000a]. Em síntese, cada nó no GAR é representado por um "box" no E-MFG e cada arco orientado no GAR representa uma

transição no E-MFG. Realizado o mapeamento do GAR para o E-MFG, necessita-se ainda:

- Incorporar as regras adicionais de controle;
- Criar conexões lógicas entre o controle de recursos e o controle de processos.

As regras adicionais de controle são incorporadas no modelo da seguinte forma:

- Emula-se através de elementos externos;
- Conectam-se logicamente ao modelo através de portas emuladoras externas, uma vez que o E-MFG é uma rede interpretada.

A conexão lógica do controle de recursos com o controle de processos deve ser realizada inserindo-se uma transição anterior a todos os 'boxes' de entrada de cada um dos processos (I_x) e outra transição posterior a todos os 'boxes' de saída de cada um dos processos do controle de recursos (O_x) e a estas transições são inseridas as portas emuladoras que conectam-se ao controle de processos. Desta forma, obtém-se a síntese do supervisório responsável pelo controle de utilização de recursos[2].

A figura 5.27 apresenta um exemplo de como é realizado o mapeamento do GAR para o respectivo E-MFG e a incorporação das regras adicionais de controle para evitar o "deadlock".

[2] Estas transições correspondem exatamente à representação de fontes e sorvedouros para mapear o início e fim da utilização de uma cadeia de recursos. É importante observar que o grafo marcado que emula o supervisório não possui marcação inicial de acordo com a interpretação semântica deste nível de controle discutida anteriormente.

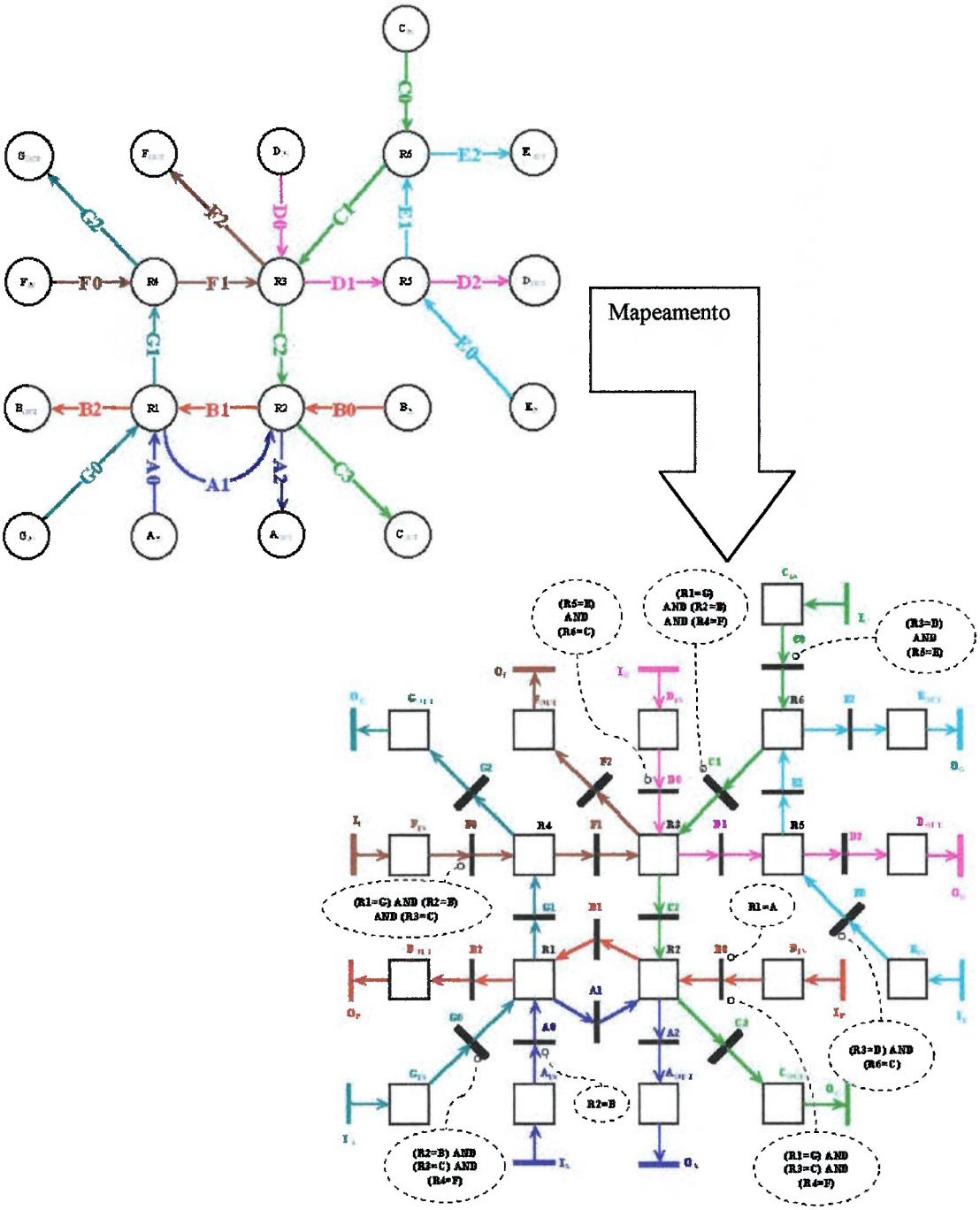


Figura 5.27 Modelo E-MFG com as regras adicionais de controle.

5.4.6. Síntese do Controle de Processos

A modelagem do controle de processos é baseado em uma metodologia estruturada denominada E-MFG/PFS (Enhanced Mark Flow Graph/Production Flow Schema) proposta por Santos Filho [1993; 2000a]. Cada processo é modelado aplicando-se o PFS para a modelagem funcional e através de refinamento sucessivo gera-se o E-MFG que resulta em um modelo funcional capaz de representar o comportamento dinâmico do processo. Por exemplo, o processo *A* utiliza seqüencialmente os recursos *R1* e *R2*, temos o seguinte modelo em E-MFG/PFS (figura 5.28):

Figura 5.28 Modelo E-MFG/PFS do processo *A*.

Realizando-se um refinamento sucessivo, a Etapa 1 pode ser detalhada em (figura 5.29):

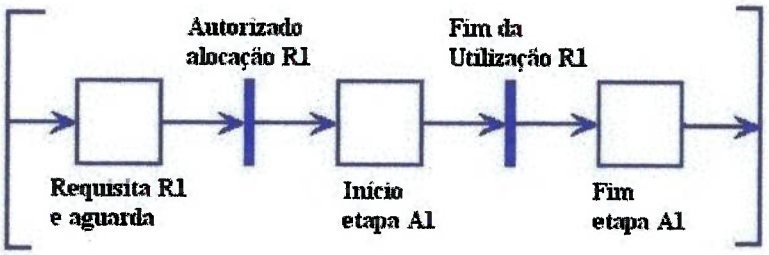


Figura 5.29 Detalhamento da Etapa 1.

Desta forma, realiza-se o refinamento para cada atividade e por fim gera-se o E-MFG do controle do processo. Durante o refinamento é necessário criar elementos ("boxes" e

transições) para conectarem-se logicamente à planta e ao controle de recursos através de portas emuladoras externas.

A conexão lógica com o controle de recursos deve ser realizada inserindo-se portas emuladoras para solicitações e autorizações de alocação de recursos.

Na figura 5.30 apresenta um exemplo de como o processo *A* é conectado ao controle de recursos e na figura 5.31 apresenta o exemplo do modelo do sistema de controle.

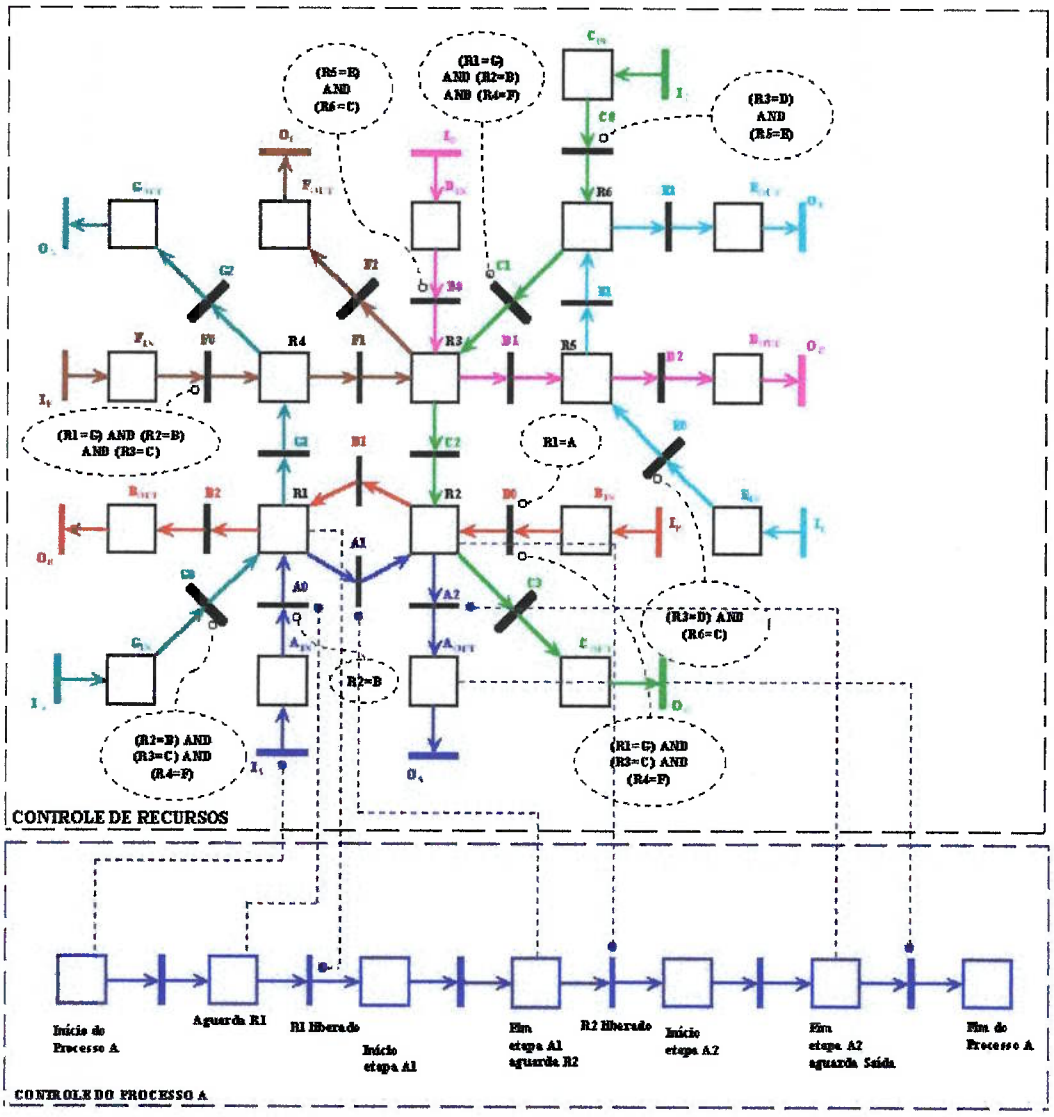


Figura 5.30 Conexão do controle de recursos e o controle do processo *A*.

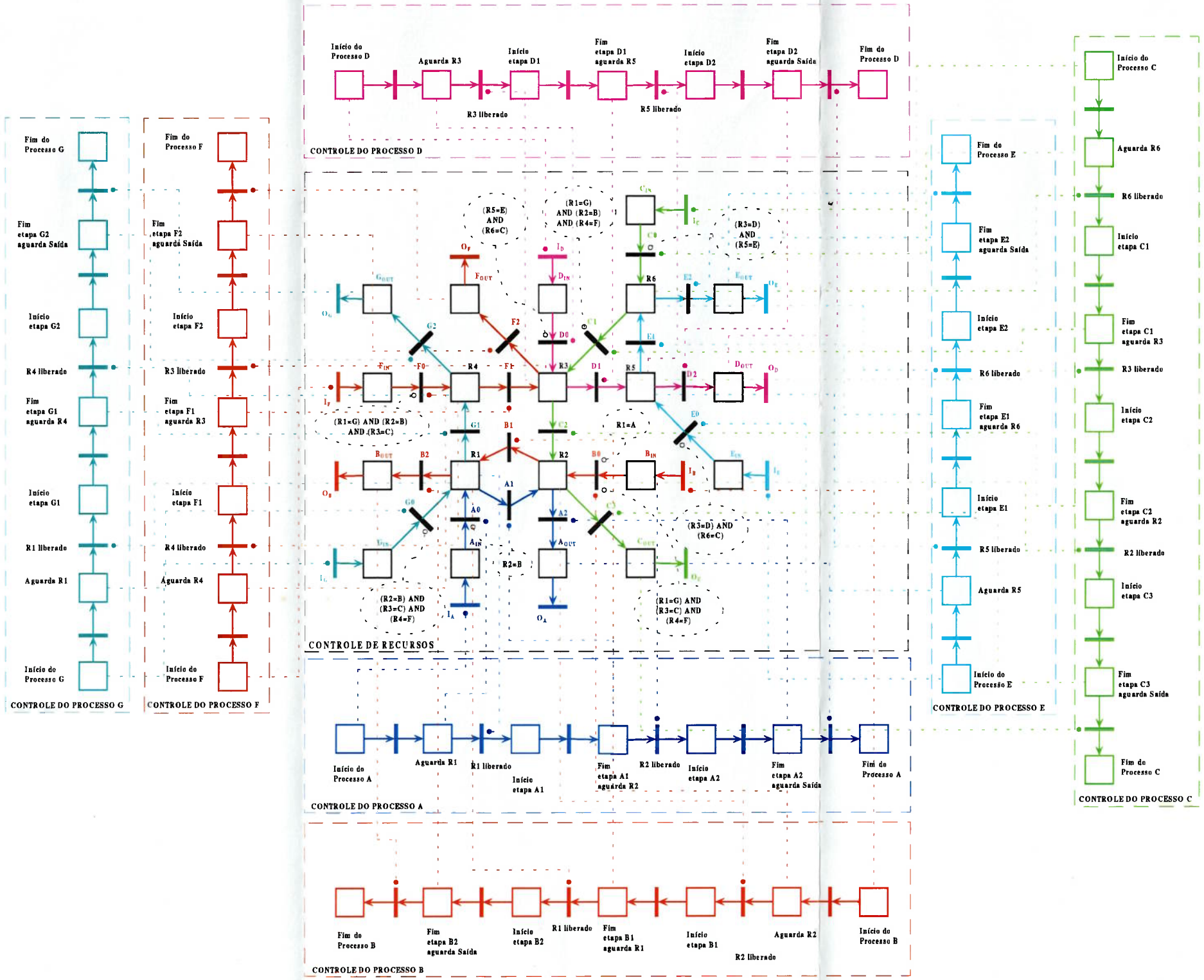


Figura 5.31 Exemplo do modelo do sistema de controle.

5.5 Observações Complementares

Este capítulo apresentou a arquitetura do sistema de controle de SPFs adotada e as diversas técnicas e métodos utilizados para a síntese do controle de utilização dos recursos e a síntese do controle de seqüenciamento dos processos.

Considerando-se o indeterminismo e complexidade presentes nos SPFs, foram abordados também os métodos e técnicas para a dedução de regras adicionais de controle que configuram um modelo de restrições baseado no método de evitar-se o "deadlock".

Desta forma, o próximo passo passa a ser a proposta de uma sistematização dos procedimentos fundamentais utilizados para a síntese do sistema de controle em questão. Este objetivo é contemplado no capítulo 6 a seguir.

CAPÍTULO 6
ABORDAGEM ALGORÍTMICA

6.1 INTRODUÇÃO

No capítulo 5 foi apresentada a descrição dos métodos adotados para a síntese do sistema de controle de SPFs. Neste capítulo é proposta uma sistematização desses métodos, ou seja, uma abordagem algorítmica que determine de forma automática o algoritmo de controle e as regras adicionais de controle para evitar o "deadlock" em SPFs.

Esta abordagem possibilitou a criação de uma ferramenta computacional que gera o algoritmos e as regras automaticamente, a partir de um conjunto de dados de entrada fornecidos, isto é, (i) Os tipos de recursos existentes e, (ii) os processos a serem executados descritos pelas respectivas seqüências de utilização dos recursos.

Primeiramente é apresentado os algoritmos projetados para o desenvolvimento da ferramenta computacional em uma plataforma IBM-PC e ambiente Windows/Linux.

Após o desenvolvimento da ferramenta, vários testes foram realizados utilizando diversos conjuntos de processos simultâneos. A partir dos resultados obtidos houve uma forte motivação para pesquisar-se uma nova forma de cálculo dos CFEs devido ao tempo de execução do algoritmo em questão ser relativamente longo em virtude da natureza da operação, isto é, um produto seqüencial de matrizes.

Portanto, quanto o maior o número de processos simultâneos e, conjuntamente, quanto maior o número individual de etapas de cada processo, a eficiência do algoritmo tornava-se comprometida.

Outro fator motivante correspondeu ao fato do algoritmo de potenciação prescindir da determinação dos ciclos reentrantes, necessitando, portanto, a implementação de outro algoritmo para selecionar os ciclos.

Baseado nesses aspectos é proposto um novo algoritmo para o cálculo dos CFEs utilizando o grafo GAR (representado pela matriz de adjacência correspondente). Este novo algoritmo é uma adaptação do algoritmo de pesquisa por profundidade, utilizado para pesquisa em estruturas de dados do tipo árvore.

Em seguida é apresentado uma breve descrição da **Notação-O** para determinar a complexidade de um algoritmo. E, desta forma, são realizadas comparações entre o algoritmo de potenciação e o algoritmo proposto.

6.2 ALGORITMOS DESENVOLVIDOS

6.2.1. Criação da Matriz de Adjacência GAR

A ordem da matriz de adjacência GAR é definida pela somatória da quantidade de tipos de recursos existentes e da quantidade de processos, sendo que cada processo possui uma entrada (IN) e uma saída (OUT);

Para cada processo, percorre-se a seqüência de utilização dos recursos da seguinte forma:

- Inicia-se pela linha que indica a entrada do processo (IN). Percorre-se esta linha até encontrar a coluna que indica o primeiro recurso da seqüência de utilização de recursos do processo em questão. Nesta célula armazena-se a etapa 0 do processo. Este procedimento é repetido até o último recurso e a correspondente saída do processo (OUT).

Por exemplo, para um processo *A* que utiliza 3 recursos seqüenciais: *R1*, *R2* e *R3*, tem-se uma entrada (A_{IN}), uma saída (A_{OUT}) e quatro etapas: *A.0*, *A.1*, *A.2* e *A.3*.

6.2.2. Criação da Representação Matricial do Grafo E-MFG

A matriz E-MFG é criada a partir da matriz que representa o grafo GAR. A criação da representação matricial do grafo E-MFG é necessária para a determinação das regras adicionais de controle para evitar o "deadlock" que será apresentado no próximo item.

O número de linhas da matriz corresponde ao total de etapas de todos os processos. O número de colunas da matriz corresponde à ordem da matriz de adjacência GAR, isto é, a somatória da quantidade de recursos existentes em gênero e de quantidade de processos, sendo que cada processo possui uma entrada (IN) e uma saída (OUT);

Para cada processo, valores são inseridos na matriz E-MFG percorrendo-se a matriz GAR a procura de etapas.

Por exemplo, dois processos A e B tal que a seqüência de utilização de recursos é :

$$A = \{R1, R2\} \wedge B = \{R2, R1\} \tag{6.1}$$

Obtém-se as seguintes matrizes GAR(6.2) e E-MFG (6.3):

$$G = \begin{array}{c} \\ R1 \\ R2 \\ A_{IN} \\ A_{OUT} \\ B_{IN} \\ B_{OUT} \end{array} \begin{array}{c} \begin{array}{cccccc} R1 & R2 & A_{IN} & A_{OUT} & B_{IN} & B_{OUT} \end{array} \\ \begin{bmatrix} 0 & A.1 & 0 & 0 & 0 & B.2 \\ B.1 & 0 & A.2 & 0 & 0 & 0 \\ A.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & B.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{6.2}$$

$$E = \begin{array}{c} \\ A.0 \\ A.1 \\ A.2 \\ B.0 \\ B.1 \\ B.2 \end{array} \begin{array}{c} \begin{array}{cccccc} R1 & R2 & A_{IN} & A_{OUT} & B_{IN} & B_{OUT} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{6.3}$$

6.2.3. Cálculo dos CFE

Diversos autores propõem a utilização e/ou modificações baseado no algoritmo de pesquisa por profundidade [Viswanadham et al.; 1990; Tanenbaum, 1995; Yoon e Lee, 2000a; 2000b]. Em Tanenbaum [1995] é utilizando um algoritmo de pesquisa por profundidade que detecta o estado de 'deadlock'. Este algoritmo toma cada um dos nós do grafo seqüencialmente como se fosse a raiz de uma árvore e faz uma pesquisa profunda. Cada nó visitado é adicionado a uma lista de nós. Se alguma vez visitar um nó encontrado anteriormente, então foi descoberto um ciclo e o algoritmo é encerrado. Ao exaurir todos os arcos de determinado nó, retorna ao nó anterior. Se voltar à raiz e não puder continuar, o subgrafo demarcado até o presente nó não possui nenhum ciclo. Sendo esta propriedade verdadeira para todos os nós, o grafo estará livre de ciclos e, portanto, não apresenta condição de 'deadlock'.

O algoritmo proposto neste trabalho utiliza o mesmo artifício porém somente será encerrado quando visitar todos os nós, determinando assim, todos os CFEs. Utilizando-se o exemplo apresentado no capítulo 5 temos o seu respectivo GAR (figura 6.1):

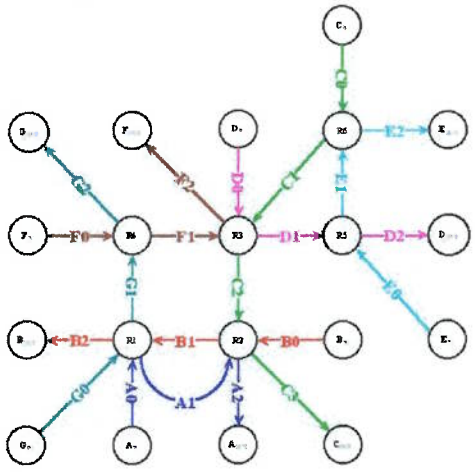


Figura 6.1 Exemplo no grafo GAR apresentado no capítulo 5.

O algoritmo realiza as seguintes etapas seqüencialmente conforme apresentado na figura 6.2.

1. Adiciona o nó inicial
2. Procura arcos que saem do ultimo nó da lista de nós
3. Encontrado algum caminho não descoberto, adiciona o arco à lista de arcos e o nó de destino à lista de nós. Do contrário, retira-se o último nó e o último arco das respectivas listas e executa a etapa 2.
4. Verifica se o caminho retornou a algum nó já descoberto.
5. Se não foi encontrado nenhum CFE executa o passo 2, do contrário adiciona-se o CFE à lista de CFE, retira-se o último nó e o último arco das respectivas listas e verifica se retornou ao nó inicial. Se retornou então o algoritmo deve terminar, do contrário executa a etapa 2.

Figura 6.2 O Algoritmo de pesquisa por profundidade.

O algoritmo é aplicado no exemplo iniciando-se pelo nó que representa o recurso 1 (*R1*) adicionando-o a uma lista (figura 6.3). Procura-se os arcos que saem deste nó, no exemplo tem-se dois arcos (*A.1* e *G.1*). Adicionando-se o arco *A.1* à lista de arcos, adiciona-se o nó de destino, no exemplo tem-se o nó *R2* que representa o recurso 2. Neste ponto é realizado uma verificação à procura de CFE, isto é, se o caminho percorrido retornou a algum nó já descoberto.

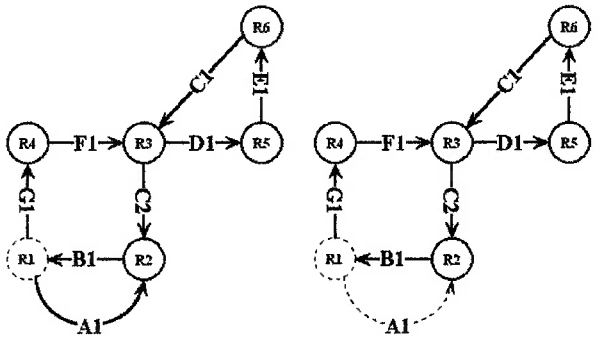


Figura 6.3 Algoritmo de pesquisa, exemplo 1.

Não encontrando nenhum ciclo, o algoritmo continua procurando arcos que saem do último nó descoberto. No exemplo o arco *B.1* é adicionado à lista de arcos e o nó de destino, representado por *R1*, à lista de nós (figura 6.4). Neste ponto ao realizar a verificação na lista de nós, descobre-se que o caminho retornou ao recurso 1. Logo, foi descoberto um CFE.

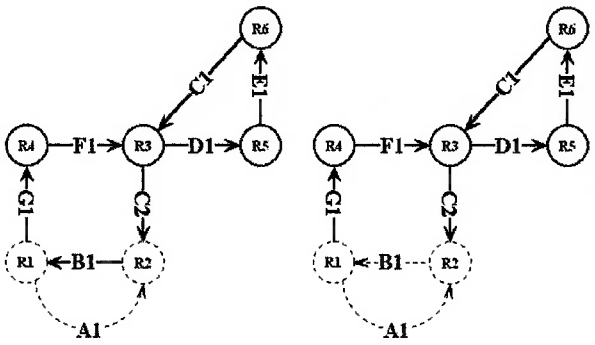


Figura 6.4 Algoritmo de pesquisa, exemplo 2.

Este CFE é adicionado à lista de CFE. O último nó e o último arco descoberto são retirados das respectivas listas (nós e arcos); no exemplo o nó *R1*e o arco *B.1*. Assim, continuando-se a seqüência, verifica-se se existe algum outro arco que sai do último nó da lista; no exemplo é o nó *R2*. Não encontrado mais nenhum arco não descoberto, este nó e o último arco (*A.1*) são retirados das listas, assim, retorna-se ao nó de origem *R1*. Novamente, verifica-se se existe algum arco não descoberto; no exemplo o arco *G.1* é adicionado à lista de arcos e o respectivo nó de destino *R.4*.

Desta forma, o algoritmo percorre todo grafo GAR à procura de CFE até que o caminho retorne à origem, isto é, o nó *R1* e não haja mais nenhum caminho não descoberto. Nas figuras 6.5 e 6.6 são apresentadas a determinação dos demais CFE no exemplo proposto.

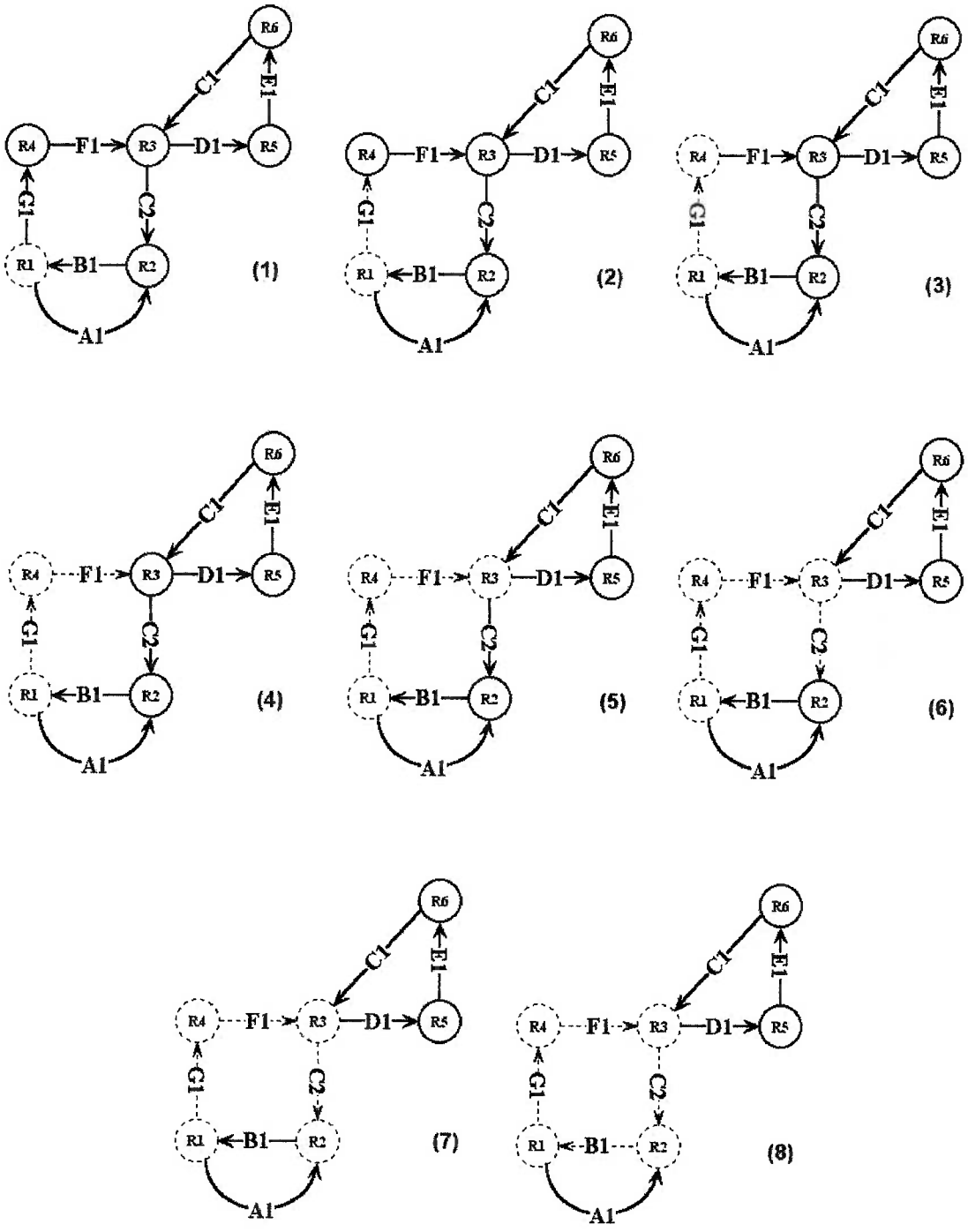


Figura 6.5 Algoritmo de pesquisa, exemplo 3.

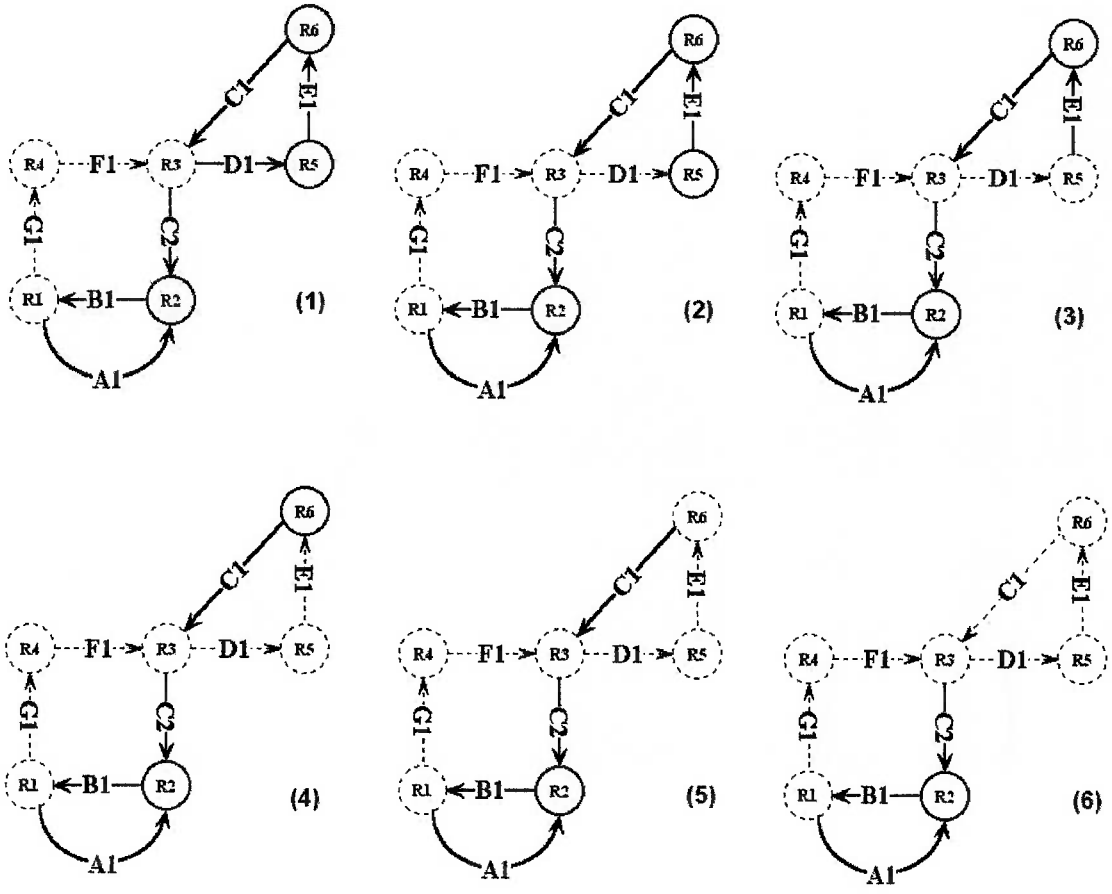


Figura 6.6 Algoritmo de pesquisa, exemplo 4.

6.2.4. Dedução Automática das Regras Adicionais de Controle para Evitar o "Deadlock"

Neste trabalho é proposto um algoritmo para a determinação automática das regras em questão [Nakamoto et al., 2001a; Nakamoto et al. 2001b] conforme apresentado na figura 6.7. Este algoritmo utiliza como parâmetro de entrada os CFEs obtidos resultado do cálculo dos CFEs.

```
Para cada CFE = R
    Q = Quantidade de Recursos do CFE
    que possuem entrada de processos
    pertencentes ao CFE
    para i=1 até Q faça
        Inibir a entrada do processo que
        pertence ao CFE em R[i], admitindo-
        se que os demais recursos estão
        sendo utilizados por outros
        processos que pertencem ao
        mesmo CFE.
```

Figura 6.7 Algoritmo proposto para determinar as regras adicionais de controle.

Para cada CFE verifica-se a condição pré "deadlock" e inibe a entrada de processo que pertence ao ciclo em questão, ou seja, inibe a entrada de processo no recurso disponível se todos os demais recursos do ciclo estiverem sendo utilizados por outros processos que pertencem ao ciclo. Em síntese, o algoritmo cria regras do tipo "IF... THEN..." da seguinte maneira:

- A condição "IF" irá descrever o estado anterior ao "deadlock", isto é, quando um recurso (nó) que possui entrada de processo pertencente ao CFE está disponível, considera-se que os demais recursos (nós) estão sendo utilizados por processos que pertencem ao CFE em questão;

- A ação "THEN" prescreve a ação que tem como objetivo inibir a entrada do processo no recurso disponível considerado na condição "IF".

Desta forma, será deduzida uma regra para cada nó do CFE que possui entrada de processo que pertence ao ciclo em questão.

6.3 Comparação entre os Algoritmos: Potenciação x Pesquisa

6.3.1. Análise de Algoritmos

Um programa de computador consiste de estrutura de dados e algoritmos [Tenenbaum, et al., 1995]. Uma estrutura de dados é um meio para armazenar e organizar dados com o objetivo de facilitar o acesso e as modificações [Cormen et al., 2001]. Logo, os algoritmos são utilizados para o acesso e as modificações desse dados.

Em Cormen et al. [2001] faz-se a seguinte afirmação:

> *"Um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. "*

Em Ziviani [1999] observa-se que:

> *"O algoritmo é uma seqüência de ações precisas, não ambíguas e executáveis para obtenção de uma solução para um determinado tipo de problema."*

A entrada de valor ou conjunto de valores são as informações para a resolução de um determinado problema proposto, logo, a saída de valor ou conjunto de valores será a solução para o respectivo problema proposto.

A solução de um determinado tipo de problema pode ser alcançado através de um conjunto de diferentes algoritmos, cada qual com suas respectivas técnicas que proporcionam vantagens e desvantagens, influenciando assim a eficiência do algoritmo

propriamente dito. Desta forma, é considerado os seguintes aspectos da eficiência [Tenenbaum et al., 1995]: tempo e espaço. Um algoritmo deve ser tão eficiente quanto possível, isto é, deve resolver o problema o mais rapidamente possível e exigir o mínimo de recursos necessários do computador, porém, freqüentemente um desses aspectos é otimizado à custa do outro [Tenenbaum et al., 1995].

Outro aspecto a ser considerado é referente à quantidade de informação a ser manipulada. A maioria dos algoritmos não atua da mesma forma em diferentes casos onde há variação na quantidade de informação, isto é, geralmente a eficiência de um algoritmo varia de acordo com os dados que são processados por ele [Loudon, 2000].

Esta eficiência é mensurável através de funções de complexidade do algoritmo, realizando-se uma análise do comportamento assintótico do mesmo. Neste tipo de análise é comum utilizar-se a **Notação-O**.

6.3.1.1. Função de Complexidade

Um algoritmo pode ser estudado através do método de análise matemática, ou seja, é possível mensurar a eficiência através da função de complexidade do algoritmo. A função de complexidade *f(n)* é a medida de tempo[1] necessário para executar um algoritmo para um problema de tamanho *n* [Ziviani, 1999].

[1] A função de complexidade de tempo não representa o tempo propriamente dito, mas o número de vezes que determinada operação é executada.

Para a elaboração da função de complexidade de um algoritmo é necessário distinguir três casos que devem ser considerados [Aho et al., 1974; Tenenbaum et al., 1995; Ziviani, 1999; Loudon, 2000]:

- Melhor caso: Corresponde ao menor tempo de execução do algoritmo;
- Pior caso: Corresponde ao maior tempo de execução do algoritmo;
- Caso médio: Corresponde à média dos tempos de execução do algoritmo.

Normalmente, na maioria dos casos, pode-se esperar a ocorrência do caso médio, porém, existem quatro razões pelas quais os algoritmos são geralmente analisados a partir dos piores casos [Loudon, 2000]:

- Muitos algoritmos executam suas piores performances boa parte do tempo. Um exemplo seria um algoritmo de pesquisa que não encontra o que está buscando;
- Melhor caso não fornece muita informação sobre a eficiência do algoritmo. Por exemplo, um algoritmo de pesquisa que localiza um elemento em uma única inspeção;
- Dificuldade em determinar o caso médio;
- O pior caso fornece um limite máximo que permite a sua análise.

Em Ziviani [1999] é apresentado um exemplo: um problema de acesso aos registro de um arquivo utilizando-se um algoritmo de pesquisa seqüencial. Este algoritmo examina cada registro na ordem em que se encontram no arquivo até que o registro procurado

seja encontrado ou que fique determinado que o mesmo não se encontra no arquivo. Desta forma, os casos a serem considerados são:

- Melhor caso: Quando o registro é encontrado na primeira verificação, ou seja, $f(n)=1$;
- Pior caso: Sendo n a quantidade total de registros, o registro é encontrado na última verificação, ou seja, $f(n)=n$;
- Caso médio: Sendo n a quantidade total de registros, o registro é encontrado na metade da quantidade de registros, ou seja, $f(n)=(n+1)/2$.

6.3.1.2. Comportamento Assintótico

A escolha do algoritmo depende do parâmetro n, uma vez que o tempo necessário para resolver o problema cresce à medida que o valor de n cresce [Ziviani, 1999].

Desta forma, a análise do algoritmo é realizada para valores grandes de n. Supondo-se que um algoritmo possui a função de complexidade $f(n)=0,01*n^2+10*n$. Para valores pequenos de n, a quantidade $10*n$ domina a quantidade $0,01*n^2$. Entretanto, à medida que o valor de n aumenta, o termo $0,01*n^2$ domina o termo $10*n$ até que a função torna-se proporcional a n^2, isto é, a função $f(n)=0,01*n^2+10*n$ é da ordem da função n^2. Esta análise fornece o comportamento assintótico da função $f(n)$.

6.3.1.3. Notação-O

A **Notação-O** é uma maneira formal de expressar a performance de um algoritmo. Esta notação expressa o limite do comportamento da função dentro de um fator constante, ou seja, se $f(n) = O(g(n))$, onde se lê $f(n)$ é assintoticamente limitada por $g(n)$, então, existe duas constantes c e m tais que $f(n) \leq c*g(n)$, para $n \geq m$ [Tenenbaum et al., 1995; Ziviani, 1999; Loudon, 2000; Cormen et al., 2001] (figura 6.8).

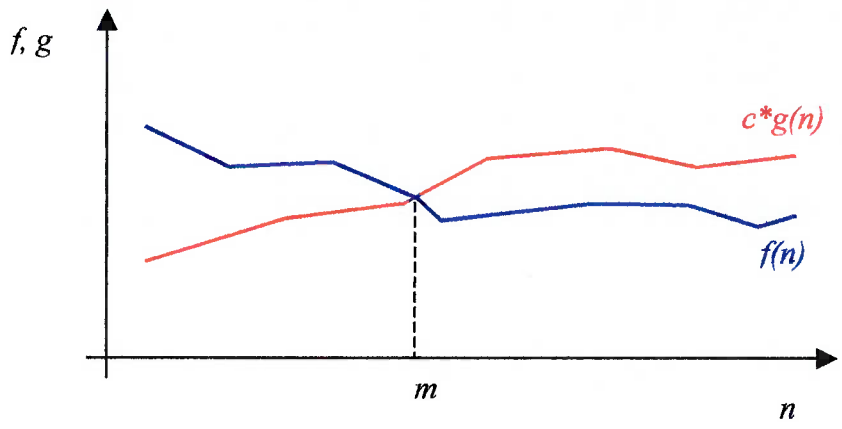


Figura 6.8 Dominação assintótica de $g(n)$ sobre $f(n)$.

Se $f(n)$ é uma função de complexidade para um algoritmo F, então $O(f)$ é considerada a complexidade assintótica ou o comportamento assintótico do algoritmo F. Por exemplo, seja $f(n)=n^2+100*n$, logo $f(n)$ é $O(n^2)$, quando $c=2$ e $m=100$. Isto porque $n^2+100*n \leq 2*n^2$ para $n \geq 100$.

A complexidade pode ser classificada em:

- Complexidade Constante – $O(1)$: O uso do algoritmo independe do tamanho de n;

- Complexidade Logarítmica – $O(\log n)$, $O(n*\log n)$: O algoritmo resolve um problema quebrando-o em problemas menores, resolvendo cada um deles independentemente e depois unindo as soluções;

- Complexidade Linear – $O(n)$: O tempo de execução é proporcional ao valor de n;

- Complexidade Polinomial – $O(n^k)$ para $k>1$: Os algoritmos desta classe são úteis para resolverem problemas onde o valor de n é relativamente pequeno;

- Complexidade Exponencial – $O(d^n)$ para $d>1$ e $n>1$: Os algoritmos desta classe não são úteis sob o ponto de vista prático.

6.3.2. Comparação dos Algoritmos

Na literatura é calculado que a operação de produtos de matrizes possui complexidade $O(n^3)$ [Aho et al., 1974; Cormen et al., 2001]. O algoritmo de potenciação de matrizes executa uma seqüência de operações desta natureza até que a matriz torne-se nula ou, no pior caso, até que a quantidade de interações seja igual ao módulo da matriz. Desta forma, a complexidade do algoritmo de potenciação de matrizes possui no mínimo complexidade $O(n^4)$. De fato, calculando-se a função de complexidade do algoritmo de potenciação (figura 6.9), tem-se:

$$f(n) = n^4 + 3*n^3 + 4*n^2 + 4*n + 6 \qquad (6.4)$$

Logo, a função $g(n)=2*n^4$ para todo $n \geq 6$, logo a complexidade do algoritmo é $\boldsymbol{O(n^4)}$.

```
{Procedimento para determinar os CFE}
procedure TMainForm.CalcCFE;
var
  k:Integer;

  {Procedimento para retirar os CFE da diagonal
  e zerar o seu conteúdo}
  procedure ZeraDiagonal;
  var
    i:Integer;
  begin
    for i:=1 to SERecurso.Value do
      if FW[i-1,i-1]<>'0'
        then
          begin
            CFEEtapa.Add(FW[i-1,i-1]);
            FW[i-1,i-1]:='0';
          end;
  end;

  {Inicia Matriz G
  Cria-se uma matriz de acordo com os dados inseridos no Grid:
  Quantidade de recursos}
  procedure IniciaMatrizG;
  var
    C,L:Integer;
  begin
    SetLength(FG,SERecurso.Value ,SERecurso.Value );
    SetLength(FW,SERecurso.Value ,SERecurso.Value );
    CFEEtapa.Clear;
    CFERecurso.Clear;
    for C:=1 to SERecurso.Value do
      for L:=1 to SERecurso.Value do
        begin
          FG[C-1,L-1]:=SGGAR.Cells[C,L];
          if C=L
            then FW[C-1,L-1]:='1'
            else FW[C-1,L-1]:='0';
        end;
  end;

  {Procedimento que realiza a potencialização da Matriz}
  procedure MultiplicaMatriz;
  var
    C,L, i:Integer;
    Aux:String;
    Temp:TMatriz;
  begin
    SetLength(Temp,SERecurso.Value ,SERecurso.Value );
    for L:=1 to SERecurso.Value do
    for C:=1 to SERecurso.Value do
      begin
        Aux:='';
        for i:=1 to SERecurso.Value do
          begin
            if (FG[C-1,i-1]<>'0') and (FW[i-1,L-1]<>'0')
              then
                begin
                  if Aux<>'' then Aux:=Aux+'/';
                  if FW[i-1,L-1]='1'
                    then Aux:=Aux+FG[C-1,i-1]
                    else Aux:=Aux+FW[i-1,L-1]+'/'+FG[C-1,i-1];
                end;
          end;
        if Aux=''
          then Temp[C-1,L-1]:='0'
          else Temp[C-1,L-1]:=Aux;
      end;
    FW:=Temp;
    Temp:=nil;
  end;

  {Função que verifica se a matriz está zerada.
  Caso alguma célula da matriz for diferente de 0
  então a matriz não está zerada.}
  function MatrizZerada:Boolean;
  var
    Resp:Boolean;
    C,L:Integer;
  begin
    Resp:=True;
    for C:=1 to SERecurso.Value do
      begin
        for L:=1 to SERecurso.Value do
          if FW[C-1,L-1]<>'0'
            then
              begin
                Resp:=False;
                Break;
              end;
        if Resp=False then Break;
      end;
    Result:=Resp;
  end;

begin
  MemoCFE.Clear;
  IniciaMatrizG;
  k:=0;
  while (k<=SERecurso.Value) do
    begin
      MultiplicaMatriz;
      if MatrizZerada
        then Break
        else ZeraDiagonal;
      inc(k);
    end;
  EtapaRepetida(CFEEtapa);
  CFERecurso:=CFER(CFEEtapa);
  UpdateView;
end;
```

Figura 6.9 Algoritmo de potenciação da matriz GAR.

O algoritmo de pesquisa possui complexidade *O(n)* sendo que n é igual ao número de nós mais o número de arcos do grafo [Aho et al., 1974; Cormen et al., 2001], isto se a estrutura de dados que o algoritmo pesquisa é uma lista de adjacência [Cormen et al., 2001; Netto, 2001]. No presente trabalho, a pesquisa é realizada sob a matriz de adjacência e, portanto, no pior caso o algoritmo possui complexidade $O(n^2)$.

Algumas mudanças significativas foram feitas no algoritmo, tal que, não poderá ser pesquisado alguma linha da matriz que anteriormente foi visitada.

Portanto, é possível concluir que o algoritmo de pesquisa possui maior eficiência que o algoritmo de potenciação de matrizes quando a quantidade de nós e arcos é relativamente grande.

6.3.3. Estudo de Caso

A comparação dos algoritmos é baseada no seguinte sistema:

a) Seis instâncias de recursos, e;

b) Cada processo possuindo 4 etapas.

Na tabela 6.1 é apresentado os casos utilizados para a comparação. A coluna P indica os processos, a coluna Etapas indica a seqüência de utilização dos recursos e a coluna Casos indica a quantidade de processos em questão, isto é, o caso 1 é considerado apenas o processo *A* e *B*, o caso 3 é considerado o processo *A*, *B*, *C* e *D*.

Para a execução dos algoritmos foram apenas considerados os cálculos dos CFE em milésimos de segundo em um computador IBM-PC com as seguintes configurações:

- Processador Pentium 4;
- 128 Mbytes de memória RAM;
- Sistema Operacional Windows 98 SE.

Tabela 6.1 Casos utilizados na comparação de algoritmos.

Casos	P	Etapas			
1	*A*	1	2	3	4
	B	4	3	2	1
2	*C*	5	3	4	2
3	*D*	1	3	2	5
4	*E*	3	5	3	1
5	*F*	4	6	2	5
6	*G*	2	5	6	3
7	*H*	3	6	2	5
8	*I*	5	1	4	6
9	*J*	1	6	4	5
10	*K*	5	6	2	3
11	*L*	3	2	6	5
12	*M*	5	4	6	1
13	*N*	5	2	6	4
14	*O*	1	3	2	4
15	*P*	2	6	5	3
16	*Q*	6	4	1	6
17	*R*	2	5	6	2
18	*S*	2	6	2	4
19	*T*	5	3	1	3
20	*U*	4	3	4	5

O valor determinado é a média de 10 tomadas de tempo por cada caso. Desta forma, foram determinadas os seguintes dados:

a) Algoritmo de Potenciação (tabela 6.2 e figura 6.10):

Tabela 6.2 Os tempos utilizando o algoritmo de potenciação da matriz.

	Algoritmo de Potenciação									
	1	2	3	4	5	6	7	8	9	10
1	0	0	60	270	440	1260	2860	8300	34160	56190
2	0	60	60	330	430	1210	2860	8290	33950	55800
3	0	60	50	380	440	1210	2850	8300	33340	55250
4	50	0	110	330	430	1270	2870	8300	33300	55310
5	0	50	60	440	440	1210	2860	8350	34220	55410
6	0	0	50	330	440	1210	2800	8400	34110	54760
7	60	50	110	330	430	1260	2860	8300	33300	55610
8	60	60	50	270	500	1210	2870	8290	33340	56680
9	0	60	50	330	440	1210	2850	8300	33340	55250
10	0	50	110	330	430	1270	2860	8350	33210	55420
Ms	17	39	71	334	442	1232	2854	8318	33627	55568

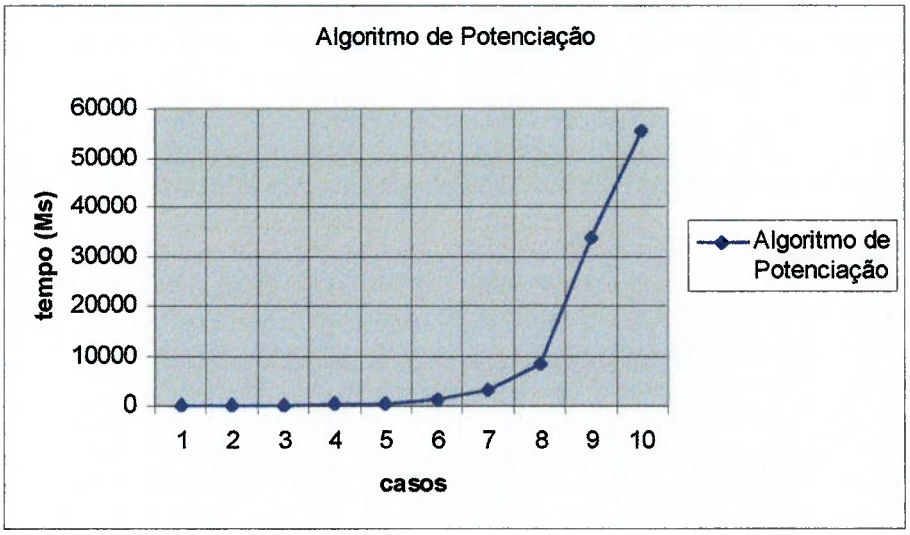


Figura 6.10 Gráfico dos tempos utilizando o algoritmo de potenciação.

b) Algoritmo de Pesquisa por Profundidade (tabela 6.3 e figura 6.11):

Tabela 6.3 Os tempos utilizando o algoritmo de pesquisa por profundidade.

	Algoritmo de Pesquisa por Profundidade									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	160	600	710
2	0	0	0	0	0	0	60	160	610	710
3	0	0	0	0	0	0	50	170	600	720
4	0	0	0	0	0	0	60	160	660	770
5	0	0	0	0	0	50	0	160	600	710
6	0	0	0	0	0	0	50	160	610	710
7	0	0	0	0	0	0	60	160	610	720
8	0	0	0	0	0	0	0	170	600	700
9	0	0	0	0	0	0	60	160	600	710
10	0	0	0	0	0	0	50	160	620	710
Ms	0	0	0	0	0	5	39	162	611	717

11	12	13	14	15	16	17	18	19	20
1380	2910	4450	6320	6980	9280	10160	10820	11090	11650
1350	2950	4230	6310	6810	9230	9890	10550	11150	11480
1290	2950	4120	6260	6980	9220	10270	10600	10990	11530
1350	2910	4230	6250	6760	9280	10250	10520	10990	11520
1380	2910	4230	6320	6800	9250	9950	10850	10990	11450
1350	2910	4210	6320	6980	9280	9890	10820	11050	11480
1310	2950	4100	6320	6980	9210	9890	10820	11050	11650
1350	2910	4450	6100	6950	9280	10010	10510	11150	11650
1350	2910	4210	6340	6980	9230	9890	10550	10990	11450
1380	2910	4210	6320	6850	9280	9980	10820	11090	11480
1349	2922	4244	6286	6907	9254	10018	10686	11054	11534

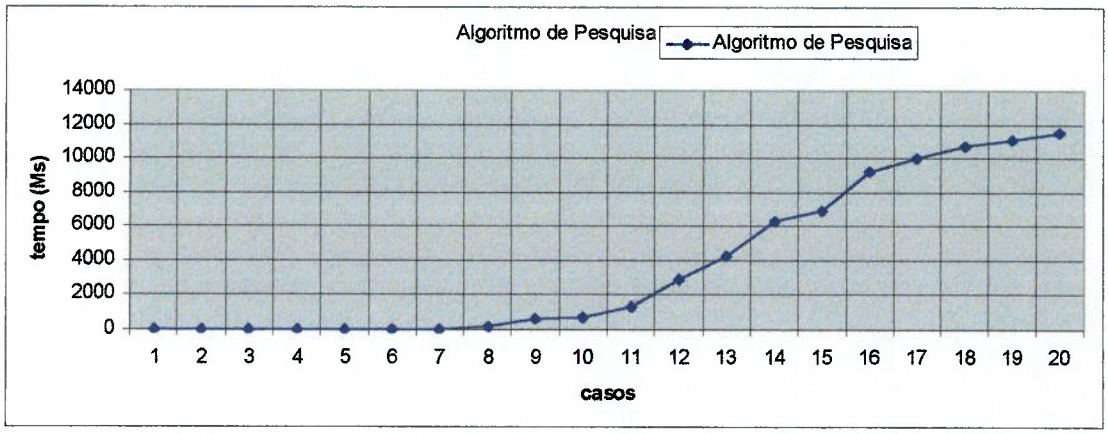


Figura 6.11 Gráfico dos tempos utilizando o algoritmo de pesquisa por profundidade.

Comparando-se os dois algoritmos em um único gráfico, temos (figura 6.12):

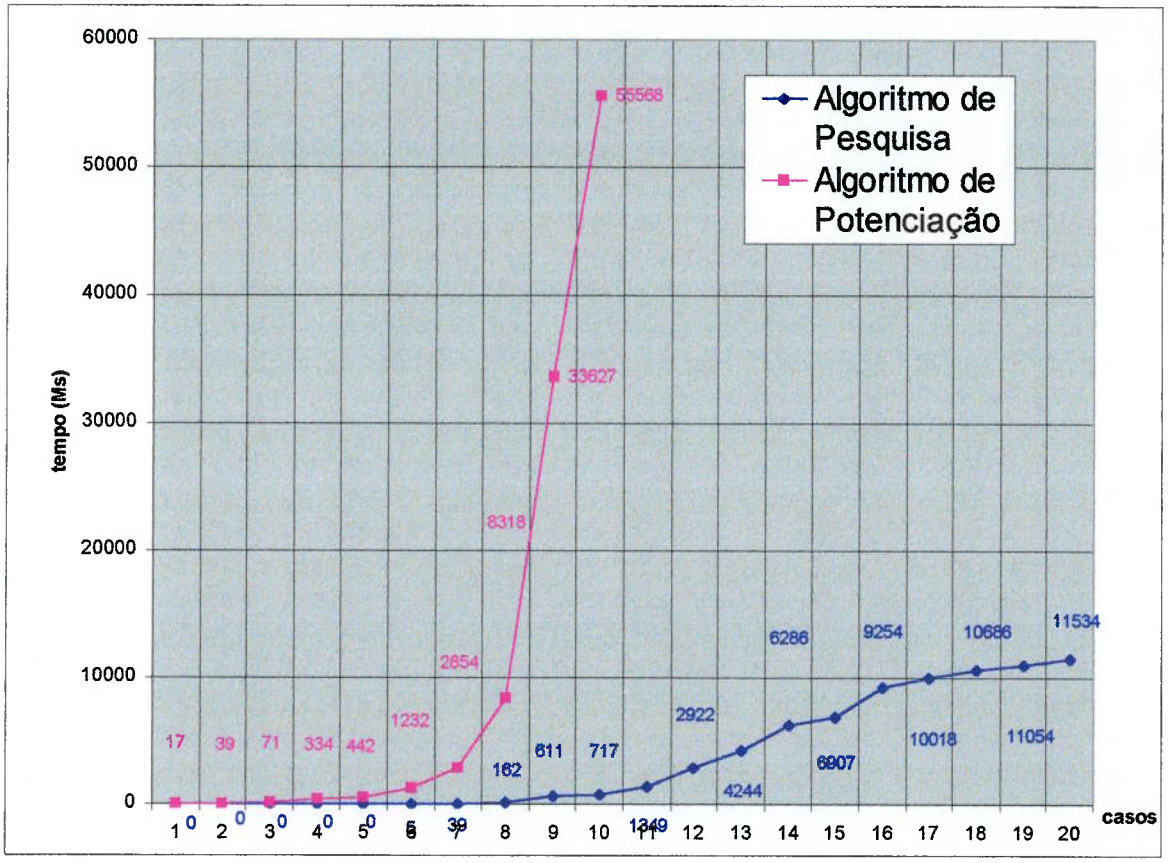


Figura 6.12 Comparação dos algoritmos.

Os tempos de execução do algoritmo de potenciação foram calculados até o décimo caso pois a partir deste, tornou-se muito longo a marcação do tempo. Mesmo assim, é possível visualizar a diferença nos tempos de execução através dos gráficos.

Desta forma, com o aprimoramento da geração automática do algoritmo de controle, viabiliza-se a possibilidade de geração do controle de forma rápida para reagir às mudanças no ambiente.

6.4 Observações Complementares

Neste capítulo apresentou a abordagem algorítmica que possibilita a determinação automática do algoritmo de controle e das regras adicionais de controle para evitar o "deadlock" em SPFs, permitindo a implementação de uma ferramenta computacional.

Após a implementação utilizando-se o algoritmo de potenciação da matriz, foi observado as seguintes limitações:

a) tempo de execução é diretamente proporcional à quantidade de tipos de recursos e a relação entre os recursos, isto é, etapas de processo;

b) São calculados todos os CFE porém em forma de ciclos reentrantes, necessitando-se assim uma separação dos CFE.

Estas limitações devem-se à natureza da operação, isto é, uma seqüência de produtos de matrizes. Desta forma, com o objetivo de eliminar estas limitações, foi proposto um novo algoritmo para o cálculo dos CFE e, inclusive, foi realizada uma comparação entre os algoritmos utilizando a **Notação-O** e uma comparação dos tempos de execução de cada algoritmo considerando-se um conjunto de casos. Este novo algoritmo calcula de forma eficiente e eficaz os CFE, uma vez que não são determinados todos os ciclo individualmente.

CAPÍTULO 7
CONCLUSÕES FINAIS

O objetivo deste trabalho é apresentar a sistematização do método de "deadlock avoidance" proposto por Santos Filho [2000a] através de uma abordagem algorítmica que possibilita a implementação da ferramenta computacional de geração automática do algoritmo de controle para SPFs gerando-se regras adicionais de controle com o objetivo de evitar o "deadlock".

No Capítulo 1 apresentou-se uma visão geral do objeto de estudo deste trabalho, ou seja, SPFs. São sistemas que tem como objetivo suprir as necessidades atuais do mercado de forma eficiente e eficaz, integrando os diversos elementos que compõem o SPF, isto é, equipamentos, recursos computacionais e recursos humanos. A chave para esta integração é o sistema de controle. Apresentou-se também as motivações que determinaram o desenvolvimento deste trabalho e os objetivos pretendidos.

No capítulo 2 apresentou-se uma caracterização dos SPFs, isto é, são sistemas que executam múltiplos processos simultaneamente com intenso compartilhamento de um conjunto finito de recursos. Um aspecto muito importante a ser considerado é que os SPFs pertencem à classe de SEDs e, portanto, possuem um indeterminismo em relação ao tempo.

No capítulo 3 apresentou-se uma caracterização do sistema de controle quanto a forma, tipo e classificação dentro do contexto de SPFs. Além disso, apresentou uma análise do

comportamento dinâmico. Verificou-se que o controle dos SPFs é complexo pois possui dois níveis de indeterminismo: (i) indeterminismo em relação ao tempo e (ii) indeterminismo em relação à seqüência dos eventos. Desta forma, adotou-se neste trabalho um modelo de controle proposto por Santos Filho [2000a; 2001] baseado em um modelo de restrições com dois blocos de controle com semânticas distintas: (i) controle de processos e (ii) controle de recursos.

No capítulo 4 apresentou-se a descrição do problema de "deadlock" em SPFs. De acordo com alocação de recursos, é possível ocorrer o travamento entre os processos que são executados simultaneamente. Nos SPFs é abordado o "deadlock" **em "Part Flow Deadlock"** com "**Single Resource Allocation**", isto é, a cada etapa do processo é requerido apenas um único recurso para a sua execução. Em seguida, foram apresentadas as condições mínimas e necessárias para a ocorrência de "deadlock" e a única condição que pode ser controlada, ou seja, a condição de espera circular. A espera circular é uma cadeia cíclica fechada de processos aguardando a liberação de recursos alocadas por outros processos pertencentes à mesma cadeia cíclica.

Definida a condição a ser controlada, apresentou-se os métodos para abordar o problema de "deadlock" e uma revisão bibliográfica de cada método, vantagens e desvantagens. No presente trabalho, adotou-se o método de "deadlock avoidance", isto é, um método baseado em algoritmos que supervisionam os processos quanto à alocação de recursos realizando em tempo real decisões para evitar o estado de "deadlock" controlando-se a condição de espera circular.

No capítulo 5 apresentou-se a descrição e aplicação do método de "deadlock avoidance" proposto por Santos Filho [2000a] para o controle de SPFs considerando-se a natureza complexa e a abordagem para o problema de "deadlock".

No capítulo 6 foi apresentada a principal contribuição deste trabalho que é a sistematização do método apresentado no capítulo 5, isto é, uma abordagem algorítmica que possibilita a criação de uma ferramenta computacional de geração automática do algoritmo de controle para SPFs baseada em regras de produção ou regras adicionais de controle. Esta ferramenta computacional recebe como dados de entrada a quantidade de recursos existentes em gênero e a quantidade de processos, gerando automaticamente o algoritmo de controle de recursos (matriz que representa o E-MFG e as regras adicionais de controle), uma vez que a matriz E-MFG é uma representação do procedimento de controle.

Neste trabalho, as regras adicionais tem como objetivo evitar o "deadlock", porém é possível inserir outras estratégias de controle como, por exemplo, critérios de otimização da produção, entre outros.

A seguir é proposto um novo algoritmo para a determinação dos CFEs utilizando o grafo GAR (matriz de adjacência). Este algoritmo é uma adaptação do algoritmo de pesquisa por profundidade, utilizado para pesquisa em estruturas tipo árvores. É apresentado também, uma comparação dos tempos de execução dos algoritmo.

1.1 Contribuições deste Trabalho

Os resultados deste trabalho mostram que:

- É possível gerar de forma automática o algoritmo de controle de forma rápida e simples bastando apenas inserir informações dos recursos e o seqüenciamento das atividades dos processos (utilização de recursos);

- Com algumas modificações é possível criar um módulo desta ferramenta computacional que integre uma ferramenta de auxílio ao projeto de sistemas de controle para SPFs.

Estes fatos implicam nas seguintes contribuições fundamentais:

- Sistematização do projeto do controle de recursos de recursos de SPFs;

- Geração automática das regras adicionais de controle para evitar o "deadlock" em SPFs complexos;

- Viabiliza a possibilidade de gerar-se sistemas de controle reativos, uma vez que o processo de geração automática dos algoritmos de controle foi aprimorado.

1.2 TRABALHOS FUTUROS

Pode-se sugerir como continuidade de trabalhos futuros a serem desenvolvidos:

a) Desenvolvimento de novos algoritmos para a geração de regras adicionais de controle para SPFs com duas ou mais instâncias de recursos;

b) Desenvolvimento de um controlador E-MFG dedicado;

c) Um simulador que possibilite a análise do algoritmo gerado, possibilitando combinações entre regras adicionais de controle com diferentes objetivos;

d) Geração automática do modelo do controle de processos

1.3 TRABALHOS PUBLICADOS

"*Uma Proposta de Algoritmo para a Determinação dos Ciclos Fechados de Espera em Sistemas Produtivos Flexíveis*", XIV Congresso Brasileiro de Automática, Natal/R.N., Brasil, 2002.

"*Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes*", V Simpósio Brasileiro de Automação Inteligente, Canelas/R.S., Brasil, 2001.

"*Geração Automática de Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes*", XVI Congresso Brasileiro de Engenharia Mecânica, Uberlândia/M.G., Brasil, 2001.

REFERÊNCIA BIBLIOGRÁFICA

AHO, A.V., HOPEROFT, J.E., ULLMAN, J.D. "*The Design and Analysis of Computer Algorithms*", Addison Wesley Pub, 1974.

BANASZAK, Z.A.; KROGH, B.H.. "*Deadlock Avoidance in Flexible Manufacturing System with Concurrently Competing Process Flows*", IEEE Transactions on Robotics and Automation, Vol.6, pp. 724-734, 1990.

CALINESCU, A.; EFSTATHIOU J.; SIVADASAN, S.; SCHIRN J.; HUACCHO HUATUCO, L. "*Complexity in Manufacturing: An Information Theoretic Approach*", Proceedings of the International Conference on Complex Systems and Complexity in Manufacturing, Warwick University, September 2000.

CALINESCU, A.; BERMEJO, J.; EFSTATHIOU J.; SCHIRN J. "*Applying and Assessing Techniques for Measuring Complexity in Manufacturing*", Proceeding of the 1st European Conference on Intelligent Management Systems in Operations, University of Salford, Greater Manchester, UK, pp. 21-28, 1997.

CARDOSO, J.; VALETTE, R. "*Redes de Petri*", Editora da UFSC, Santa Catarina, Brasil, 1997.

CASSANDRAS, C.G. "*Discrete Event Systems – Modeling and Performance Analysis*", Richard D. Irvin, Inc,. and Aksen Associates, Incorporated Publichers, 1993.

CORMEN, T.H., LEISERSON, C.E., RIVEST, R.L., STEIN, C. "*Algoritmos – Teoria e Prática*", Editora Campus, 2001.

CHO, H., KUMARAN, T.K.; WYSK, R.A. "*Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems*", IEEE Transactions on Robotics and Automation, vol.11, no.3, pp.413-421, 1995.

CHO, H. "*An Intelligent WorkStation Controller for Computer Integrated Manufacturing*", Doctor of Philosophy, Texas A&M University, 1993.

EDMONDS, B. "*What is Complexity? – The Philosophy of Complexity per se With Application to Some Examples in Evolution*", Symposium: The Evolution of Complexity, University of Brussels, Belgium, 1995.

FANTI, M.P.; MAIONE, B.; MASCOLO, S.; TURCHIANO, B. "*Control Policies Conciliating Deadlock Avoidance and Flexibility in FMS Resource Allocation*", INRIA/IEEE Conference on Emerging Technologies and Factory Automation, pp.343-351, Paris, France, 1995.

FANTI, M.P.; MAIONE, B.; MASCOLO, S.; TURCHIANO, B. "*Event-Based Feedback Control for Deadlock Avoidance in Flexible Manufacturing Systems*", IEEE Transactions on Robotics and Automation, vol.13, mp.3, pp.347-363, 1997.

GOMES, L.F.S. "*Redes de Petri Reactivas e Hierárquicas – Integração de Formalismos no Projeto de Sistemas Reactivos de Tempo-Real*", Tese de Doutorado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Lisboa, Portugal, 1997.

GROOVER, M.P. "*Automation, Production Systems, and Computer-Integrated Manufacturing*", Second Edition, Prentice-Hall, 2000.

HO, Y.C.; CAO, X.R. "*Pertubation Analysis of Discrete Event Dynamic Systems*", Kluwer Academic Publishers, 1991.

ISLOOR, S.S.; MARSLAND, T.A. "*The Deadlock Problem: An Overview*", IEEE Compu. Mag., vol.13, no.9, pp.58-78, 1980.

KALPAKJIAN, S.; SCHMID, S.R. "*Manufacturing Engineering and Technology*", 4th Edition, Prentice-Hall, 2000.

KUMARAN, T.K.; CHANG, W.; CHO, H.; WYSK, R.A. "*A Structured Approach to Deadlock Detection, Avoidance and Resolution in Flexible Manufacturing Systems*", Int. J. Prod. Res., vol. 32, no.10, pp.2361-2379, 1994.

KUSIAK, A. "*Parts and Tools Handling Systems*", Modelling and Design of Flexible Manufacturing Systems, Elsevier Science Publishers B.V., Amsterdam, Printed in The Netherlands, 1986.

LAWLEY, M.; REVELIOTIS, S.; FERREIRA, P. "*Design Guidelines for Deadlock Handling Strategies in Flexible Manufacturing Systems*", The International Journal of Flexible manufacturing Systems, vol.9, pp.5-30, 1997.

LOUDON, K. "*Dominando Algoritmos com C*", Editora Ciência Moderna Ltda., Rio de Janeiro, Brasil, 2000.

MIYAGI, P.E.. "*Controle Programável – Fundamentos do Controle de Sistemas a Eventos Discretos*", Editora Edgard Blücher Ltda, São Paulo, Brasil, 1996.

MIYAGI, P.E.; MARUYAMA, N.; SANTOS FILHO, D.J.. "*An Anthropocentric Approach for the design of Production Systems Control*", Proceeding of Mechatronics 2000. 7th Mechatronics Forum International Conference, Atlanta, 2000.

MORAES, C.C.; CASTRUCCI, P.L. "*Engenharia de Automação Industrial*", LTC – Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, Brasil, 2001.

MURATA, T. "*Petri Nets: Properties, Analysis and Applications*", Proceedings of the IEEE, vol. 77, no.4, pp.541-580, 1989.

NAKAMOTO, F.Y.; MIYAGI, P.E.; SANTOS FILHO, D.J. "*Uma Proposta de Algoritmo para a Determinação dos Ciclos Fechados de Espera em Sistemas Produtivos Flexíveis*", XIV Congresso Brasileiro de Automática, 2002.

NAKAMOTO, F.Y.; MARUYAMA, N.; MIYAGI, P.E.; SANTOS FILHO, D.J. "*Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes*", V Simpósio Brasileiro de Automação Inteligente, 2001a.

NAKAMOTO, F.Y.; MARUYAMA, N.; MIYAGI, P.E.; SANTOS FILHO, D.J. "*Geração Automática de Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes*", XVI Congresso Brasileiro de Engenharia Mecânica, 2001b.

NETTO, P.O.B. "*Grafos Teoria Modelos Algoritmos*", 2ª.Edição revista e ampliada, Editora Edgard Blücher Ltda, São Paulo, Brasil, 2001.

OGATA, K. "*Engenharia de Controle Moderno*", 3ª.Edição, Prentice-Hall Inc., LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, Brasil, 2000.

PALAZZO, L.A.M.; CASTILHO, J.M.V. "*Sistemas Complexos e Auto-Organização*", VI Escola Regional de Informática, Curitiba, Blumenau e Pelotas, 1998.

PETERSON, J.L. "*Petri Net Theory and the Modeling of Systems*", Prentice-Hall, 1981.

RAMADGE, P.J.G.; WONHAM, W. "*The Control of Discrete Event Systems*", Proceedings of the IEEE, vol.77, no.1, pp.81-98, January 1989.

REISIG, W. "*A Primer in Petri Nets Desing*", Springer-Verlag, Berlin Heidelberg, 1992.

REISIG, W. "*Petri Nets: An Introduction*", Springer-Verlag, Berlin Heidelberg, 1985.

ROCHA, D. "*Fundamentos Técnicos da Produção*", Makron Books do Brasil Editora Ltda, 1995.

SANTOS FILHO, D.J.; MOTOHASHI MATSUSAKI, C.T.; MARUYAMA, N.; MIYAGI, P.E.. "*Arquitetura de Controle Baseada em Modelos Estruturados Aplicada a Sistemas Produtivos*", XVI Congresso Brasileiro de Engenharia Mecânica, 2001.

SANTOS FILHO, D.J. "*Aspectos do Projeto de Sistemas Produtivos*", Tese de Livre docência, Escola Politécnica da Universidade de São Paulo, 2000a.

SANTOS FILHO, D.J.; MOTOHASHI MATSUSAKI, C.T.; NELLI SILVA, E.C.; MIYAGI, P.E. "*Automatic Generation of Metamodels to Design Production Systems Control*", Proceeding of Mechatronics 2000. 7th Mechatronics Forum International Conference, Atlanta, 2000b.

SANTOS FILHO, D.J. "*Controle de Sistemas Antropocêntricos de Produção Baseado em Redes de Petri Interpretadas*", Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, 1998.

SANTOS FILHO, D.J. "*Proposta do Mark Flow Graph Estendido para a Modelagem e Controle de Sistemas Integrados de Manufatura*", Escola Politécnica da Universidade de São Paulo, Dissertação de Mestrado, 1993.

TANENBAUM, A.S. "*Sistemas Operacionais Modernos*", Prentice-Hall, Inc., LTC – Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, Brasil, 1995.

TENENBAUM, A.M., LANGSAM, Y., AUGENSTEIN, M.J. "*Estruturas de Dados Usando C*", Editora Makron Books do Brasil, São Paulo, Brasil, 1995.

VISWANADHAM, N.; NARAHARI, Y.; JOHNSON, T.L. "*Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturin Systems Using Petri Net Models*", IEEE Transactions on Robotics and Automation, Vol. 6, no.6, pp.713-723, 1990.

WU, N. "*Necessary and Sufficient Conditions for Deadlock-Free Operation in Flexible Manufacturing Systems Using a Colored Petri Net Model*", IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, vol.29, no.2, pp.192-204, 1999.

YOON, H.J.; LEE, D.Y. "*Deadlock-Free Scheduling Method for Track Systems in Semiconductor Fabrication*", Proceedings of the 2000 IEEE International Conference on Systems, Man and Cybernetics, Nashiville, Tennessee, USA, October 8-11, 2000a.

YOON, H.J.; LEE, D.Y. "*Deadlock-Free Scheduling for Automated Manufacturing Cells*", Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision, Singapore, 2000b.

ZIVIANI, N. "*Projeto de Algoritmos com Implementações em Pascal e C*", 4ª Edição, Editora Pioneira, São Paulo, Brasil, 1999.