

Supervisores Distribuídos para Sistemas Flexíveis de Manufatura Utilizando Redes de Petri

Zilma Betania de Sá Ribeiro

Dissertação submetida à Coordenação do Curso de Pós-Graduação em
Informática da Universidade Federal da Paraíba - Campus II como
parte dos requisitos necessários para obtenção do grau de Mestre em
Informática.

Área de Concentração: Redes de Petri

Angelo Perkusich

Orientador

Campina Grande, Paraíba, Brasil

©Zilma Betania de Sá Ribeiro, Agosto de 1999



R484s Ribeiro, Zilma Betania de Sa
 Supervisores distribuidos para sistemas flexiveis de
 manufatura utilizando redes de petri / Zilma Betania de Sa
 Ribeiro. - Campina Grande, 1999.
 85 f.

 Dissertacao (Mestrado em Informatica) - Universidade
 Federal da Paraiba, Centro de Ciencias e Tecnologia.

 1. Redes de Petri 2. Sistemas Flexiveis de Manufatura 3.
 Supervisores Distribuidos 4. Dissertacao I. Perkusich,
 Angelo II. Universidade Federal da Paraiba - Campina Grande
 (PB)

CDU 004.7(043)

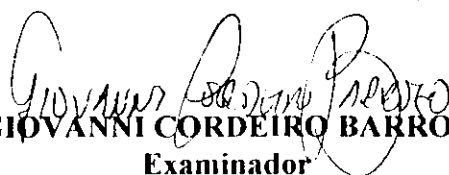
**SUPERVISORES DISTRIBUÍDOS PARA SISTEMAS FLEXÍVEIS DE
MANUFATURA UTILIZANDO REDES DE PETRI**

ZILMA BETÂNIA DE SÁ RIBEIRO

DISSERTAÇÃO APROVADA EM 16.08.1999

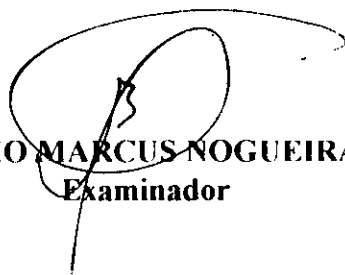


PROF. ANGELO PERKUSICH, D.Sc
Orientador



PROF. GIOVANNI CORDEIRO BARROSO, D.Sc
Examinador

Maria de Fátima Q. V. Turnell
PROF. MARIA DE FÁTIMA QUEIROZ V. TURNELL, Ph.D
Examinadora



PROF. ANTÔNIO MARCUS NOGUEIRA LIMA, Dr.
Examinador

CAMPINA GRANDE – PB

Resumo

Este trabalho tem como objetivo modelar e analisar, utilizando de redes de Petri coloridas, um supervisor distribuído para Sistemas Flexíveis de Manufatura (SFM). Para promover a coordenação de eventos de forma distribuída, atribuímos um supervisor para cada célula do SFM. Cada Supervisor de Célula (SC) executa a coordenação dos eventos internos da célula à qual está associado, aloca recursos de produção e transporte, e trata a ocorrência de faltas tanto nos recursos como nos supervisores.

Um aspecto importante no contexto deste tipo de abordagem é a comunicação entre as entidades descentralizadas para permitir a execução de tarefas cooperativas. Para tanto, propomos um *protocolo de reserva de recurso* para disciplinar uma tal interação entre os SCs. Outros aspectos abordados são o escalonamento de recursos em tempo real através de critérios de custo, e a habilidade de tolerar faltas em recursos.

A principal motivação para esta pesquisa é a crescente tendência de utilizar abordagens distribuídas para a supervisão de SFM, pois estas promovem uma maior escalabilidade, facilidade em tolerar faltas em recursos, além de simplificar a gerência local de recursos.

Abstract

This work aims at the modeling, using Coloured Petri Nets, of a distributed supervisor for Flexible Manufacturing Systems. To promote the coordination of events in a distributed way, a supervisor is assigned to each cell from the FMS. Each Cell Supervisor (CS) performs the internal events coordination from its associated cell, allocate production and transport resources and handle both production resources faults and in the supervisor itself.

A significant feature of such an approach is the of communication among decentralized entities to allow cooperative tasks execution. To do so, we propose a *reservation resource protocol* to discipline the interaction among CSs. Others features considered are the real time resource scheduling through costs criteria, and the ability to tolerate faults.

The main motivation to carry on this research is the increasing tendency to adopt distributed supervisors for FMS, since a greater scalability, fault tolerance and the simplification of local resource coordination can be achieved.

Agradecimentos

Sou grata a todas as pessoas que de alguma forma contribuíram para que eu pudesse concluir este trabalho. Agradeço também à Escola Técnica Federal do Pará, que me forneceu todo apoio, inclusive financeiro, viabilizando minha formação.

Sou grata à minha mãe, meus irmãos e meu pai que sempre me auxiliaram nos momentos que precisei. À família do Sr. José Leite e a Cid, pela paciência e apoio nesta cidade.

Agradeço ao professor Ângelo, que apesar de estar muito atarefado durante o decorrer do curso, tornou possível a existência desta dissertação. Ao professor Einar e ao professor Tomaz, que me forneceram valiosas informações, e sempre me encorajaram e se dispuseram a discutir problemas encontrados durante a pesquisa.

Agradeço também a todos os meus colegas e professores do Curso de Mestrado, que me incentivaram e apoiaram durante momentos difíceis e proporcionaram uma convivência gratificante, tornando o ambiente de trabalho um lugar de harmonia e tranqüilidade.

Conteúdo

1	Introdução	1
1.1	Os Sistemas Flexíveis de Manufatura	2
1.2	Supervisores Distribuídos	4
1.3	Redes de Petri Coloridas	6
1.4	Objetivo do Trabalho	8
1.5	Organização do Trabalho	8
2	Conceitos Básicos	9
2.1	Sistemas Flexíveis de Manufatura	9
2.1.1	Escalonamento de Recursos	11
2.1.2	Tolerância a Faltas em SFM	12
2.2	Redes de Petri	14
2.3	Redes de Petri Coloridas	16
2.4	Redes de Petri Hierárquicas	17
2.5	Redes de Petri Temporizadas	19
2.6	Métodos de Análise	20
3	Supervisores Distribuídos para SFM	22
3.1	O Supervisor da Célula	22
3.2	O Protocolo de Reserva de Recurso	24
3.3	Sistema de Transporte Entre Células	26
3.4	Tolerância a Faltas em Recursos	27
3.5	Conclusão	30

4	Procedimento de Modelagem	31
4.1	Hierarquia de Páginas do Modelo	31
4.2	O Supervisor para as Células	32
4.3	Protocolo de Reserva de Recurso: Módulo de Envio	37
4.4	Geração de Solicitações de Serviço	40
4.5	Protocolo de Reserva de Recurso: Módulo de Recepção	41
4.6	Tratamento das Respostas	43
4.7	Sistema de Transporte Entre Células	46
4.8	Nó de Declaração	47
4.9	Análise do Modelo	49
4.10	Conclusão	51
5	O Supervisor Tolerante a Faltas	54
5.1	Hierarquia de Páginas do Modelo	54
5.2	O Supervisor Tolerante a Faltas	55
5.3	Verificação da Operacionalidade dos Recursos e Reescalonamento de Peças	57
5.4	Geração de Solicitações de Serviço	60
5.5	Protocolo de Reserva de Recurso: Módulo de Recepção	61
5.6	Tratamento de Respostas	62
5.7	Tratamento de Mensagens de Erro	64
5.8	Nó de Declaração	65
5.9	Análise do Modelo	68
5.9.1	Situações 1 e 2	68
5.9.2	Situação 3	69
5.9.3	Situação 4	73
5.10	Conclusão	76
6	Conclusão	77

Lista de Figuras

2.1	Planta de um SFM	11
3.1	O Supervisor da Célula	23
3.2	Situação de falha em todos os recursos de um mesmo tipo	26
3.3	Falha em um dos recursos	28
3.4	Falha em todos os recursos	28
3.5	Falha com peça no <i>buffer</i> da célula	29
3.6	Falha com peça no <i>buffer</i> da máquina	30
4.1	Hierarquia de páginas para o modelo CPN do supervisor	32
4.2	O Supervisor para Célula	33
4.3	Transporte de peças do <i>buffer</i> de saída de uma máquina para o <i>buffer</i> de entrada de outra máquina	37
4.4	Protocolo de Reserva de Recurso: Módulo de Envio	38
4.5	Geração de Solicitações de Serviço	40
4.6	Protocolo de Reserva de Recurso: Módulo de Recepção	42
4.7	Tratamento de Respostas	44
4.8	Transporte de Peças entre Células	46
4.9	Marcação inicial da entrada e saída do sistema	51
4.10	Estados Finais	52
4.11	Marcações correspondentes a informações de estado	53
5.1	Hierarquia de páginas para o modelo do supervisor tolerante a faltas	55
5.2	Supervisor Tolerante a Faltas	56
5.3	Verificação de <i>status</i> de recursos e reescalonamento de peças I	57

5.4	Verificação de <i>status</i> de recurso e reescalonamento de peças II	59
5.5	Geração de Solicitações de Serviço	61
5.6	Protocolo de Reserva de Recurso: Módulo de Recepção	62
5.7	Tratamento de Respostas	63
5.8	Modelo de Tratamento de Mensagens de Erro	64
5.9	Marcação inicial para as situações de falha 1 e 2	69
5.10	Estados finais para as situações de falha 1 e 2	70
5.11	Marcação inicial para a situação de falha 3.a	71
5.12	Estado final para a situação de falha 3.a	72
5.13	Marcação inicial para a situação de falha 3.b	73
5.14	Estado final para a situação de falha 3.b	73
5.15	Marcação inicial para a situação de falha 4.a	74
5.16	Estado final para a situação de falha 4.a	75
5.17	Marcação inicial para a situação de falha 4.b	75
5.18	Estado final para a situação de falha 4.b	76

Capítulo 1

Introdução

Produzir significa “fazer uma coisa nova” originária da palavra *producere* (“conduzir para frente”). Desde o início dos tempos, a história da humanidade está ligada à capacidade de produzir. A contar da idade da pedra, quando o homem polia rochas para fazer utensílios.

Atualmente, os processos de produção são bem mais refinados. Existem sofisticadas tecnologias para transformar materiais brutos em produtos utilizáveis. O conceito de manufatura evoluiu de “fazer com as mãos” (*manu factum*) para um sentido bem mais amplo, que é a conversão de um projeto num produto acabado. Já a produção se refere ao sentido mais restrito do ato físico de fazer o produto.

Bens são produzidos a fim de satisfazer necessidades humanas como vestir, locomover, divertir, etc. Para isso, processos de manufatura são desenvolvidos para agregar valores aos materiais, convertendo matérias-primas em produtos utilizáveis de forma mais eficiente possível.

Um sistema de manufatura é uma coleção ou arranjo de operações e processos utilizados para fabricar um determinado produto ou componente. O sistema de manufatura implementa o planejamento das atividades de produção, a partir de estímulos de entradas e produz saídas (produtos) para os consumidores.

Hoje em dia, a humanidade é bastante heterogênea em relação às atividades de consumo. Existem diferentes pessoas com diferentes gostos e estilos. Além disso, existe uma crescente e dinâmica demanda de produtos na sociedade. Devido a esses fatores, e também a alta competitividade do mercado capitalista, têm sido desenvolvidos sistemas

de produção, como os Sistemas Flexíveis de Manufatura (SFM), que correspondem a essa tendência de mercado.

1.1 Os Sistemas Flexíveis de Manufatura

SFM são uma classe de Sistemas de Produção cuja característica principal é produzir uma variedade produtos de forma automatizada. Um SFM pode ser visto como um complexo estruturado de tecnologias de produção cujos componentes operam em conjunto, com o objetivo de manufaturar uma diversidade de produtos. Exemplos desses componentes são: robôs, máquinas, elementos de transporte, componentes de computador, etc. [11]. A *planta* de um SFM define a forma como os componentes estão dispostos no chão de fábrica e conectados uns aos outros [39], que varia conforme a especificação particular do SFM.

Um SFM pode ser organizado numa hierarquia de cinco níveis de abstração [45; 37; 46]: *planejamento, escalonamento, supervisão, coordenação e comando local*.

- O *nível de planejamento*, é o nível mais alto na hierarquia. Nele implementa-se as atividades de planejamento a longo prazo, considerando toda a fábrica e a demanda estimada. Estabelecendo como a quantidade de produtos necessária será produzida ao longo do tempo;
- No *nível de escalonamento* são implementados mecanismos de decisão, objetivando definir a ordem de utilização dos diferentes recursos de produção, para toda ordem de produção definida no nível de planejamento;
- No *nível de supervisão* mantém-se informações de estado atualizadas da fábrica e implementa-se mecanismos de decisão em tempo real, considerando-se o estado de cada recurso, e o estado das peças sendo processadas;
- A principal função executada no *nível de coordenação* é a gerência de recursos. Este nível pode ser decomposto em módulos especializados na gerência de sub-sistemas como por exemplo, o controle de um robô, um *buffer*, etc;

- No *nível de comando local* a principal função é o controle direto dos dispositivos associados ao processo controlado, tais como, robôs, máquinas, etc.

Neste trabalho, estamos interessados no nível de supervisão do SFM, isto é, no comportamento do supervisor para conduzir a manufatura de produtos.

Um *produto* é manufaturado a partir de matérias-primas, um *subproduto* é um produto não acabado. No contexto deste trabalho, uma *peça* é tanto um produto ou subproduto.

Para ser manufaturada, a matéria-prima precisa passar por uma seqüência de operações que são executadas por *recursos de produção no SFM*. Os recursos de produção são ferramentas que operam sobre a matéria-prima para transformá-la em peças. A seqüência de recursos de produção que a matéria-prima deve visitar para ser manufaturada é denominada *rota de produção* [11]. As rotas de produção podem ser estáticas ou dinâmicas dependendo da forma como os recursos são escalonados, previamente ao processo de manufatura, ou em tempo real.

Células são agrupamentos de recursos de produção segundo algum critério de organização, tais como, funcionalidade, tipo de peça a ser produzida, etc. Para ser produzida, uma peça deve sempre ser transportada para células que possuem o mais recente recurso especificado numa ordem de produção. Os componentes utilizados para o transporte de peças no SFM são denominados de *recursos de transporte* [39].

Para coordenar as operações de todos esses componentes, fazendo a alocação adequada de recursos de produção e transporte a fim de permitir a execução das rotas de produção, é necessária a existência de um sistema supervisor. O sistema supervisor coordena a seqüência de eventos no SFM, através da troca de mensagens com os *sistemas de gerenciamento locais* [2] dos recursos de produção e transporte. O supervisor coordena o processo de manufatura a fim de garantir que as rotas de produção sejam completadas com sucesso, podendo também usar estratégias de escalonamento que melhor utilizem os recursos de produção e transporte com o objetivo de otimizar a manufatura de produtos. Outra atribuição do supervisor é aplicar medidas que garantam a continuidade do serviço mesmo na presença de faltas no SFM.

Como quaisquer elementos do mundo real, os componentes de um SFM estão também sujeitos a falhas, que podem interferir no funcionamento desejado do SFM.

Faltas podem fazer com que as tarefas do sistema não sejam completadas a contento, afetando a qualidade dos produtos finais, ou mesmo impedindo a continuidade do fluxo de produção. Exemplos de faltas podem ser: quebra de ferramenta, desalinhamento, etc. Devido a esses fatores, a possibilidade de tolerar faltas é um atributo importante para um SFM. Para isso, é necessário definir e implementar estratégias e mecanismos apropriados para permitir soluções adequadas para este tipo de situação. Faltas em SFM são tratadas através de réplicas, que podem assumir a função de um componente em caso de falha deste. No caso de faltas em recursos de produção, o supervisor deve redirecionar as rotas que passariam por recursos defeituosos, para as correspondentes réplicas desses recursos.

1.2 Supervisores Distribuídos

O sistema supervisor pode ser centralizado, no qual um sistema central é usado para coordenar os eventos relativos à produção como alocação de recursos, transporte e mecanismos de tolerância a faltas do SFM; ou pode ser distribuído, no qual o controle supervísório é formado por subsistemas independentes e cooperantes que supervisionam cada um, uma parte do SFM. Nesta abordagem, a coordenação de eventos é diluída entre os supervisores descentralizados do sistema, sendo os mecanismos de comunicação essenciais para possibilitar a sincronização de suas atividades.

SFM são sistemas inerentemente distribuídos. Por esse motivo, a aplicação de abordagens descentralizadas têm sido largamente investigadas para projetos de supervisores de SFM. Uma das vantagens é a promoção de uma maior escalabilidade ao sistema, tornando-o mais flexível. Além disso, o controle centralizado pode ficar inviável se a quantidade de componentes para gerenciar se tornar muito grande. Com o gerenciamento distribuído, a coordenação de eventos localmente torna mais simples a tarefa de supervisão do SFM. Outra vantagem é que com a replicação de supervisores, torna-se mais natural a implementação de tolerância a faltas nos supervisores.

Existem porém, algumas desvantagens dos sistemas distribuídos em relação aos centralizados. Os sistemas distribuídos necessitam de um suporte de *software* mais robusto que os centralizados para suportar elementos adicionais que precisam ser gerenciados

em consequência da distribuição. Como por exemplo, na troca de mensagens entre os componentes distribuídos no sistema, pode haver perda ou sobregarga de mensagens na rede de comunicação. Por esse motivo, é necessário que o ambiente distribuído possua um suporte satisfatório para este tipo situação. Outro problema é a replicação de dados. Pode haver problemas se cópias de bases de dados locais não estiverem atualizadas e consistentes.

O ambiente de manufatura é de natureza dinâmica. Não é possível prever antecipadamente o estado do sistema em cada momento do processo de produção. Tarefas podem levar mais ou menos tempo que o esperado. Além disso, eventos não planejados podem acontecer. Alguns recursos podem ficar indisponíveis e outros recursos podem ser adicionados. Por esse motivo, a utilização de escalonamento dinâmico tem sido largamente discutida na literatura.

Em [49] é feito um estudo de comparação entre os conceitos de manufatura distribuída emergentes, como a Manufatura Biônica, que traça um paralelo com sistemas biológicos, capturando determinadas propriedades desses sistemas, como a composição por células e o comportamento autônomo, para utilizar em sistemas de manufatura. E Sistemas de Manufatura Holônicos, compostos por unidades denominadas de *holons*, que são entidades que exibem características de comportamento autônomo e cooperativo. Esses sistemas podem ser vistos como uma espécie de sistema distribuído composto por agentes de manufatura inteligentes [17].

Em [42], um projeto de manufatura baseado em agentes utilizando roteamento dinâmico é apresentado. Em [1], uma abordagem de descentralização do gerenciamento de dados é discutida para distribuir o comando operacional de um SFM e obter módulos de controle independentes. Em [33], um sistema de controle distribuído para manipulação de material foi desenvolvido utilizando pequenas células que trabalham de modo cooperativo, através de trocas de mensagens, para movimentação de peças sobre suas superfícies.

Barros [2; 3; 7; 5; 4; 6; 8], formalizou a especificação para SFM e usou redes de Petri coloridas para modelar o supervisor de um SFM. O supervisor modelado utiliza um controle centralizado para coordenar as operações do sistema, e um esquema para redirecionamento dinâmico de rotas em caso de falhas em recursos.

O supervisor que apresentamos neste trabalho está dentro do contexto da especificação definida em [2], na qual um SFM é composto por produtos, *buffers*, células e recursos de produção e transporte. A definição formal para SFM conforme introduzida em [2] é mostrada no Capítulo 2.

1.3 Redes de Petri Coloridas

Uma das ferramentas utilizadas para estudo e modelagem de SFM são as Redes de Petri, cujas características gráficas, com fundamentação matemática consolidada, as tornam adequadas para o estudo e a modelagem desses sistemas. Há vários anos as redes de Petri têm sido usadas para modelagem de SFM, protocolos de comunicação e outros sistemas a eventos discretos que exibem características como paralelismo e concorrência, entre outras [25; 11; 12; 23].

Uma *rede de Petri Lugar/Transição* (PT-Net), do inglês *Place/Transition Nets*, é um grafo bipartido, direcionado e com pesos [35]. O termo bipartido implica que o grafo é composto de dois tipos de nós, denominados de *lugares* e *transições*, e que os *arcos* nunca ligam dois nós do mesmo tipo. O termo direcionado é usado para denotar que os arcos têm origem e destino especificados [18]. Um arco é rotulado com pesos (inteiros positivos), que podem ser interpretados como replicações de arcos. Assim, um arco com peso k denota a existência de um conjunto de k arcos paralelos entre os mesmos nós.

Associam-se marcas, denominadas *fichas*, às estruturas da PT-Net para expressar estados de um sistema. As fichas podem ser associadas apenas aos lugares da rede, e cada lugar pode ser associado a qualquer número inteiro não negativo de fichas. O conjunto total de fichas nos lugares é denominado de *marcação*.

Apesar das vantagens mencionadas no início desta seção, para sistemas grandes os modelos produzidos em PT-Nets podem se tornar muito extensos e difíceis de visualizar. Por esse motivo, foram desenvolvidas as *redes de Petri de Alto Nível*, que têm um poder de expressão maior que as redes de Petri clássicas, permitindo uma descrição mais compacta dos modelos.

As *redes de Petri Coloridas* (CP-Net) [25; 23], do inglês *Coloured Petri Nets*, são um

tipo de redes de Petri de Alto Nível para a especificação, projeto, simulação, validação e implementação de sistemas a eventos discretos.

Uma CP-Net é composta essencialmente por uma estrutura, um conjunto de inscrições e um conjunto de declarações. Como as PT-Nets, as CP-Nets são também um grafo direcionado e bipartido. Entretanto, ao invés de pesos inteiros, aos arcos são associadas inscrições, que determinam quantas e quais fichas devem ser removidas ou adicionadas aos lugares associados na ocorrência de uma transição [26; 23]. Inscrições, denominadas guardas, podem ser também associadas às transições. Guardas restringem a ocorrência de transições a determinadas condições. O estado inicial de uma CP-Net também é determinado por inscrições associadas aos lugares [26; 23].

Por razões históricas, em CP-Nets usa-se a expressão *conjunto de cores (colour set)* em substituição a tipos de dados e, por conseqüência, cada valor é denominado cor (*colour*), que pode ser de um tipo arbitrário de dados (inteiro, real, lista, etc.). Maiores detalhes sobre este assunto podem ser encontrados no Capítulo 2.

Do ponto de vista da modelagem e análise de sistemas a eventos discretos, as redes de Petri apresentam as seguintes vantagens [50; 44]:

1. Facilidade para modelar determinadas características dos sistemas a eventos discretos, tais como: concorrência, sincronismo e assincronismo, conflito, exclusão mútua, relações de precedência, não determinismo e bloqueio;
2. Permitem a utilização de diferentes abordagens para a construção de um dado modelo, como refinamento (top-down) e composição modular (bottom-up);
3. Possibilidade de analisar e verificar propriedades indesejáveis para os sistemas, como bloqueio, e desejáveis, como progressão;
4. Simulação de eventos discretos diretamente a partir do modelo;
5. Informação do estado atual do sistema que permite monitoração em tempo real.

1.4 Objetivo do Trabalho

O objetivo deste trabalho é modelar e analisar, utilizando redes de Petri coloridas, um supervisor distribuído tolerante a faltas para SFM. Para promover a coordenação de eventos de forma distribuída, atribuímos um supervisor para cada célula do SFM. Dessa maneira, ao invés de um controle centralizado, temos vários Supervisores de Célula (SC) constituindo subsistemas de gerenciamento local, que interagem de modo cooperativo para promover a produção. Cada SC executa a coordenação dos eventos internos de uma célula à qual está associado, aloca recursos de produção e transporte de peças, e trata a ocorrência de faltas em recursos.

Como dito anteriormente, uma peça deve visitar recursos especificados numa ordem de produção para poder ser produzida. Para isso, a peça deve sempre ser enviada para células que contenham o mais recente recurso especificado nessa ordem de produção. Por esse motivo, é necessário que os SCs possam interagir para disciplinar o envio de peças de uma célula para outra. Com esse objetivo, propomos um *protocolo de reserva de recurso* para sincronizar as atividades entre esses SCs, e um sistema de transporte baseado em AGVs. Em caso de haver mais de um recurso do mesmo tipo no sistema, critérios de escalonamento são utilizados para aplicar políticas de escolha de destino de peças em tempo real. O tratamento de faltas é abordado através de redundância, executando o reescalonamento de peças para recursos replicados no caso de faltas em recursos de produção, e a geração de mensagens de erro para o ambiente do sistema. Alguns dos resultados apresentados nesta dissertação podem ser encontrados em [41; 40].

1.5 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: O Capítulo 2 apresenta os conceitos básicos relacionados a redes de Petri e SFM. O Capítulo 3 apresenta a solução para a distribuição do controle supervísório para SFM. O Capítulo 4 apresenta o procedimento de modelagem em redes de Petri para o supervisor. O Capítulo 5 apresenta as mudanças necessárias ao modelo para prover tolerância a faltas em recursos de produção. E finalmente, no Capítulo 6, apresentamos a conclusão do trabalho.

Capítulo 2

Conceitos Básicos

Neste Capítulo apresentamos os conjuntos de idéias nos quais foram embasados os modelos que são apresentados nos capítulos seguintes. O principal objetivo é expor os conceitos e definições relacionados a SFM e redes de Petri necessários à compreensão do restante deste trabalho. Na Seção 2.1 apresentamos os conceitos básicos relacionados a SFM. Nas Seções 2.2, 2.3, 2.4 e 2.5 apresentamos os conceitos básicos relacionados a redes de Petri. Finalmente, na Seção 2.6 apresentamos uma introdução informal aos métodos de análise para redes de Petri.

2.1 Sistemas Flexíveis de Manufatura

Como dito no Capítulo 1, os SFM são uma classe de Sistemas de Produção cuja característica principal é produzir uma variedade de produtos de forma automatizada.

Um *produto* é manufaturado a partir de matérias-primas, um *subproduto* é um produto não acabado. Sendo esses termos relativos, por exemplo, uma fábrica pode prover subprodutos para outra, de forma que o produto acabado de uma pode ser matéria-prima para a outra [45].

Para ser manufaturada, a matéria-prima precisa passar por uma seqüência de operações que são executadas por recursos de produção do SFM. Os recursos de produção são acoplados a máquinas que podem ser dispostas em unidades denominadas células. No contexto deste trabalho, uma célula é uma unidade estrutural básica composta de: um *buffer* de entrada, um *buffer* de saída, um ou mais recursos de produção e recur-

mentos de transporte, como robôs, para transportar subprodutos de um lugar para outro dentro da célula.

Os *recursos de transporte* são os componentes utilizados para o transporte físico de peças no SFM [39]. AGVs (*Automated Guided Vehicles*) são veículos auto guiados para transportar peças de uma célula para outra. AGVs podem transportar produtos de qualquer ponto para qualquer outro ponto do SFM [39]. Por esse motivo, são considerados recursos de transporte bastante flexíveis.

Os *Gerenciadores Locais de Recursos* (GLR) são componentes que atuam diretamente sobre os recursos locais de produção e transporte através de mensagens recebidas pelo supervisor [2].

Para sincronizar as atividades de todos esses elementos num SFM, é necessária a existência de um sistema supervisor. O sistema supervisor coordena a seqüência de eventos no SFM, através da troca de mensagens com os sistemas de gerenciamento locais dos recursos de produção e transporte. O supervisor coordena o processo de manufatura a fim de garantir que as rotas de produção sejam completadas com sucesso, podendo também usar estratégias de escalonamento que melhor utilizem os recursos de produção e transporte com o objetivo de otimizar a manufatura de produtos.

Na Figura 2.1 mostramos a planta de um SFM formada por um *buffer* de entrada e um *buffer* de saída, três AGVs, quatro células compostas cada uma de dois recursos de produção e um robô. Cada célula e cada recurso de produção possuem um *buffer* de entrada e um *buffer* de saída. Além disso, duas rotas de produção são denotadas pelas linhas r_1 e r_2 , que atravessam a planta desde a entrada até a saída.

Conforme introduzido em [2], um SFM é a 6-upla (P, B, Y, C, M, R) , onde:

1. $Pr = \{pr_1, pr_2, \dots, pr_i, \dots, pr_m\}$ é um conjunto finito e não vazio de produtos,
2. $B = \{b_1, b_2, \dots, b_i, \dots, b_u\}$ é um conjunto finito e não vazio de *buffers*,
3. $Y = Y_p \cup Y_t = \{y_1, y_2, \dots, y_i, \dots, y_n\}$ é um conjunto finito e não vazio de recursos.
Se $y_i \in Y_t$ então y_i é um recurso de transporte, caso contrário, se $y_i \in Y_p$ então y_i é um recurso de produção, onde $Y_p \cap Y_t = \emptyset$.
4. $C = \{c_1, c_2, \dots, c_i, \dots, c_n\}$ é um conjunto finito e não vazio de células,

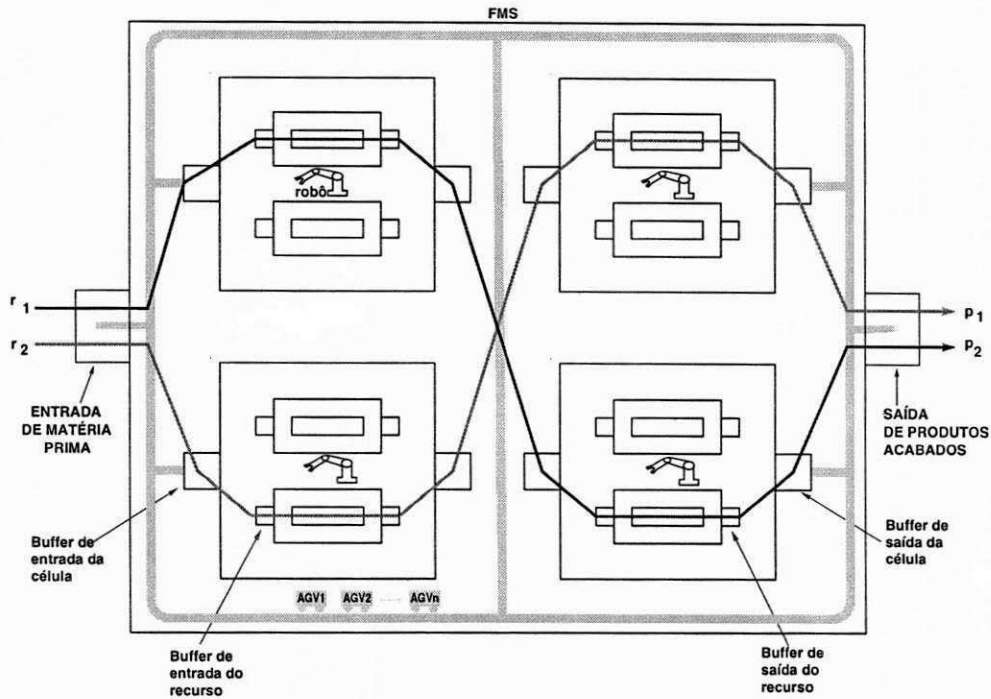


Figura 2.1: Planta de um SFM

5. $M = \{m_1, m_2, \dots, m_i, \dots, m_k\}$ é um conjunto finito e não vazio de máquinas,
6. $R = \{r_1, r_2, \dots, r_i, \dots, r_m\}$ é um conjunto finito e não vazio de rotas de produção.

2.1.1 Escalonamento de Recursos

Escalonamento de recursos é um dos aspectos mais importantes do planejamento e operação de um SFM. Diz respeito a atribuir operações a recursos à fim de executar um conjunto finito de tarefas [11; 13; 9]. Geralmente, o objetivo é otimizar medidas de desempenho no sistema [22; 32; 36], como por exemplo [11; 13]:

- Tempo total requerido para completar todas as tarefas (*Makespan*);
- Prazos de entrega;
- Tempo médio na fábrica (tempo médio de fluxo);
- Tempo de máquina ocioso;
- Percentagem de tarefas atrasadas;

- Tempo médio em fila.

O problema de escalonamento é bastante complexo devido ao grande número de alternativas que podem ser consideradas num SFM [28; 13], sendo que algumas medidas de desempenho podem ser conflitantes, como por exemplo: prazos de entrega são mais facilmente atingidos se a utilização de máquinas for baixa. Dessa forma, os critérios de escalonamento devem ser previamente planejados de acordo com os requisitos de prioridade do SFM [13].

O escalonamento pode ser estático, dinâmico ou uma combinação de ambos [13]. É estático quando o conjunto de recursos a serem utilizados no processo de produção são escalonados no início do processo. Em contraste, na abordagem dinâmica, decisões de escalonamento de recursos são tomadas em tempo real, a partir de variáveis de estado durante o processo de produção.

Critérios de custos são os fatores de otimização que podem ser usados como base para decidir, a partir de valores de determinadas variáveis do sistema, qual a “melhor” política de escalonamento que satisfaz os requisitos de desempenho desejados. Por exemplo, se o objetivo for minimizar o tempo requerido para completar as tarefas, pode-se considerar uma combinação de valores tais como: menor fila, menor tempo de processamento, menor distância, etc. [48; 10].

2.1.2 Tolerância a Faltas em SFM

Como quaisquer elementos do mundo real, SFM são suscetíveis a falhas. Componentes podem falhar resultando em erros que interfiram no funcionamento correto do sistema. Por isso, torna-se relevante a possibilidade de tolerar faltas através de mecanismos que permitam a continuidade do processo de manufatura mesmo na ocorrência de mau funcionamento de alguns recursos. Essas características introduzem um alto grau de complexidade na alocação e gerenciamento de recursos desses sistemas. Por esse motivo, é importante deixar claro alguns conceitos relacionados a esse assunto.

Erro é um estado de um sistema que corresponde à diferença entre o comportamento esperado e o produzido, podendo ser observado e avaliado [21]. Um erro corresponde à manifestação de uma falta no sistema. *Falta* é associada com a noção de defeito. Um

sistema em falta é um sistema com defeito. Uma falta é uma fonte que tem o potencial de gerar erros [21]. Uma *falha* é o efeito de um erro. A falha ocorre quando o sistema não se comporta conforme as especificações, sendo deduzida pela detecção de erros num estado do sistema [21]. A detecção de erros em algum estado do sistema implica na ocorrência de faltas, entretanto, a ocorrência de faltas não implica necessariamente na manifestação de erros. As faltas podem ser classificadas como transientes ou permanentes. Faltas transientes são faltas com duração limitada, isto é, com duração de vida menor que os requisitos de duração do sistema. Faltas permanentes, são aquelas que uma vez manifestadas através da falha de um componente, implicam em que este não irá se comportar corretamente de novo [21], ou seja, sua duração de vida coincide com a duração de vida do sistema [30].

Um sistema é dito tolerante a faltas se puder mascarar a presença de faltas através de redundância. O objetivo da tolerância a faltas é evitar que o sistema falhe mesmo na presença de faltas [21], isto é, que o comportamento do sistema seja consistente com as especificações dadas mesmo na ocorrência de defeitos em alguns componentes. Os mecanismos de tolerância a faltas podem ser a nível de *hardware* ou *software* [21]. Faltas a nível de *hardware* são classificadas como permanentes e tratadas através de réplicas, que podem assumir a função de um componente em caso de falha deste.

Em um SFM, componentes podem vir a falhar, afetando a qualidade dos produtos finais ou mesmo impedindo a continuidade do fluxo de produção. Recursos de produção estão sujeitos a falhas (quebra da ferramenta, desalinhamento, etc.), fazendo com que as tarefas do sistema não sejam completadas a contento. O supervisor deve implementar medidas que garantam a continuidade do serviço mesmo na presença de recursos de produção defeituosos no caminho de produtos a serem manufaturados. Para isso, é necessário que existam mecanismos apropriados para permitir soluções adequadas para este tipo de situação.

Um SFM pode ter réplicas de recursos de produção em diferentes máquinas, e pode ter também ferramentas duplicadas como reserva para troca em caso de quebra. As réplicas de recursos podem também ser usadas para agilizar o processo de manufatura, diminuindo-se o tempo de espera de peças para obter um serviço. Para tolerar faltas em recursos de produção, o supervisor precisa redirecionar as rotas que passariam por

recursos em falha, para as correspondentes réplicas desses recursos.

A replicação de supervisores pode também ser usada para tolerar faltas nos próprios supervisores. Neste caso, o sistema pode continuar funcionando se houver outro supervisor para assumir a funcionalidade daquele que falhou.

2.2 Redes de Petri

Redes de Petri são um conjunto de formalismos, com características gráficas, utilizadas como ferramenta para o estudo e a modelagem de SFM, e outros de sistemas a eventos discretos [25; 11; 12; 23].

Uma rede de Petri Lugar/Transição (PT-Net), do inglês *Place Transition Nets*, é um grafo bipartido, direcionado e com pesos [35]. O termo bipartido implica que o grafo é composto de dois tipos de nós, denominados de *lugares* e *transições*, e que os *arcos* nunca ligam dois nós do mesmo tipo. O termo direcionado é usado para denotar que os arcos têm origem e destino especificados [18]. Um arco é rotulado com pesos (inteiros positivos), que podem ser interpretados como replicações de arcos. Assim, um arco com peso k denota a existência de um multiconjunto de k arcos paralelos entre os mesmos nós.

A estrutura descrita até agora, estabelece apenas a natureza sintática de uma PT-Net. Para expressar aspectos dinâmicos de um sistema, é necessário que as redes de Petri possam exprimir estados desse sistema e suas evoluções. Para isso, associam-se marcas, denominadas *fichas*, que denotam o estado em que se encontra o sistema. As fichas podem ser associadas apenas aos lugares da rede, e cada lugar pode ser associado a qualquer número inteiro não negativo de fichas. O conjunto total de fichas nos lugares é denominado de *marcação*.

Uma marcação é um vetor M , com m componentes, onde m é o número de lugares na rede. A p -ésima componente de M , denotada por $M(p)$, é o número de fichas no lugar p . Na modelagem, usando o conceito de condições e eventos, lugares representam condições, e transições representam eventos. Na representação gráfica, lugares são representados por círculos, e transições por retângulos ou barras. Uma transição (evento) tem um certo número de lugares de *entrada* e lugares de *saída* representando

as pré-condições e pós-condições dos eventos, respectivamente. A presença de uma ficha num lugar indica que a condição associada àquele lugar é verdadeira. Numa outra interpretação, k fichas em um lugar indicam que k recursos ou itens de dados estão disponíveis [35].

Uma PT-Net é a união de uma estrutura de PT-Net (um grafo) e uma marcação inicial (um estado). Formalmente [35], uma PT-Net é uma 5-upla, $PN = (P, T, F, W, M_0)$, onde:

1. $P = \{p_1, p_2, \dots, p_m\}$ é um conjunto finito de lugares,
2. $T = \{t_1, t_2, \dots, t_n\}$ é um conjunto finito de transições,
3. $F \subseteq (P \times T) \cup (T \times P)$ é um multiconjunto finito de arcos,
4. $W : F \rightarrow \mathbb{N}^*$ é uma função peso,
5. $M_0 : P \rightarrow \mathbb{N}$ é a marcação inicial
6. $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$

Para simular o comportamento dinâmico de um sistema e sua evolução, o estado ou marcação numa PT-Net é alterado de acordo com a seguinte regra de transição:

1. Um transição é dita habilitada se cada lugar de entrada p de t é marcado com pelo menos $w(p, t)$ fichas, onde $w(p, t)$ é o peso do arco direcionado de p a t .
2. Uma transição habilitada pode ou não disparar, dependendo se ou não o evento ocorreu.
3. O disparo de uma transição t remove $w(p, t)$ fichas de cada lugar de entrada p de t , e adiciona $w(t, p)$ fichas a cada lugar de saída p de t , onde $w(t, p)$ é o peso do arco direcionado de t a p .

Uma transição sem qualquer lugar de entrada é denominada de transição fonte, e uma transição sem qualquer lugar de saída é denominada de transição sorvedouro. Uma transição fonte é incondicionalmente habilitada, e o disparo de uma transição sorvedouro consome fichas mas não produz nenhuma. O par formado por um lugar

p e uma transição t é um auto-laço se p é simultaneamente lugar de entrada e saída de t . Uma PT-Net é dita *pura* se não possuir nenhum auto-laço. Uma PT-Net é dita *ordinária* se o peso de todos os seus arcos é igual a 1.

2.3 Redes de Petri Coloridas

Sistemas do mundo real normalmente contêm partes que são similares, mas não idênticas. Usando PT-Nets, essas partes devem ser representadas por subredes separadas com estruturas quase idênticas. Isso significa que a rede total pode ficar muito grande. Além disso, pode ser difícil observar as similaridades e diferenças entre as redes individuais que representam as partes similares. Por isso, para descrever sistemas complexos, torna-se necessário utilizar redes com maior poder de descrição, possibilitando então descrever sistemas complexos de uma forma mais viável. As redes de Petri coloridas (CP-Nets) são uma classe de redes de Petri de alto nível que representam um grande avanço nesse sentido. Com a utilização dessas redes, pode-se conseguir uma representação mais compacta de qualquer sistema que pode ser modelado com PT-Nets. [25; 23].

Uma CP-Net é composta essencialmente por uma estrutura, um conjunto de inscrições e um conjunto de declarações. Como as PT-Nets, as CP-Nets são também um grafo direcionado e bipartido. Entretanto, ao invés de pesos inteiros, aos arcos são associadas inscrições que determinam dinamicamente quantas e quais fichas devem ser removidas ou adicionadas aos lugares associados, na ocorrência de uma transição. Inscrições, denominadas guardas, podem ser também associadas às transições. Guardas restringem a ocorrência de transições a determinadas condições. O estado inicial de uma CP-Net também é determinado por inscrições associadas aos lugares. Cada inscrição é, em geral, uma expressão construída a partir de constantes, variáveis e operadores previamente definidos [18]. Uma CP-Net também possui um conjunto de declarações para indicar a natureza dos elementos citados nas diversas inscrições, à semelhança de uma área de declarações de uma linguagem de programação qualquer.

As inscrições e declarações de uma CP-Net podem, *a priori*, ser escritas em praticamente qualquer linguagem com sintaxe e semântica bem definidas. Em geral, CP-Nets

vêm sendo utilizadas em associação com uma linguagem denominada CPN-ML, derivada da linguagem funcional *Standard ML*, cuja sintaxe é bastante semelhante à usada por linguagens de programação convencionais [18].

Por razões históricas, em CP-Nets usa-se a expressão *conjunto de cores* (*colour set*) em substituição a tipos de dados e, por conseqüência, cada valor é denominado cor (*colour*), que pode ser de um tipo arbitrário de dados (inteiro, real, lista, etc.). Desta forma, cada lugar na estrutura interna é associado a um conjunto de cores, que indica o tipo de fichas que o lugar pode conter, i.e., para um dado lugar, todas as fichas devem ter cores que pertencem a um mesmo tipo. A linguagem CPN-ML dispõe de mecanismos que permitem a definição de conjuntos de cores relativamente complexos.

Variáveis de Transição referem-se ao conjunto de variáveis presentes nas inscrições dos arcos e na guarda da referida transição. Uma *ligação* (*binding*) pode ser vista como a substituição de cada variável da transição por uma cor (valor). É requerido, entretanto, que as cores pertençam aos conjuntos de cores apropriados e que impliquem na avaliação da guarda como verdadeira [18].

Em cada marcação, a ocorrência de uma transição sob uma determinada ligação é dita habilitada se todos os seus lugares de entrada tiverem fichas suficientes para satisfazer às expressões dos arcos. Cada expressão deve ser devidamente avaliada segundo as substituições determinadas pela ligação, a fim de determinar quantas e quais fichas são requeridas nos lugares de entrada. Caso a transição ocorra, então são retiradas fichas dos lugares de entrada e depositadas novas fichas nos lugares de saída. A quantidade de fichas é determinada também pela avaliação das expressões dos arcos segundo as substituições implicadas pela ligação [18]. Para definição formal de CP-Nets, ver [23].

2.4 Redes de Petri Hierárquicas

A idéia básica das redes de Petri hierárquicas é possibilitar a construção de um grande modelo através da combinação de um conjunto de pequenas CP-Nets denominadas *páginas*, de forma análoga à construção de um programa a partir de um conjunto de módulos e subrotinas [23].

O poder de modelagem teórico de uma PT-Net, de uma CP-Net e de uma CP-Net hierárquica são equivalentes. É sempre possível traduzir uma CP-Net hierárquica para uma CP-Net não hierárquica, que por sua vez pode ser traduzida para uma PT-Net. Contudo, de um ponto de vista prático, essas três classes de redes possuem diferentes propriedades. Em termos de linguagem de programação, podemos comparar PT-Nets com as linguagens de máquina, CP-Nets com introdução de elementos de dados estruturados na programação e CP-Nets Hierárquicas com o uso de módulos e rotinas.

Um dos grandes problemas enfrentados por desenvolvedores de sistemas atualmente, é lidar com muitos detalhes ao mesmo tempo. As CP-Nets Hierárquicas podem ser vistas como um mecanismo de abstração que permite ao modelador lidar com uma parte selecionada de um modelo sem se distrair com detalhes de baixo nível das partes restantes.

As redes de Petri Hierárquicas são implementadas utilizando-se o conceito de *lugares de fusão e transições de substituição*. Lugares de fusão são estruturas que permitem especificar um conjunto de lugares como funcionalmente um único lugar, isto é, se uma ficha é removida ou adicionada de um dos lugares, uma ficha idêntica é adicionada ou removida de todos os outros lugares pertencentes ao conjunto. Um conjunto de lugares de fusão é denominado de conjunto de fusão (*fusion set*).

Uma transição de substituição pode ser vista como uma transição de mais alto nível que se relaciona a uma CP-Net mais complexa que fornece maiores detalhes das atividades representadas pela transição de substituição. A página que contém a transição de substituição é denominada de *superpágina* da CP-Net mais detalhada correspondente, que por sua vez é denominada de *subpágina*. Cada transição de substituição é denominada de *supernó* da subpágina correspondente.

Uma transição de substituição se relaciona com sua subpágina através da utilização de um tipo de conjunto de fusão de dois membros denominados portas e *sockets*. Estas estruturas descrevem a interface entre a transição de substituição e a subpágina. *Sockets* são atribuídos aos lugares conectados à transição de substituição, e portas são associadas a determinados lugares na subpágina tal que um par *socket*/porta formam um conjunto de fusão. Dessa forma, quando uma ficha é depositada num *socket*, ela

aparece também na porta associada àquele *socket*, permitindo assim a conexão entre a superpágina e a subpágina. Cada *socket* pode ser associado a uma ou mais portas, e uma porta pode ser associada somente a um *socket*.

É sempre possível traduzir uma rede de Petri hierárquica para sua correspondente não hierárquica. Para isso, basta substituir cada transição de substituição e arcos conectados, por sua respectiva subpágina "colando" cada *socket* com sua respectiva porta. Para definição formal de CP-Nets Hierárquicas ver [23].

2.5 Redes de Petri Temporizadas

Nas classes de redes de Petri descritas até agora, somente aspectos estruturais e comportamentais podem ser verificados nos sistemas modelados. Nenhuma suposição é feita em relação à duração das atividades desses sistemas. Entretanto, isso nem sempre é suficiente para o estudo de todas as propriedades que se deseja analisar no sistema. Sistemas do mundo real não são atemporais, aspectos de tempo como avaliação de desempenho, avaliação de custos e/ou critérios de prioridade baseados no tempo podem ser relevantes para o problema em questão. Por isso, foram propostas extensões temporais de redes de Petri para atender às necessidades de modeladores que precisam lidar com aspectos de tempo de diversos tipos de sistemas.

Diferentes abordagens foram propostas ao longo dos anos no sentido de promover modelagens com representação de tempo [24; 16]. Em relação ao *tipo*, as especificações de restrições de tempo podem ser determinísticas, intervalos ou estocásticas. Em relação a *localização*, restrições de tempo podem ser associadas a lugares, transições, arcos ou fichas [16; 34].

Na abordagem de tempo associada a fichas [24], cada ficha carrega, além dos valores de cor associados, um valor de tempo denominado de *rótulo de tempo* (*time stamp*). Intuitivamente, o rótulo de tempo descreve o tempo mais cedo que a ficha pode ser usada. Para isso, introduz-se também o conceito de *relógio global*, cujos valores representam o tempo do sistema modelado (tempo de simulação), que pode ser representado tanto por inteiros (tempo discreto) quanto por reais (tempo contínuo).

Numa CP-Net temporizada uma transição é dita *cor habilitada* quando satisfaz

todos os requisitos da regra de habilitação para CP-Nets não temporizadas (quando todos as fichas requeridas nos lugares de entrada e guardas avaliarem verdadeiras). Contudo, para ser habilitada, a transição também precisa estar *pronta*, isto é, os rótulos de tempo das fichas a serem removidas devem ser menores ou iguais ao tempo de simulação corrente [24].

A execução de uma CP-Net temporizada é orientada pelo tempo, o sistema se mantém num dado tempo de simulação enquanto houver transições habilitadas para disparar. Quando não houver mais nenhuma transição para disparar no tempo de simulação corrente, o sistema avança o relógio para o próximo tempo de simulação no qual uma transição poderá ser disparada. Cada marcação existe num intervalo fechado do tempo de simulação (que pode ser um ponto, i.e., um único momento). A ocorrência de transições é instantânea [24].

Para modelar o fato de uma operação levar r unidades de tempo, associa-se a uma correspondente transição t uma *região de tempo*, que permite a criação de rótulos de tempo para suas fichas de saída, que são r vezes unidades de tempo maiores que o valor do relógio no qual t ocorre. Isso significa que as fichas produzidas por t são indisponíveis por r unidades de tempo. Para definição formal de CP-Nets Temporizadas ver [24].

2.6 Métodos de Análise

Uma das grandes vantagens do emprego de redes de Petri é a possibilidade de fazer a verificação das propriedades dos modelos construídos através de métodos formais de análise. As propriedades que dependem da marcação inicial do modelo são denominadas de *propriedades comportamentais* ou *dinâmicas* e as que não dependem da marcação inicial são denominadas *propriedades estruturais* [24; 23].

Os métodos de análise são baseados na construção do *grafo de ocorrência*, para verificação das propriedades comportamentais do sistema, no emprego de *equações de estado*, para verificação de propriedades estruturais do sistema, ou nas técnicas de *redução e decomposição de redes*, para reduzir o modelo facilitando sua análise [24; 23].

O grafo de ocorrência permite investigar propriedades dinâmicas em redes de Petri.

A idéia básica é construir um grafo que tenha um nó para cada marcação alcançável e um arco para cada elemento de ligação ocorrente na rede. O conteúdo da marcação é descrito numa inscrição de texto do nó [25; 24].

Um grafo de ocorrência pode vir a ser muito grande mesmo para pequenas redes. Na prática, é comum lidar com uma CP-Net cujo grafo de ocorrência tenha mais de 100000 nós. Conseqüentemente, sua construção e investigação através de métodos manuais tornam-se bastante suscetíveis a erros. Entretanto, a construção de grafos de ocorrência e a verificação associada das propriedades dinâmicas podem ser completamente automatizados provendo um método mais direto de analisar uma dada rede de Petri [25; 23].

Algumas das propriedades comportamentais, que podem ser verificadas através do grafo de ocorrência, consideradas normalmente para análise dos sistemas de manufatura são [51]:

1. *Alcançabilidade*: Uma marcação M_i é dita alcançável a partir da marcação inicial M_0 se existir uma seqüência de disparos de transições σ que transforma M_0 em M_i , ou seja, $M_0[\sigma]M_i$. Denota-se por $R(M_0)$, ao conjunto de todos os possíveis estados alcançados a partir de M_0 e por $L(M_0)$ ao conjunto de todas as possíveis seqüências de disparos de transições. Então o problema de determinar a existência de um estado específico M_i , consiste em determinar se $M_i \in R(M_0)$ [20].
2. *Conservação*: Uma rede é dita conservativa se existir um vetor $w = (w_1, w_2, \dots, w_m)$, onde, m é o número de lugares da rede e $w(p) > 0$ para cada $p \in P$, tal que a soma ponderada de fichas contidas nos lugares é uma constante para qualquer marcação alcançável a partir de M_0 .
3. *Vivacidade* Esta propriedade é empregada para verificar a existência ou não de bloqueios (*deadlocks*) no sistema [14; 15; 19]. Uma transição $t \in T$ é dita viva se para qualquer marcação $M_i \in R(M_0)$ existir, a partir de M_i , uma seqüência de disparo σ contendo t . Uma rede de Petri é viva se todas as suas transições são vivas.

Capítulo 3

Supervisores Distribuídos para SFM

Neste capítulo, objetivamos expor uma solução para supervisores distribuídos em um SFM através de um esquema de cooperação entre vários subsistemas independentes. Para isso, levamos em conta aspectos de comunicação, transporte, escalonamento, tolerância a faltas e outros elementos relevantes para permitir uma abordagem coerente.

A aplicação de abordagens descentralizadas tem sido cada vez mais utilizadas em projetos de supervisores para SFM devido à natureza distribuída desses sistemas. Como dito no Capítulo 1, uma das vantagens é a promoção de uma maior escalabilidade ao sistema, tornando-o mais flexível. Além disso, o controle centralizado pode ficar inviável se a quantidade de elementos para gerenciar se tornar muito grande. Com o gerenciamento distribuído, a coordenação de eventos de forma local simplifica a descrição do supervisor. E torna mais natural a tolerância a faltas nos supervisores.

O ambiente de manufatura é de natureza dinâmica. Não é possível prever antecipadamente o estado do sistema em cada momento do processo de produção. Por isso, as tomadas de decisão do supervisor devem considerar o estado do sistema em tempo real. Por esse motivo, a utilização de escalonamento dinâmico tem sido largamente discutida na literatura, como discutido no Capítulo 1.

3.1 O Supervisor da Célula

O SC implementa a coordenação de eventos dentro de uma célula do SFM à qual está associado, restringindo, dentre outras funcionalidades, a possível trajetória de uma

peça dentro da célula. Essa tarefa de supervisão é realizada através da comunicação com os gerenciadores locais de recursos. A seguir descrevemos a seqüência de eventos que o supervisor deve coordenar. Essa seqüência é ilustrada na Figura 3.1, de acordo com a definição de SFM introduzida no Capítulo 2.

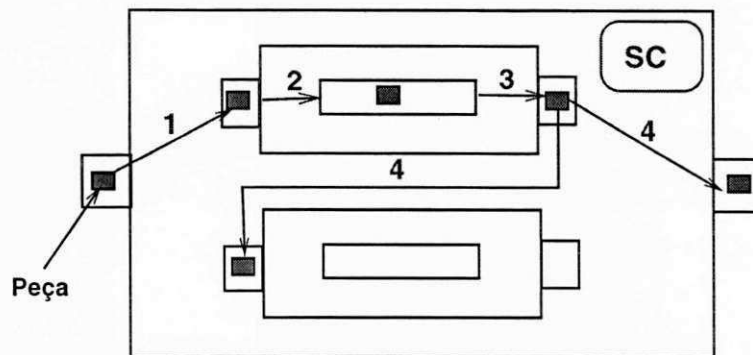


Figura 3.1: O Supervisor da Célula

1. Quando uma peça é depositada no *buffer* de entrada de uma célula, ela é transportada para o *buffer* de entrada do próximo recurso especificado na ordem de produção, quando houver espaço disponível neste *buffer*.
2. O recurso deve ser alocado para uma peça quando estiver disponível.
3. Após obter o recurso e ser processada, a peça é movimentada para o *buffer* de saída desse recurso.
4. A peça é transportada para o *buffer* de entrada de um outro recurso dentro da mesma célula, ou para o *buffer* de saída da célula.

A peça é depositada no *buffer* de saída da célula quando o processo de produção tiver sido completado, ou o próximo recurso especificado na ordem de produção não existir dentro daquela célula. No primeiro caso, o SC deve providenciar o envio da peça para a saída do sistema. No segundo caso, o SC deve interagir com os SCs que possuem o próximo recurso do tipo especificado na ordem de produção, para negociar o envio da peça.

3.2 O Protocolo de Reserva de Recurso

Como mencionando anteriormente, devido à solução distribuída, um SC precisa interagir com outros SCs para negociar o envio de peças para outras células. Para isso, propomos um protocolo de reserva de recurso com o objetivo de disciplinar a interação entre esses SCs. Quando uma peça incompleta é depositada no *buffer* de saída de uma célula, o SC associado àquela célula envia uma mensagem para todos os SCs que possuem o próximo recurso do tipo especificado na ordem de produção daquela peça.

As respostas recebidas vêm acompanhadas de informações de estado das células candidatas ao processamento da peça. Para decidir qual a melhor opção dentre as propostas enviadas, o SC solicitante avalia as alternativas e retorna o “melhor” valor de acordo com os critérios definidos para o sistema, as quais apresentaremos adiante. Essa forma de escolha de destino de peças promove um roteamento dinâmico de peças entre células, isto é, ao invés de uma rota estática pré-determinada para peça, o próximo destino é decidido localmente em tempo real com base em informações globais coletadas através de mensagens pelo SC. A seguir descrevemos o algoritmo do protocolo de reserva de recurso:

1. Um SC solicita o envio de peças através de mensagens de solicitação de serviço para os SCs que podem executar a próxima operação,
2. Após um período de tempo (desprezível em relação aos tempos de manipulação e transporte de subprodutos), o supervisor deve receber respostas dos SCs solicitados,
3. O SC avalia as alternativas e, com base num critério de custo, escolhe um destino para enviar a peça, e
4. Informa aos outros supervisores que enviaram respostas, que não precisa mais do serviço solicitado.

Se após um certo intervalo de tempo t , o supervisor solicitante não receber todas as respostas aguardadas, a avaliação é feita somente com as alternativas disponíveis. Se não houver mais de uma alternativa após o intervalo de espera, o supervisor seleciona

a única oferta disponível. As respostas que chegarem posteriormente são simplesmente descartadas.

A avaliação das alternativas para a escolha do destino da peça é feita com base nos seguintes critérios de custo:

1. Menor trabalho em fila: escolhe a máquina com a menor soma de tempos de processamento de peças já esperando no *buffer*,
2. Menor tempo de processamento: escolhe a máquina com o menor tempo de processamento para a operação requerida.

Se após um intervalo de tempo $t' > t$, o supervisor solicitante não receber nenhuma resposta, assume-se que todos os recursos que poderiam atendê-lo falharam, e não há como continuar o processo de manufatura daquela peça. Com isso, o supervisor envia uma mensagem para o ambiente do sistema, informando que não é possível dar andamento àquela atividade, e aguarda até receber um sinal que permita a reinicialização do protocolo, indicando que o problema foi resolvido.

Na Figura 3.2, ilustramos uma situação em que todos os recursos de um determinado tipo falharam. Nesse caso, como o SC solicitante não recebe nenhuma resposta, ele envia uma mensagem de erro para o ambiente do sistema, e aguarda um retorno para poder continuar o serviço.

Como o supervisor solicitado simplesmente não responde se não puder atender a solicitação, a ocorrência de faltas nos SCs solicitados é tratada da mesma forma, do ponto de vista do SC solicitante.

É importante observar que a parte de solicitação de serviço do protocolo deve ser implementada também na entrada do sistema para que a matéria prima possa ser transportada da entrada do sistema para o *buffer* de entrada das células.

Na recepção de uma solicitação de serviço, um supervisor responde quando há espaço disponível em *buffer* e o recurso solicitado está operacional. Se o *buffer* estiver cheio, o pedido é guardado e atendido tão logo haja espaço disponível. Se o recurso solicitado estiver em falta, o supervisor não responde aos pedidos endereçados para aquele recurso e providencia o reescalonamento das peças que estiverem naquele *buffer*.

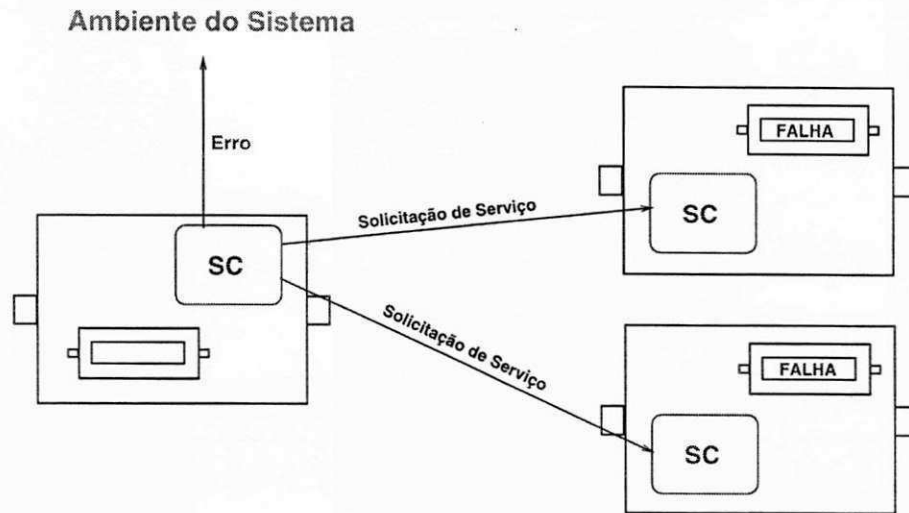


Figura 3.2: Situação de falha em todos os recursos de um mesmo tipo

3.3 Sistema de Transporte Entre Células

Assumimos uma central de AGVs para coordenar o transporte físico de peças de uma célula para outra. Como mencionado no Capítulo 2, no contexto deste trabalho AGVs são veículos auto guiados utilizados para transportar peças entre células. Entretanto, é importante observar que AGVs poderiam transportar peças entre outros componentes do SFM, e por esse motivo, são considerados recursos de transporte bastante flexíveis [39]. Detalhes de sistemas de AGVs, como políticas de escalonamento de veículos, rotas de passagem, definição de pontos de parada, tratamento de colisão e outros, não são tratados neste trabalho. O leitor interessado pode encontrar informações em [43; 31; 47; 29; 27; 38].

Assumimos neste trabalho uma central de AGVs com controle supervisor independente. Isto significa que quando um SC tiver decidido, após a interação com outros SCs, para onde enviar uma peça, ele deve solicitar o serviço da central para transportar a peça para o destino especificado. De fato, a forma de organização da solução para o sistema de AGVs é irrelevante no contexto deste trabalho.

A central deve então alocar um AGV para fazer o serviço e enviá-lo para a célula solicitante. Como neste trabalho não estamos tratando políticas de escalonamento de AGVs, o único critério de alocação considerado é que o AGV deve estar livre, ou seja, não alocado para outra chamada. Isto significa que o AGV escolhido fica

indisponível para outras tarefas até ter completado o transporte da peça para o qual foi designado. Após entregar a peça ao seu destino, o AGV fica disponível para atender novas chamadas.

A central de AGVs pode ser comparada a uma central de rádio táxi, isto é, uma vez concluído o transporte de um cliente para o seu destino, o veículo está livre para uma próxima corrida notificada pela central.

3.4 Tolerância a Faltas em Recursos

Uma fonte de incertezas em SFM é a confiabilidade das máquinas, que estão sujeitas a falhas, como quebra de ferramentas, desgaste pelo uso, desalinhamento, etc., podendo resultar em erros indesejados no comportamento do sistema. A possibilidade de tolerar faltas através de mecanismos que permitam a continuidade do processo de manufatura mesmo na ocorrência de mau funcionamento de alguns componentes é um atributo importante para um SFM. Para tratar falhas em recursos, assumimos réplicas de recursos de produção que podem assumir a função de um outro recurso em caso de falha.

As peças que estão no *buffer* de um recurso em falta ou no *buffer* de sua célula devem ser reescaloadas para uma réplica do recurso. O supervisor executa esta tarefa e, ao ser notificado que um determinado recurso quebrou, providencia o reescaloadamento das peças para uma réplica.

No caso de falha num supervisor, a produção pode continuar se houver outro supervisor que possa atender às solicitações para aquele que falhou. Os seguintes casos de falha são considerados no modelo:

Situação 1

Existem ordens de produção na entrada do sistema ou *buffer* de saída de uma célula, e o supervisor solicitado ou uma das réplicas do próximo recurso de produção falharam, Figura 3.3.

Neste caso, o SC falho ou que contém o recurso em falha não responde ao SC solicitante no sistema. Entretanto, como chegam respostas de outros SCs, o SC solicitante avalia somente as alternativas que recebeu.

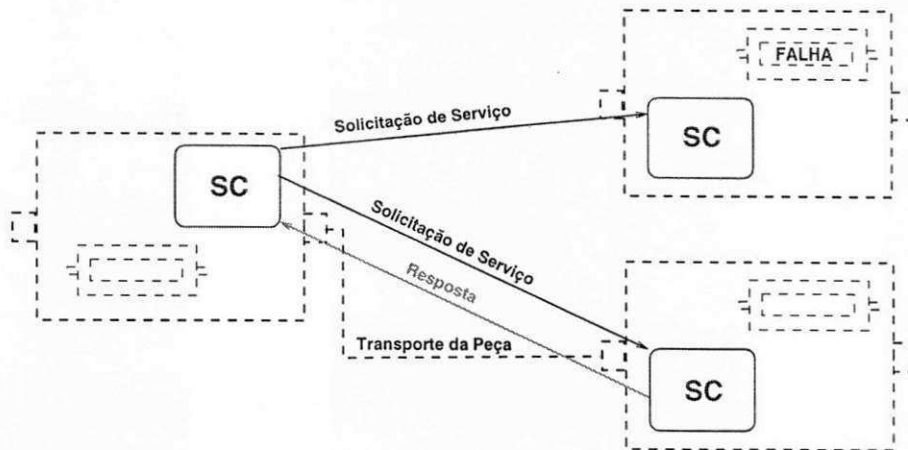


Figura 3.3: Falha em um dos recursos

Situação 2

Existem ordens de produção na entrada do sistema ou *buffer* de saída de uma célula e os supervisores solicitados ou todas as réplicas do próximo recurso solicitado falharam, Figura 3.4.

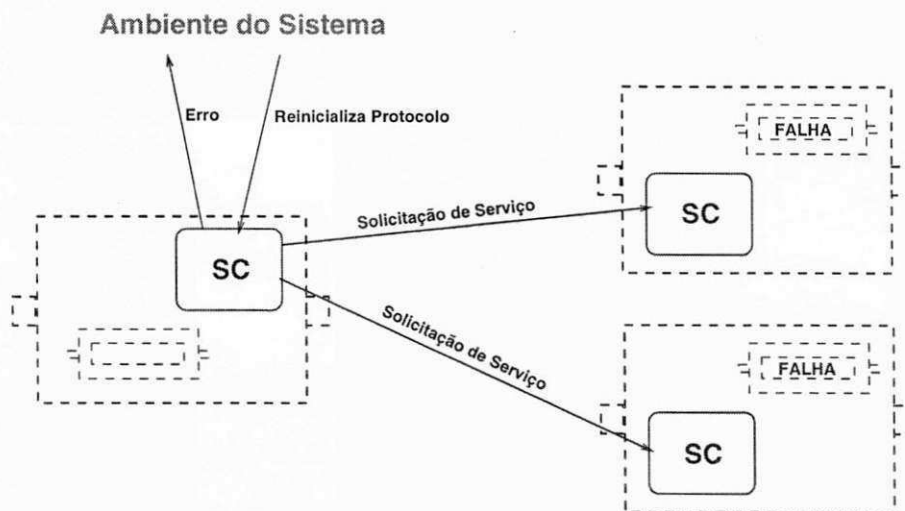


Figura 3.4: Falha em todos os recursos

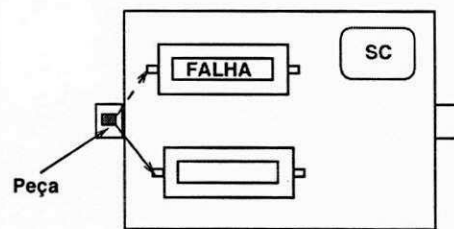
Neste caso, o sistema fica bloqueado e não há como continuar o processo de produção. O SC solicitante então envia uma mensagem de erro para o ambiente do sistema, e aguarda até receber uma notificação indicando que o protocolo pode ser reinicializado.

Situação 3

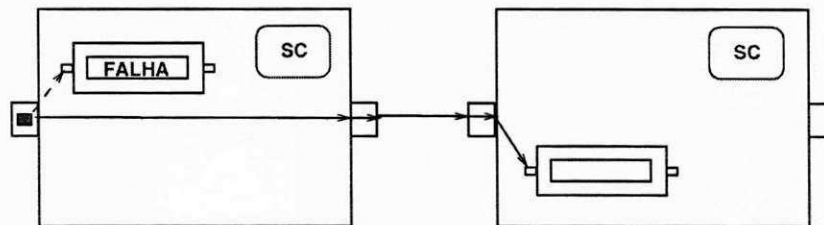
Houve falha no recurso e peças estavam armazenadas no *buffer* da célula aguardando para serem enviadas para o *buffer* do recurso, Figura 3.5. Neste caso, as peças são reescaloadas para a respectiva réplica do recurso que pode estar:

3a) Na mesma célula.

3b) Em outra célula.



3a) Réplica na mesma célula



3b) Réplica em outra célula

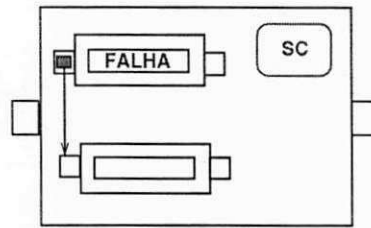
Figura 3.5: Falha com peça no *buffer* da célula

Situação 4

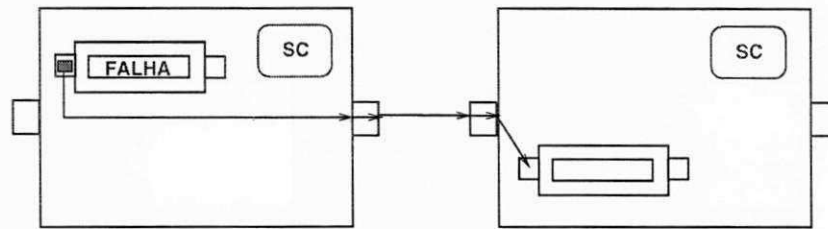
Houve falha no recurso e peças estavam armazenadas no *buffer* da máquina, aguardando para serem processadas, Figura 3.6. Neste caso, as peças são reescaloadas para a respectiva réplica do recurso que pode estar:

4a) Na mesma célula.

4b) Em outra célula.



4a) Réplica na mesma célula



4b) Réplica em outra célula

Figura 3.6: Falha com peça no *buffer* da máquina

3.5 Conclusão

Neste capítulo apresentamos uma solução conceitual para supervisores distribuídos em SFM. Para tanto apresentamos a descrição de um supervisor para as células do SFM (o Supervisor da Célula SC). Descrevemos um protocolo de reserva de recurso para disciplinar a interação entre os SCs, um sistema de transporte entre células baseado em AGVs, e as situações de falha em recursos consideradas no trabalho.

Capítulo 4

Procedimento de Modelagem

Conforme exposto no capítulo anterior, nosso objetivo é apresentar uma solução supervisores distribuídos de um SFM. Para isso, construímos modelos em redes de Petri utilizando a ferramenta Design/CPN [26], que permite a edição, simulação e análise formal de redes de Petri Coloridas, Hierárquicas e Temporizadas. Como já apresentado no Capítulo 2, essas redes permitem que o projetista modele o sistema de forma mais compacta. As inscrições e declarações são feitas em CPN-ML [25; 23].

4.1 Hierarquia de Páginas do Modelo

Nesta seção, apresentamos a página que define a hierarquia do modelo. Na Figura 4.1 os retângulos arredondados ligados por arcos, representam as outras páginas para os módulos do supervisor¹. Os retângulos de onde se originam os arcos orientados denotam as superpáginas, e os retângulos onde terminam tais arcos denotam as subpáginas associadas às transições de substituição. O retângulo Hierarchy representa a mesma página de hierarquia. O retângulo Declarations representa a página de declarações de *color sets*, variáveis, constantes e funções em CPN-ML. O retângulo Cells representa a página do modelo dos SCs. O retângulo Snd_BufBuf representa a página do modelo do transporte de uma peça de um *buffer* de recurso para outro *buffer* de recurso dentro da mesma célula. O retângulo Send representa a página do módulo de Envio do Protocolo

¹A inscrição prime associada à página, informa ao simulador que a rede daquela página deve ser executada na simulação

de Reserva de Recurso. O retângulo Request representa a página do modelo para Geração das Solicitações de Serviço. O retângulo Answers representa a página do modelo de Tratamento de Respostas. O retângulo Receive representa a página do módulo de Recepção do Protocolo de Reserva de Recurso. O retângulo Transport representa a página do modelo do Sistema de Transporte entre Células. O retângulo Verify_Net representa a página utilizada para fazer as verificações das propriedades do modelo.

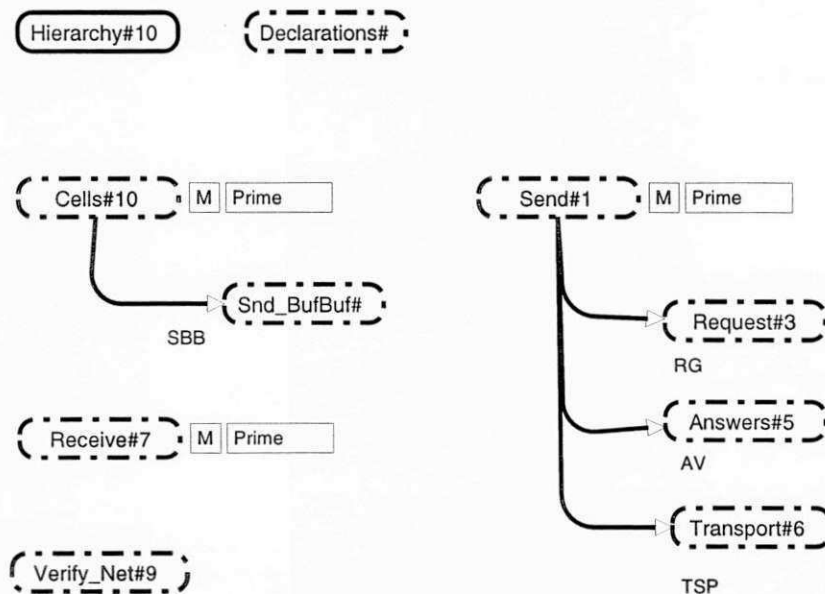


Figura 4.1: Hierarquia de páginas para o modelo CPN do supervisor

4.2 O Supervisor para as Células

Na Figura 4.2 mostramos o modelo do supervisor para as células. As declarações para o conjunto de cores, variáveis e as funções utilizadas são apresentadas na Seção 4.8. Os lugares e transições possuem o seguinte significado:

Lugares:

CIB_C modela os *buffers de entrada* das células;

COB_C modela os *buffers de saída* das células²;

²As inscrições FG+nome adjacentes aos lugares, denotam um conjunto de fusão. Por exemplo, FG e

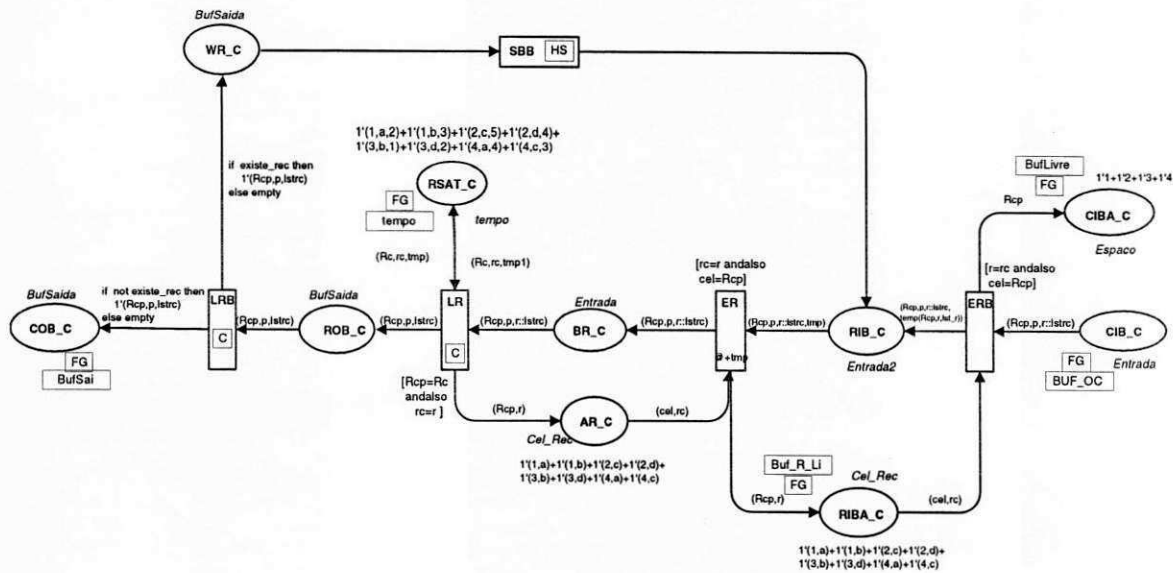


Figura 4.2: O Supervisor para Célula

RIB_C modela os *buffers de entrada* dos recursos;

ROB_C modela os *buffers de saída* dos recursos;

BR_C e AR_C modelam respectivamente *recurso ocupado* e *recurso livre*;

RIBA_C e CIBA_C modelam o estado de livre dos *buffers* de entrada dos recursos e dos *buffers* de entrada das células respectivamente;

WR_C modela o estado de espera de peças para serem transportadas para o *buffer* de entrada de um recurso;

RSAT_C possui informação sobre a estimativa de tempo de atendimento para os recursos, considerando também as peças já esperando para serem atendidas.

Transições:

ERB representa o transporte de uma peça do *buffer* de entrada de uma célula para o *buffer* de entrada de um recurso;

ER representa a alocação de um recurso³;

Bufsai próximo ao lugar COB_C, significa que este lugar pertence a um conjunto de fusão denominado de Bufsai.

³A inscrição @+tmp representa uma região de tempo associada à transição, que define um atraso para a disponibilização das fichas de saída

LR representa a liberação de um recurso e a atualização sobre estimativa de tempo de atendimento para o recurso;

LRB modela o transporte de uma peça para o *buffer* de saída de uma célula ou a transição de uma peça para um estado de espera para ir para o *buffer* de entrada de outro recurso dentro da mesma célula;

SBB representa o transporte de uma peça do *buffer* de saída de um recurso para o *buffer* de entrada de outro recurso dentro de uma mesma célula⁴.

Conforme mostrado na Figura 4.2, a marcação inicial do modelo define fichas nos lugares CIB_C, CIBA_C, RIBA_C, AR_C e RSAT_T. A marcação do lugar CIB_C é:

$M(\text{CIB_C}) = (\text{Cel}, \text{Peça}, \text{lstRc})$ onde,

1. Cel identifica a célula atual;
2. Peça identifica a peça que está sendo manufaturada;
3. lstRc identifica a lista de recursos necessários à manufatura da peça.

A marcação inicial mostrada no lugar $M(\text{CIB_C})$ é $1'(1, \text{Purse}, [a,b,c])$, indicando que no *buffer* de entrada da célula 1 existe uma peça do tipo Purse, para a qual será alocado inicialmente um recurso do tipo a.

A marcação do lugar CIBA_C é:

$M(\text{CIBA_C}) = (\text{Cel})$ onde,

Cel é um tipo inteiro que identifica a célula;

A marcação mostrada em $M(\text{CIBA_C})$ é $1'1 + 1'2 + 1'3 + 1'4$ indicando uma posição disponível no *buffer* de cada célula. Por exemplo, $1'1$ indica uma posição disponível no *buffer* da célula 1, $1'2$ indica uma posição disponível no *buffer* da célula 2 e assim por diante.

A marcação no lugar RIBA_C identifica os espaços disponíveis no *buffer* de cada recurso, onde temos:

$M(\text{RIBA_C}) = (\text{Cel}, R)$ onde:

⁴A inscrição HS indica que SBB é uma transição de substituição.

1. Cel identifica a célula;
2. R identifica o recurso.

A marcação do lugar AR_C indica os recursos disponíveis, onde tem-se:

$M(AR_C) = (Cel, R)$ indica os recursos disponíveis e:

1. Cel identifica a célula;
2. R identifica o recurso.

A marcação do lugar RSA_T indica a estimativa de tempo de espera para utilização de cada recurso:

$M(RSAT_C) = (Cel, R, tmpo)$ onde:

1. Cel identifica a célula;
2. R identifica o recurso;
3. tmpo indica a estimativa de tempo de atendimento de um recurso, considerando também as peças já esperando para serem atendidas.

A informação de tempo para processamento do recurso de produção é recuperada através da função:

$temp(Rcp, r, lst_r)$

Esta função é invocada na inscrição do arco de saída da transição ERB para o lugar RIB_C. Desta maneira, a informação tempo para processamento do recurso é colocada na ficha depositada em RIB_C. Quando a transição ER ocorre, o atraso de disponibilização da ficha, definido pela inscrição @+tmp, é substituído pelo valor da variável tmp, do arco de saída do lugar RIB_C para a transição ER.

O depósito de uma ficha no lugar CIB_C, que representa a chegada de uma peça no *buffer* de entrada de uma célula, habilita a transição ERB, cuja ocorrência representa o envio de uma peça do *buffer* de entrada de uma célula para o *buffer* de entrada de uma máquina. A ocorrência da transição ERB deposita uma ficha no lugar RIB_C, que modela o *buffer* de entrada de uma máquina. Uma ficha em RIB_C habilita a transição ER, que representa a obtenção de um recurso por uma peça. A ocorrência

da transição ER deposita uma ficha em BR_C, que modela uma peça sendo processada por uma máquina. Essa ficha habilita a transição LR, cuja ocorrência deposita uma ficha no lugar ROB_C, que representa o *buffer* de saída de um recurso. Uma ficha no lugar ROB_C habilita a transição LRB que representa a passagem da peça para um estado de espera para ser transportada para o *buffer* de entrada de outra máquina, ou o transporte de uma peça do *buffer* de saída de uma máquina para o *buffer* de saída de uma célula. A ocorrência de LRB deposita uma ficha no lugar WR_C no primeiro caso, ou uma ficha no lugar COB_C, que modela o *buffer* de saída da célula, no segundo caso.

As transições deste modelo representam atividades de transporte de peças dentro das células. Entretanto, detalhes do procedimento de transporte não foram modelados. Podendo contudo, serem facilmente adicionado ao modelo através da construção de redes hierárquicas associadas às referidas transições.

Na Figura 4.3 mostramos a página associada à transição de substituição SBB⁵, do modelo do supervisor para células (Figura 4.2), que representa o transporte de uma peça do *buffer* de saída de uma máquina para o *buffer* de entrada de outra máquina dentro da mesma célula. Os significados dos lugares e transições são os seguintes:

Lugares:

RIBAI_C é lugar de fusão com RIBA_C da Figura 4.2;

WR_C é lugar de fusão com WR_C da Figura 4.2;

RSAT_E é lugar de fusão com RSAT_C da Figura 4.2;

RIB_C é lugar de fusão com RIB_C da Figura 4.2;

Transição:

SDBUF representa o transporte de peças do *buffer* de saída de uma máquina para o *buffer* de entrada de outra máquina dentro de uma mesma célula.

⁵As inscrições P In e P Out representam portas de entrada e de saída respectivamente, associadas aos lugares WR_C e RIB_C da Figura 4.2, os quais representam os *sockets* associados a essas portas.

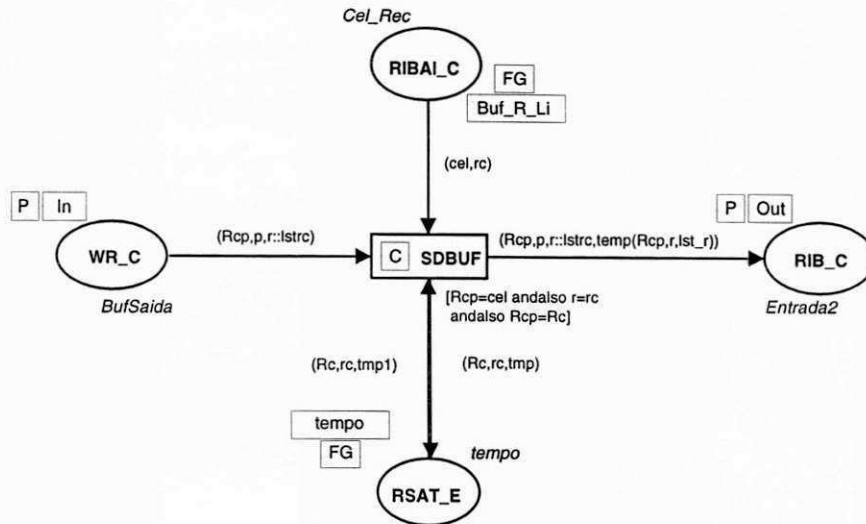


Figura 4.3: Transporte de peças do *buffer* de saída de uma máquina para o *buffer* de entrada de outra máquina

Uma ficha em WR_C representa o estado de espera de uma peça para ir do *buffer* de saída de uma máquina para o *buffer* de entrada de uma outra máquina dentro da mesma célula. Havendo espaço disponível no *buffer* da máquina destino (ficha em RIBAI_C), a ocorrência da transição SDBUF representa o envio de uma peça para o *buffer* destino e a atualização da informação sobre estimativa de tempo para utilização do recurso no lugar RSAT_E.

4.3 Protocolo de Reserva de Recurso: Módulo de Envio

Na Figura 4.4 mostramos o módulo de Envio do Protocolo de Reserva de Recurso. O significado dos lugares e transições são os seguintes:

Lugares:

B_S representa o *buffer* de entrada do sistema;

COB_S representa o *buffer* de saída das células, e o *buffer* de entrada do sistema, é lugar de fusão com o lugar COB_C da Figura 4.2;

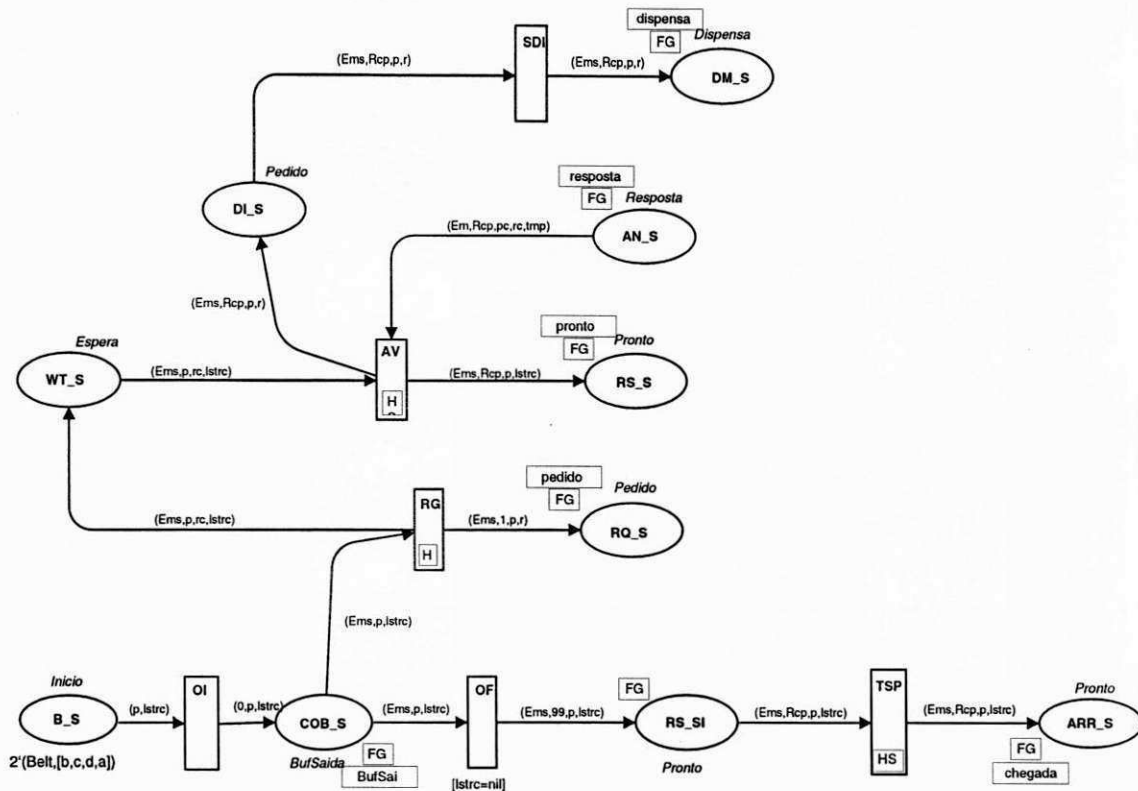


Figura 4.4: Protocolo de Reserva de Recurso: Módulo de Envio

Fichas em RQ_S representam os pedidos de envio de peças para células;

Fichas em AN_S representam a chegada de respostas aos pedidos feitos;

Fichas em DI_S representam informação de dispensa para as respostas às solicitações feitas que não serão utilizadas;

Fichas em DM_S representam o envio dessa informação de dispensa;

Uma ficha em WT_S representa o estado de espera de uma resposta após o envio de pedidos de envio de peças;

Fichas em RS_S e RS_SI são lugares de fusão e representam o estado de peças prontas para serem transportadas;

Fichas em ARR_S representam a chegada de uma peça ao seu destino após ser servida pelo sistema de transporte entre células.

Transições:

OI representa a chegada de uma ordem de produção no sistema;

RG representa a geração de solicitações de envio de peças para as células que possuem o próximo recurso necessário para a manufatura da peça;

AV representa o tratamento das respostas às solicitações que chegam;

TSP representa o sistema de transporte de peças entre células;

SDI representa o envio de informação de dispensa para pedidos não utilizados;

OF representa a passagem da peça para o estado de pronta para ser transportada para a saída do sistema.

Conforme mostra a Figura 4.4, a marcação inicial do modelo contém fichas no lugar B_S :

$M(B_S) = (Peça, lstRc)$ em que,

1. Peça identifica a peça a ser manufaturada;
2. $lstRc$ identifica a lista de recursos necessários à manufatura da peça;

A marcação inicial mostrada no lugar $M(B_S)$ é $2'(Belt, [b, c, d, a])$, indicando que existe uma ordem de produção no *buffer* de entrada do sistema para manufaturar dois produtos do tipo Belt, e os recursos necessários para a manufatura são a seqüência b, c, d, a.

No início do processamento, têm-se as ordens de produção de peças no lugar B_S . A ocorrência de OI representa o envio da matéria-prima para o *buffer* de entrada do sistema (lugar COB_S). A transição RG representa a geração de um pedido para cada célula habilitada a receber o serviço. A chegada de respostas, representadas por fichas no lugar AN_S , habilita a transição AV, cuja ocorrência representa a utilização de um critério de custo para decidir para onde enviar a peça colocando a informação de dispensa para as células não escolhidas no lugar DI_S , e a peça no estado *pronto para envio* no lugar RS_S . A ocorrência da transição SDI representa o envio de informações de dispensa de recursos reservados para outras células.

Quando o processo de manufatura da peça estiver completo, a ocorrência de OF representa a passagem da peça para o estado de pronta para ser transportada para a saída do sistema. A transição TSP representa o transporte da peça entre as células do sistema.

Deve ser observado que cada produto corresponde a uma ordem de produção. Desta forma, diversas ordens de produção podem estar sendo executadas concorrentemente.

4.4 Geração de Solicitações de Serviço

Na Figura 4.5 mostramos a página associada à transição RG, da Figura 4.4, que gera solicitações de envio de peças. O significado dos lugares e transições são os seguintes:

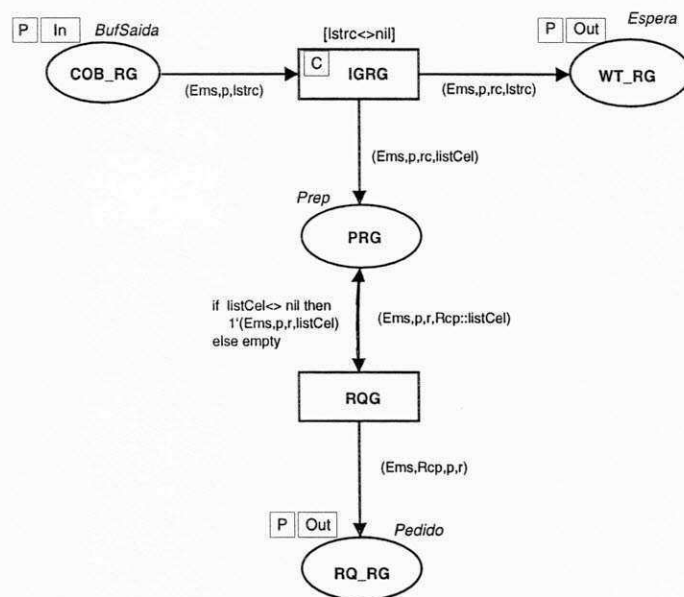


Figura 4.5: Geração de Solicitações de Serviço

Lugares:

COB_RG é lugar de fusão com COB_S da Figura 4.4;

WT_RG é lugar de fusão com WT_S da Figura 4.4;

Uma ficha em PRG modela a informação sobre todas as células que possuem um determinado recurso;

Uma ficha em RQ_RG modela o envio de uma solicitação de serviço para uma determinada célula, sendo lugar de fusão com RQ_S da Figura 4.4.

Transições:

IGRG representa a recuperação de informação sobre todas as células que possuem um determinado recurso;

RQG representa a geração de uma solicitação de serviço para cada célula que possui um determinado recurso.

Uma ficha no lugar COB_RG, representando uma peça no *buffer* de saída de uma célula, habilita a transição IGRG. A ocorrência de IGRG representa a recuperação de informação sobre todas as células que possuem o próximo recurso necessário para atender a peça, colocando essa informação numa ficha no lugar PRG e a peça em estado de espera depositando uma ficha em WT_RG. A ocorrência de RQG representa a geração de uma solicitação de serviço para cada célula que possui o próximo recurso necessário para a continuação da tarefa de manufatura.

4.5 Protocolo de Reserva de Recurso: Módulo de Recepção

A Figura 4.6 mostra o módulo de Recepção do Protocolo de Reserva de Recurso. O significado dos lugares e transições são os seguintes:

Lugares:

SO modela a saída do sistema;

CIB_R modela o *buffer* de entrada das células (lugar de fusão com CIB_C da Figura 4.2);

CIBA_R é lugar de fusão com CIBA_C da Figura 4.2;

Uma ficha em CIBR_R significa que um espaço no *buffer* de uma célula está reservado;

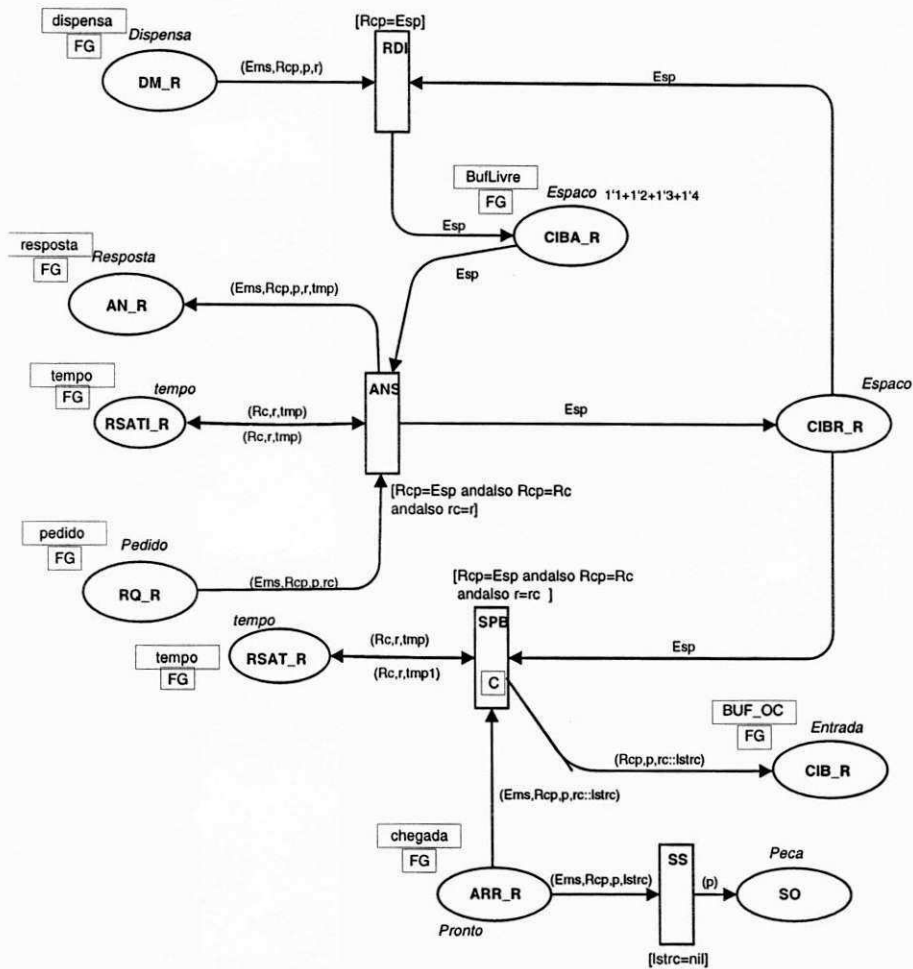


Figura 4.6: Protocolo de Reserva de Recurso: Módulo de Recepção

Fichas em RQ_R modelam a chegada dos pedidos de envio de peças para células (lugar de fusão com RQ_S da Figura 4.4);

Fichas em AN_R modelam o envio das respostas aos pedidos de solicitação de serviço (lugar de fusão com AN_S da Figura 4.4);

Fichas em DM_R modelam a chegada de informação de dispensa para células que responderam (lugar de fusão com DM_S da Figura 4.4);

Uma ficha em ARR_R modela a chegada de uma peça a uma célula destino (lugar de fusão com ARR_S da Figura 4.4);

Os lugares RSAT_R e RSATI_R são lugares de fusão e mantêm informação sobre estimativa de tempo de espera para utilização dos recursos (lugar de fusão com

RSAT_C da Figura 4.2).

Transições:

SS representa a saída de uma peça do sistema;

ANS representa a geração de reserva de *buffer* para uma solicitação de serviço e o envio de resposta à solicitação;

SPB representa o transporte de uma peça para o *buffer* de entrada de uma célula e a atualização de RSAT_RI;

RDI representa a liberação de reserva de *buffer* para pedidos não utilizados.

Uma solicitação de envio de peça é aceita por uma célula quando houver espaço livre no seu *buffer* (ficha em CIBA_R). Então uma reserva de *buffer* é feita pela célula receptora. Este evento é representado pela ocorrência de ANS, que usa informações de estado mantidas no lugar RSAT_R para representar o envio de uma resposta para a célula solicitante com informação sobre estimativa de tempo de atendimento para aquele recurso.

O depósito de uma ficha em ARR_R modela a chegada de uma peça na célula destino, que pode ir para o *buffer* de entrada de uma célula (lugar CIB_R), evento representado pela ocorrência da transição SPB, que atualiza RSAT_RI e libera a reserva de *buffer* removendo uma ficha de CIB_R, ou ir para a saída do sistema (ocorrência de SS), quando do término da manufatura. Se em vez de uma peça para a célula receptora, vier uma informação de dispensa de reserva (ficha em DM_R), o espaço em *buffer* reservado para aquele pedido é liberado (ocorrência de RDI).

4.6 Tratamento das Respostas

Na Figura 4.7 mostramos a página associada à transição de substituição AV do modelo de envio do protocolo de reserva de recurso (Figura 4.4) que trata as respostas recebidas pela célula solicitante. O significado dos lugares e transições são os seguintes:

Lugares:

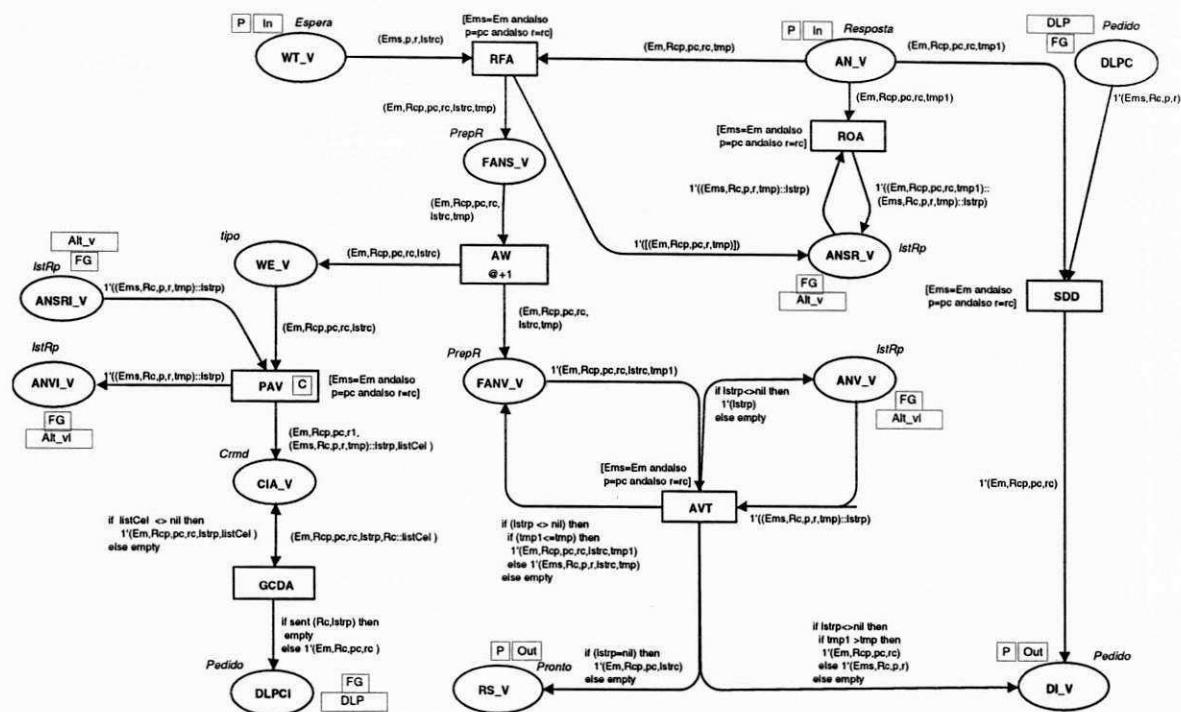


Figura 4.7: Tratamento de Respostas

WT_V é lugar de fusão com WT_S da Figura 4.4;

AN_V é lugar de fusão com AN_S da Figura 4.4;

Uma ficha em FANS_V modela a primeira resposta recebida pela célula;

Fichas em ANSR_V e ANSRI_V são lugares de fusão e modelam a recepção de respostas subsequentes à primeira;

Uma ficha em WE_V modela o atraso da primeira resposta recebida, indicando um estado de espera por outras respostas;

Uma ficha em FANV_V modela a primeira resposta recebida aguardando para ser avaliada;

ANV_V e ANVI_V são lugares de fusão e modelam respostas, que chegaram após a primeira, no estado de prontas para serem avaliadas;

RS_V é lugar de fusão com RS_S da Figura 4.4;

DI_V é lugar de fusão com DI_S da Figura 4.4;

Fichas em CIA_V modelam informação sobre células para as quais foram solicitados serviços;

DLPC e DLPCI são lugares de fusão e modelam informação de controle para respostas atrasadas.

Transições:

RFA representa a recepção da primeira resposta à uma solicitação;

A ocorrência de AW representa o início de um procedimento de atraso para espera de outras respostas depois da recepção da primeira. A inscrição @+tmp representa uma região de tempo associada à transição, que define um atraso para a disponibilização das fichas de saída;

ROA representa a recepção de outras respostas depois da primeira;

PAV representa a habilitação do procedimento de avaliação de respostas, e a recuperação de informações sobre células para as quais foram solicitados serviços;

GCDA representa a geração de informação de controle para respostas atrasadas;

AVT representa o evento de avaliação das respostas;

SDD representa o evento de disponibilizar para envio informação de dispensa para respostas atrasadas.

Na chegada da resposta a uma solicitação, uma ficha no lugar AN_V, se for a primeira resposta, ela é enviada para o lugar FANS_V (ocorrência de RFA). A ocorrência de ROA representa a recepção de respostas subsequentes à primeira, que são modeladas por fichas depositadas no lugar ANSR_V. A ocorrência de AW representa o início de um procedimento de espera por outras respostas depois da recepção da primeira. Isso é modelado pela inclusão da inscrição @+tmp, que define um atraso na disponibilidade das fichas depositadas nos lugares WE_V e FANV_V. Após um tempo t , as fichas depositadas nos lugares WE_V e FANV_V ficam disponíveis, indicando o término do tempo de espera por outras respostas. A ocorrência de PAV representa a recuperação

de informação sobre as células solicitadas, e transfere as fichas contidas em ANSR_V para ANV_V, habilitando a transição AVT, que representa a avaliação das respostas.

A transição GCDA modela a comparação entre as respostas enviadas com as solicitações feitas e gera informação de controle para as respostas atrasadas. A transição AVT representa o evento de selecionar a célula com menor estimativa de tempo de atendimento, colocar a peça no estado de pronto para envio (ficha em RS_V) e enviar informação de dispensa para outras células (fichas em DI_V). A transição SDD representa a disponibilização de informação de dispensa para as células que responderem após o esgotamento do tempo de espera por outras respostas.

4.7 Sistema de Transporte Entre Células

Na Figura 4.8 mostramos a página associada à transição TSP (Figura 4.4) que modela o sistema de transporte entre células. O significado dos lugares e transições são os seguintes:

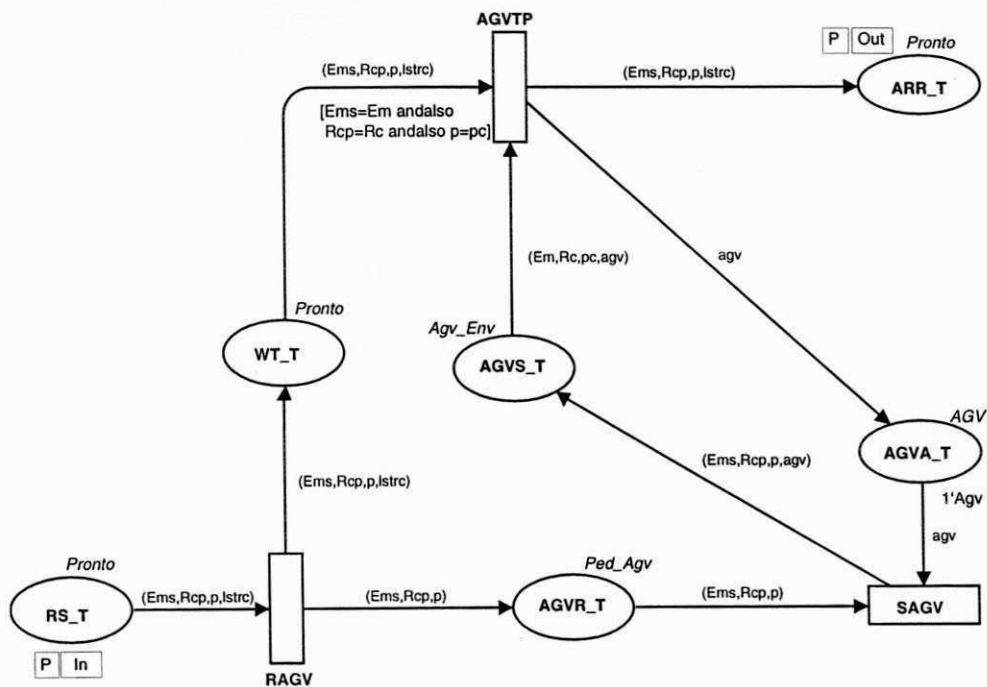


Figura 4.8: Transporte de Peças entre Células

Lugares:

RS_T modela o estado de pronto para envio de uma peça; é lugar de fusão com RS_S da Figura 4.4;

AGVR_T modela chamadas de AGVs para atender serviços de transporte;

WT_T modela o estado de espera por AGV de uma peça;

Fichas em AGVA_T modelam AGVs para atender serviços;

AGVS_T modela o envio de AGVs para atender serviços de transporte;

ARR_T modela a chegada de peças ao seu destino; é lugar de fusão com ARR_S da Figura 4.4.

Transições:

RAGV representa a geração de chamadas de AGVs para atender serviços de transporte;

SAGV representa a alocação de AGVs para atenderem solicitações de transporte;

AGVTP representa o transporte de peças por AGVs.

Havendo uma peça no estado de *pronto para envio* (fichas no lugar RS_T), o sistema de transporte entre células providencia a chamada de um AGV livre para atender o serviço (ocorrência de RAGV), e a peça vai para um estado de *espera por AGV* (ficha em WT_T). Havendo um AGV disponível para atender a chamada (ficha em AGVA_T), ele é enviado (ocorrência de SAGV) para atender o serviço (ficha em AGVS_T). Com uma ficha no lugar AGVS_T e uma ficha no lugar WT_T (peça aguardando AGV), a transição AGVTP fica habilitada, e representa o transporte de uma peça por um AGV. Após o transporte, a peça chega ao seu destino (ficha em ARR_T), e o AGV retorna ao estado livre (ficha em AGVA_T).

4.8 Nó de Declaração

A seguir mostramos o nó de declaração que define os conjuntos de cores, as variáveis e funções utilizadas no modelo.

```

val lista_a = [1,4];
val lista_b = [1,3];
val lista_c = [2,4];
val lista_d = [2,3];
color existerec= bool;
color AGV = with Agv;
color Peca = with Piece1|Piece2 timed;
color INT = int;
color Emissor = INT;
color Receptor = INT;
color R = with a|b|c|d timed;
color Rp = with S|N;
color tmpo = INT;
color Espaco = INT;
color Cel = INT;
color Celulas = list Cel;
color Tab = product R*Celulas;
color lstRc = list R;
color Inicio = product Peca*lstRc;
color BufSaida = product Emissor*Peca*lstRc;
color Espera = product Emissor*Peca*R*lstRc;
color Pedido = product Emissor*Receptor*Peca*R;
color Prep = product Emissor*Peca*R*Celulas;
color PrepR = product Emissor*Receptor*Peca*R*lstRc*tmpo;
color tipo = product Emissor*Receptor*Peca*R*lstRc;
color Entrada = product Receptor*Peca*lstRc;
color Entrada2 = product Receptor*Peca*lstRc*tmpo;
color Pronto = product Emissor*Receptor*Peca*lstRc;
color Resposta = product Emissor*Receptor*Peca*R*tmpo;
color lstRp = list Resposta;
color Crmd = product Emissor*Receptor*Peca*R*lstRp*Celulas;
color Dispensa = product Emissor*Receptor*Peca*R;
color Ped_Agv = product Emissor*Receptor*Peca;
color Chegada = product Emissor*Receptor*Peca*AGV*lstRc;
color Agv_Env = product Emissor*Receptor*Peca*AGV;
color Cel_Rec = product Cel*R;
color tempo = product Receptor*R*tmpo;
var lstrp : lstRp;
var tt : Resposta;
var tmp,tmp1,tp : INT;
var p,pc,p1 : Peca;
var lstrc,lstrc1: lstRc;
var Esp,Espc : Espaco;
var r,rc,r1 : R;
var Ems,Em,Em1 : Emissor;
var listCel : Celulas;
var Rcp,Rc,cel : INT;
var existe_rec : existerec;
var agv : AGV;
val lst_r = [(1,a,2),(1,b,3),(2,c,3),(2,d,4),(3,b,2),(3,d,2),(4,a,1),(4,c,1)];

(* Recupera a informação de tempo de processamento para um recurso *)
fun temp(Rcp,r,(Rc,rc,tp)::y)=
if (Rcp=Rc andalso r=rc)
then tp
else temp(Rcp,r,y);

(* Atualiza a informação de tempo de atendimento para um recurso, adicionando *)
fun calc_temp(Rcp,r,tmp:INT,(Rc,rc,tp)::y)=
if (Rcp=Rc andalso r=rc)
then tmp+tp

```

```

else calc_temp(Rcp,r,tmp,y);

(* Atualiza a informação de tempo de atendimento para um recurso, subtraindo *)
fun calc_temp2(Rcp,r,tmp:INT,(Rc,rc,tp)::y)=
if (Rcp=Rc andalso r=rc)
then tmp-tp
else calc_temp2(Rcp,r,tmp,y);

(* Verifica a existência de um recurso numa célula *)
fun Existe_rec(Rcp,r::lstrc,[]) = false|
Existe_rec(Rcp,r::lstrc,(Rc,rc,tp)::y) =
if (Rcp=Rc andalso r=rc) then
true
else Existe_rec(Rcp,r::lstrc,y);

(* Recupera a lista de células que possui um determinado recurso *)
fun acha_lista(rc::lstrc) =
if (rc=a) then
(rc,lista_a)
else
if (rc=b) then
(rc,lista_b)
else
if (rc=c) then
(rc,lista_c)
else
if (rc=d) then
(rc,lista_d)
else (rc,[]);

(* Verifica se um pedido foi enviado para um determinado supervisor *)
fun sent(Rcp,[]) = false|
sent(Rcp,(Em,Rc,p,r,tmp)::y)=
if Rcp = Rc then
true
else sent(Rcp,y);

```

4.9 Análise do Modelo

Como já mencionado na Seção 2.6 do Capítulo 2, uma das grandes vantagens do emprego de redes de Petri é a possibilidade de fazer a análise dos modelos construídos. Com a ferramenta Design/CPN, podemos analisar os modelos através de simulação e da análise do grafo de ocorrência para verificação das propriedades comportamentais do sistema.

A simulação é a forma mais direta de se analisar um modelo. É bastante útil na fase inicial da modelagem, pois permite ao modelador observar algumas possíveis seqüências de execução. Entretanto, com este método, não é possível obter uma prova completa da exatidão do modelo. Para verificação formal das propriedades comportamentais do

modelo, utilizamos o conjunto de módulos disponíveis na ferramenta Design/CPN para geração do grafo de ocorrência e aplicações de funções de análise ao grafo.

Após a geração do grafo de ocorrência, uma das características que verificamos foi a *inexistência de bloqueios*, indicando que o modelo é vivo em todos os estados, com exceção dos nós referentes aos estados finais. Isto quer dizer que, dada uma ordem de produção modelada pela marcação inicial do lugar B_S (Entrada do Sistema, ver Seção 4.3, Figura 4.4), devemos obter os produtos manufaturados na saída do sistema, modelados pela marcação do lugar SO (Saída do Sistema, ver Seção 4.5, Figura 4.6) para qualquer ordem que as transições ocorram.

Nossa análise foi feita a partir do grafo de ocorrência para a seguinte marcação inicial:

$$M(B_S) = 1'(Piece1, [c,a,d]) + 1'(Piece2,[d,b,a])$$

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 18344

Arcs: 69356

Secs: 5480

Status: Full

Indicando que a geração foi completa com 18344 nós e 69356 arcos. Na Figura 4.9 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO e B_S, indicando duas ordens de produção na entrada do sistema e vazio na saída.

Na Figura 4.10 exibimos a lista de marcações mortas encontradas. Estas marcações correspondem aos estados finais que denotam as diversas possibilidades ou caminhos por onde podem ser processadas as tarefas até o seu resultado final. Podemos também observar na Figura 4.10, que em todos os estados finais encontrados, o conteúdo do lugar SO corresponde às tarefas completadas. Por exemplo, no nó 18344 temos a

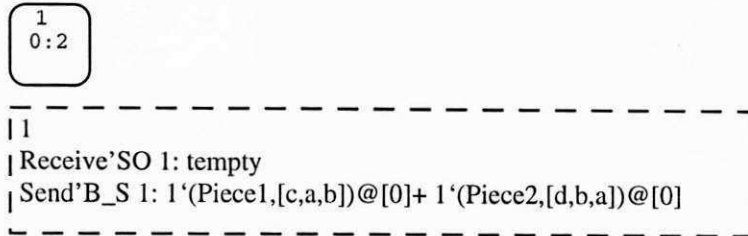


Figura 4.9: Marcação inicial da entrada e saída do sistema

marcação $1'Piece1@[7] + 1'Piece2@[8]$, indicando que as peças Piece1 e Piece2 foram processadas em 7 e 8 unidades de tempo respectivamente.

Outra característica importante que verificamos é a propriedade de *reinicialização* do modelo, isto é, após o término da execução de uma tarefa, é possível ao sistema iniciar outra tarefa. Para verificar isto, observamos a propriedade de *conservação* na rede, e fizemos também a comparação das marcações de alguns lugares nos estados finais com as marcações desses lugares no estado inicial.

Na Figura 4.11 podemos observar que as marcações correspondentes aos lugares que mantêm informações de estado do sistema como: recursos disponíveis (lugar AR_C), espaço em *buffer* de recursos (lugar RIBA_C), espaço em *buffer* de células (lugar CLBA_C) preservam seus valores, indicando que o sistema pode dar início a um novo processo.

4.10 Conclusão

O objetivo deste capítulo foi apresentar uma solução, em redes de Petri, para a modelagem de Supervisores Distribuídos para SFM. Para isso, utilizamos a ferramenta Design/CPN para edição, simulação e análise dos modelos.

Iniciamos o Capítulo com a descrição da página que define a hierarquia do modelo, seguida das redes de Petri modeladas, com a descrição dos lugares, transições e comportamento do modelo. Por fim mostramos os resultados da análise, em que observamos que o modelo é vivo em todos os estados e é reinicializável.

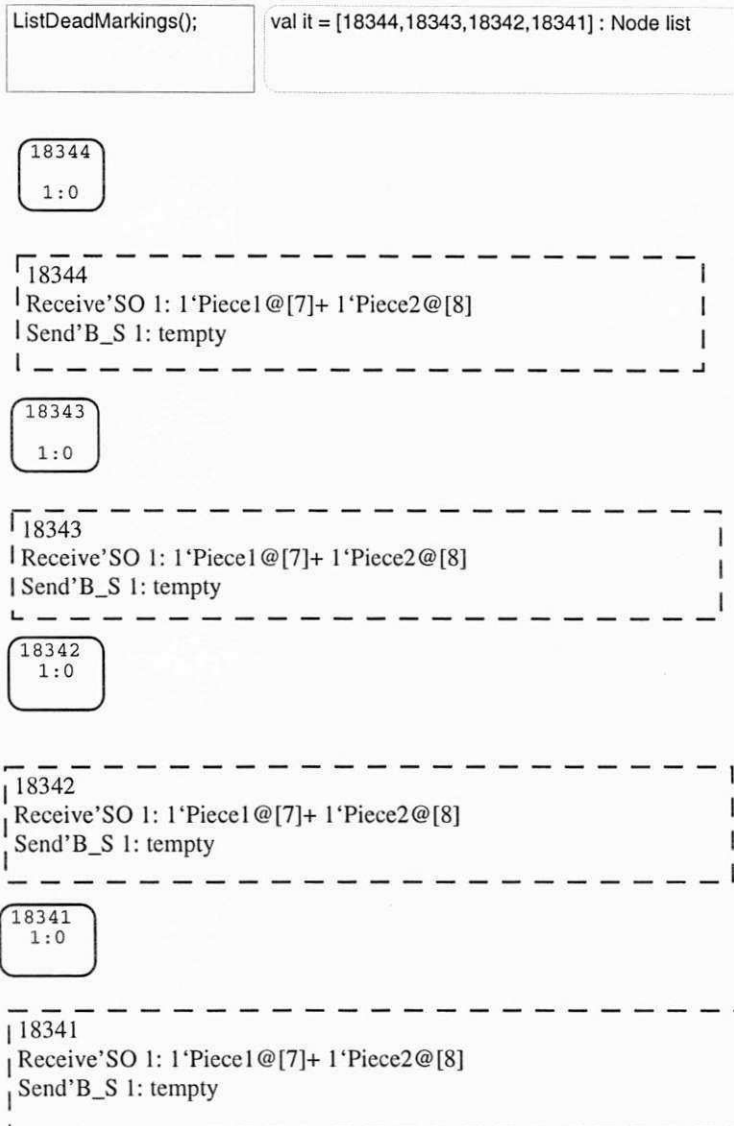


Figura 4.10: Estados Finais

```
ListDeadMarkings(); val it = [18344,18343,18342,18341] : Node list
```

1
0:2

```
1
|
| Cells'RIBA_C 1: 1'(1,a)@[0]+ 1'(1,b)@[0]+ 1'(2,c)@[0]+ 1'(2,d)@[0]+ 1'(3,b)@[0]+ 1'(3,d)@[0]+ 1'(4,a)
| @ [0]+ 1'(4,c)@[0]
|
| Cells'RSAT_C 1: 1'(1,a,2)@[0]+ 1'(1,b,3)@[0]+ 1'(2,c,5)@[0]+ 1'(2,d,4)@[0]+ 1'(3,b,1)@[0]+ 1'(3,d,2)@[
| 0]+ 1'(4,a,4)@[0]+ 1'(4,c,3)@[0]
|
| Cells'CIBA_C 1: 1'1+ 1'2+ 1'3+ 1'4
|
| Cells'AR_C 1: 1'(1,a)@[0]+ 1'(1,b)@[0]+ 1'(2,c)@[0]+ 1'(2,d)@[0]+ 1'(3,b)@[0]+ 1'(3,d)@[0]+ 1'(4,a)@[
| 0]+ 1'(4,c)@[0]
|
| Transport'AGVA_T 1: 1'Agv
|
```

18344
1:0

```
18344
|
| Cells'RIBA_C 1: 1'(1,a)@[8]+ 1'(1,b)@[0]+ 1'(2,c)@[0]+ 1'(2,d)@[0]+ 1'(3,b)@[7]+ 1'(3,d)@[3]+ 1'(4,a)
| @ [3]+ 1'(4,c)@[2]
|
| Cells'RSAT_C 1: 1'(1,a,2)@[8]+ 1'(1,b,3)@[3]+ 1'(2,c,5)@[0]+ 1'(2,d,4)@[1]+ 1'(3,b,1)@[7]+ 1'(3,d,2)@[
| 3]+ 1'(4,a,4)@[5]+ 1'(4,c,3)@[2]
|
| Cells'CIBA_C 1: 1'1+ 1'2+ 1'3+ 1'4
|
| Cells'AR_C 1: 1'(1,a)@[8]+ 1'(1,b)@[0]+ 1'(2,c)@[0]+ 1'(2,d)@[0]+ 1'(3,b)@[7]+ 1'(3,d)@[3]+ 1'(4,a)@[
| 3]+ 1'(4,c)@[2]
|
| Transport'AGVA_T 1: 1'Agv
|
```

Figura 4.11: Marcações correspondentes a informações de estado

Capítulo 5

O Supervisor Tolerante a Falhas

Neste capítulo, apresentamos o modelo do supervisor tolerante a falhas para o supervisor de SFM. Para isso, acrescentamos ao modelo apresentado no Capítulo 4, as mudanças necessárias para suportar tolerância a falhas em recursos. Conforme apresentado no Capítulo 3, abordaremos o tratamento de falhas através de redundância, promovendo o reescalonamento de peças para recursos de produção replicados e a geração de mensagens de erro para o ambiente do sistema. Os casos de falhas considerados são os discutidos na Seção 3.4 do Capítulo 3.

5.1 Hierarquia de Páginas do Modelo

Nesta seção, apresentamos a página que define a hierarquia do modelo. Na Figura 5.1 mostramos os retângulos arredondados representando as páginas para os módulos do sistema. Os retângulos *Hierarchy*, *Declarations*, *Cells*, *Snd_BufBuf*, *Send*, *Request*, *Answers*, *Receive*, *Transport*, *Verify_Net* já foram descritos no Capítulo 4. O retângulo *CHKBFREC* representa a página de verificação da operacionalidade dos recursos e reescalonamento de peças antes de enviar a peça para um recurso. O retângulo *CHKBFCEL* representa a página de verificação da operacionalidade dos recursos e reescalonamento de peças antes de enviar a peça para o *buffer* de entrada de um recurso. O retângulo *Fix* representa a página do modelo do Tratamento de Mensagens de Erro.

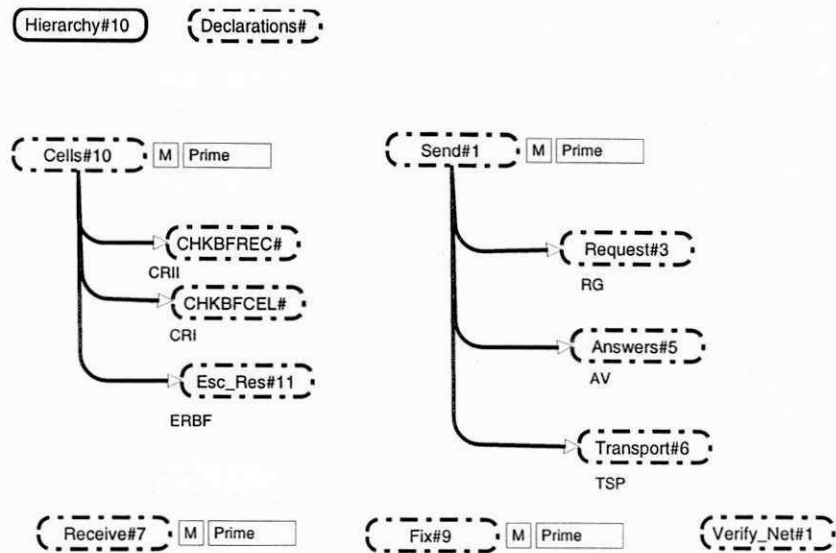


Figura 5.1: Hierarquia de páginas para o modelo do supervisor tolerante a faltas

5.2 O Supervisor Tolerante a Faltas

Na Figura 5.2 mostramos o modelo do supervisor para as células, modificado para suportar o tratamento de faltas em recursos. O lugar CIB_Cl possui o mesmo significado que o lugar CIB_C , e o lugar RIB_Cl possui o mesmo significado que o lugar RIB_C . As transições CRI e $CRII$ são transições de substituição e representam processos de verificação de *status* de recursos e procedimentos de reescalonamento de peças em caso de falha. A transição CRI representa a verificação de um recurso antes de uma peça ser transportada para o *buffer* de entrada desse recurso; e a transição $CRII$ representa a verificação de um recurso antes de uma peça obter esse recurso. As páginas associadas a essas transições são descritas nas seções seguintes.

Acrescentamos uma identificação interna para fazer a distinção entre dois ou mais recursos, no caso da célula possuir mais de um recurso do mesmo tipo. Para isso, fizemos as seguintes alterações à marcação inicial do modelo.

À marcação do lugar $RIBA_C$ incluímos uma cor para identificação de recurso. Ficando na forma:

$$M(RIBA_C) = (Cel, R, Replica)$$

1. Cel identifica a célula;

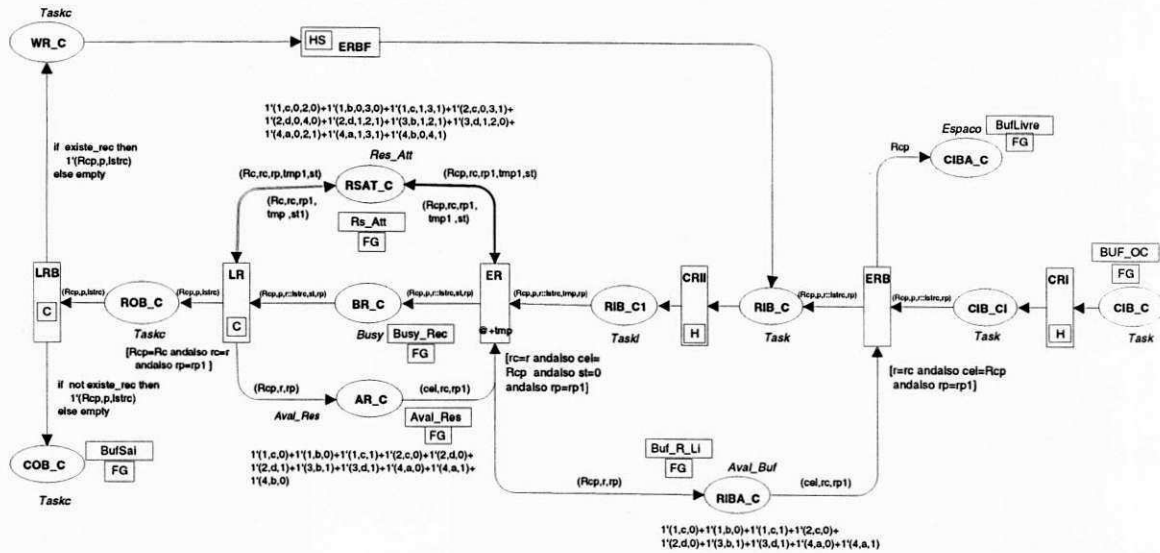


Figura 5.2: Supervisor Tolerante a Falhas

2. R identifica o tipo do recurso;
3. Replica é um identificador de recurso.

À marcação do lugar AR_C incluímos a mesma identificação interna de recurso, ficando na forma:

$M(AR_C) = (Cel, R, Replica)$, as variáveis possuem o mesmo significado do lugar RIBA_C.

À marcação do lugar RSAT_C foi acrescentado o *status* e a identificação interna de recurso, ficando na forma:

$$M(RSAT_C) = (Cel, R, Replica, tmpo, Status)$$

1. Cel identifica a célula;
2. R identifica o tipo do recurso;
3. Replica é um identificador de recurso;
4. tmpo identifica o tempo de processamento associado aquele recurso;
5. Status identifica o *status* do recurso.

A verificação do *status* do recurso permite implementar mecanismos para o supervisor identificar se houve falha e reescalonar as rotas se for necessário. A identificação interna do recurso permite a diferenciação entre dois recursos caso haja mais de um recurso do mesmo tipo dentro de uma mesma célula.

5.3 Verificação da Operacionalidade dos Recursos e Reescalonamento de Peças

A página associada à transição de substituição CRI, do modelo do supervisor tolerante a faltas (Figura 5.2), é mostrada na Figura 5.3. Esta rede modela a verificação do *status* do recurso antes de enviar a peça para o *buffer* de entrada da máquina e o reescalonamento das peças em caso de falha do recurso.

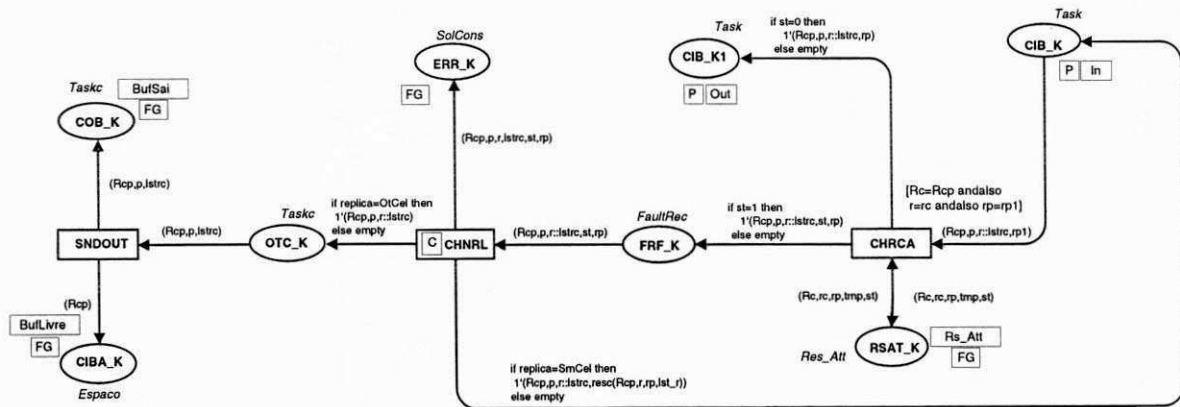


Figura 5.3: Verificação de *status* de recursos e reescalonamento de peças I

O significado dos lugares e transições são os seguintes:

Lugares:

CIB_K é lugar de fusão com CIB_C da Figura 5.2;

CIB_K1 é lugar de fusão com CIB_C1 da Figura 5.2;

RSAT_K é lugar de fusão com RS_AT da Figura 5.2;

CIBA_K é lugar de fusão com CIBA_C da Figura 5.2;

COB_K é lugar de fusão com COB_C da Figura 5.2;

Uma ficha FRF_K significa que foi identificada falha no recurso;

Uma ficha em OTC_K indica que a réplica do recurso se encontra em outra célula;

ERR_K representa o envio de uma mensagem de erro para o ambiente do sistema.

Transições:

CHRCA representa o procedimento de verificação do *status* do recurso;

CHNRL representa a verificação da localização da réplica em caso de falha do recurso e o envio de mensagens de erro para o ambiente do sistema;

SNDOUT representa o envio da peça para o *buffer* de saída da célula.

A ocorrência de CHRCA representa a verificação do *status* do recurso no lugar RSAT_K, antes da peça ser transportada para o *buffer* de entrada desse recurso. O recurso pode estar com *status* operacional ou em falha. Se o recurso estiver operacional, uma ficha é depositada no lugar CIB_K1, indicando que a peça está pronta para ser movida para o *buffer* deste recurso. Se o recurso estiver em falha, uma ficha é depositada no lugar FRF_K, habilitando a transição CHNRL, cuja ocorrência representa o envio de uma mensagem de erro para o ambiente do sistema, e a localização da réplica do recurso. Se a réplica estiver dentro da mesma célula, a ocorrência da transição CHNRL deposita uma ficha no lugar CIB_K, com a inscrição do arco modificada para a réplica do recurso; se a réplica estiver em outra célula, a transição CHNRL deposita uma ficha no lugar OTC_K habilitando a transição SNDOUT, cuja ocorrência representa o envio da peça para o *buffer* de saída da célula (lugar COB_K).

A página associada à transição de substituição CR11 do modelo do supervisor tolerante a faltas (Figura 5.2), é mostrada na Figura 5.4. Esta rede modela a verificação de faltas antes da peça obter o recurso e o reescalonamento das peças em caso de falha.

O significado dos lugares e transições são os seguintes:

Lugares:

RIB_H é lugar de fusão com RIB_C da Figura 5.2;

RIB_H1 é lugar de fusão com RIB_C1 da Figura 5.2;

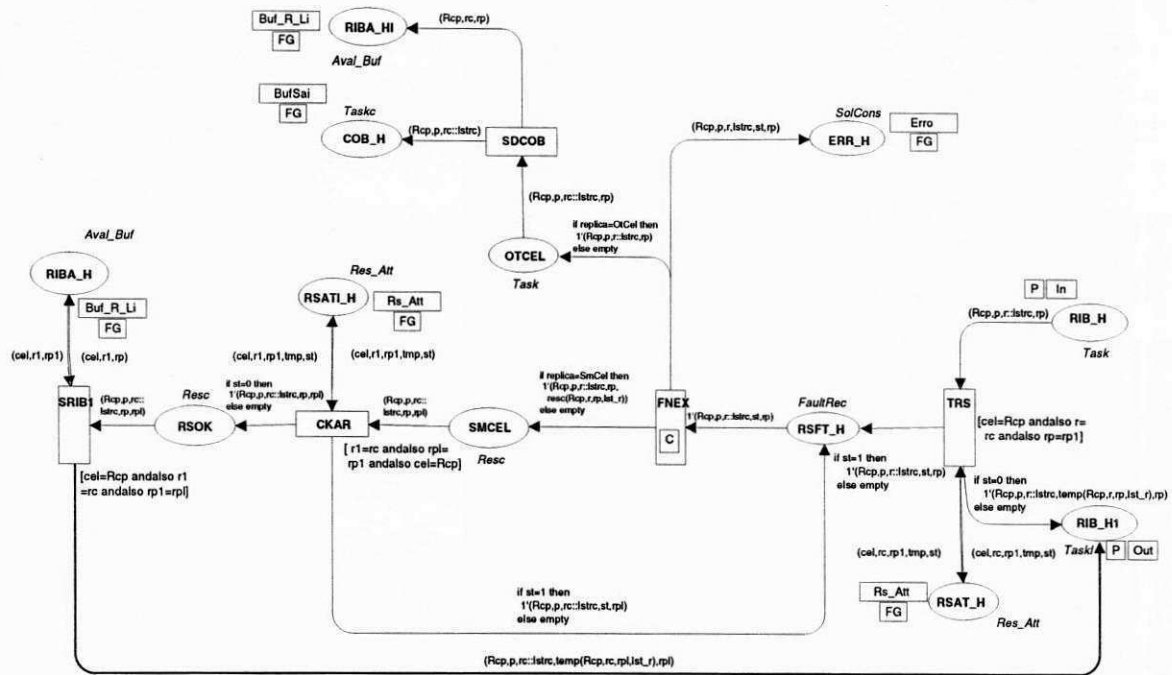


Figura 5.4: Verificação de *status* de recurso e reescalonamento de peças II

RSAT_H é lugar de fusão com RS_AT da Figura 5.2;

Uma ficha em RSFT_H significa que foi identificada falha no recurso;

Uma ficha em ERR_H modela o envio de uma mensagem de erro para o ambiente do sistema;

Uma ficha em SMCEL significa que a réplica do recurso encontra-se na mesma célula;

Uma ficha em OTCEL significa que a réplica do recurso encontra-se em outra célula;

Um ficha em RSOK significa que foi verificado o *status* de uma réplica dentro da mesma célula com retorno normal;

RIBA_H e RIBA_HI são lugares de fusão com RIBA_C da Figura 5.2;

COB_H é lugar de fusão com COB_C da Figura 5.2.

Transições:

TRS representa um procedimento de verificação do *status* de recursos;

FNEX representa a verificação da localização da réplica de recursos e o envio de mensagens de erro;

SDCOB representa o envio da peça para o *buffer* de saída da célula;

CKAR representa o procedimento de verificação de *status* da réplica do recurso;

SRIB representa o reescalonamento de uma peça para um outro recurso dentro da mesma célula.

A ocorrência de TRS representa a verificação do recurso antes da peça obtê-lo. Se o recurso estiver operacional, uma ficha é depositada em RIB_H1, e a peça pode ser transportada para o recurso. Se o recurso estiver em falha, uma ficha é depositada em RSFT_H habilitando a transição FNEX. A ocorrência de FNEX representa a verificação da localização da réplica e envia uma mensagem de erro para o ambiente do sistema. Se a réplica estiver em outra célula, uma ficha é depositada em OTCEL, habilitando a transição SDCOB, cuja ocorrência representa o envio da peça para o *buffer* de saída da célula.

Uma ficha é depositada em SMCEL se a réplica estiver na mesma célula, e a ocorrência de CKAR representa a verificação do *status* da réplica. Se a réplica estiver em falha, uma ficha é depositada no lugar RSFT_H, e o supervisor faz uma nova tentativa de reescalonamento. Caso contrário, uma ficha é depositada no lugar RSOK habilitando a transição SRIB, cuja ocorrência representa o reescalonamento da peça para o *buffer* da réplica.

5.4 Geração de Solicitações de Serviço

A página associada à transição de substituição RG do módulo de Envio do Protocolo de Reserva de Recurso (Figura 4.4), é mostrada na Figura 5.5. Esta rede modela a geração de solicitações de envio de peças, acrescentada das correspondentes alterações para promover o tratamento de faltas no sistema.

Foi acrescentado à rede o lugar TOI_RG, que modela a inicialização de um procedimento de temporização de espera por respostas de solicitações de serviço para as

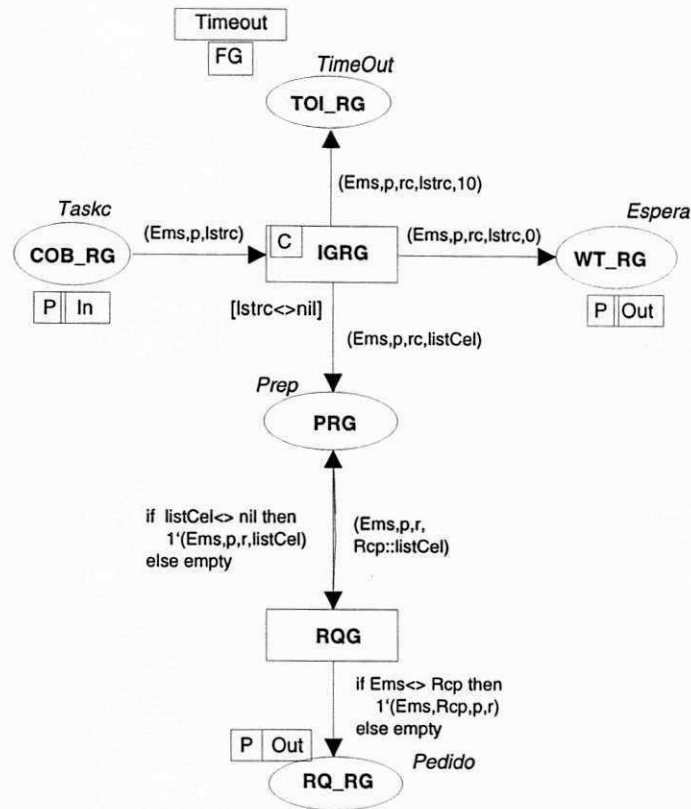


Figura 5.5: Geração de Solicitações de Serviço

células. A ocorrência da transição IGRG representa a recuperação de endereços de células para solicitação de serviço, e deposita uma ficha em TOI_RG.

5.5 Protocolo de Reserva de Recurso: Módulo de Recepção

A Figura 5.6 mostra o módulo de Recepção do Protocolo de Reserva de Recurso. Foram realizadas as seguintes alterações:

- Marcações dos lugares RSATI_R e RSAT_R são lugares de fusão com o lugar RS_AT já explicado na Seção 5.2 e;
- Guarda da transição ANS, que agora verifica o *status* dos recursos para enviar as respostas. A resposta só são enviadas quando há espaço disponível no *buffer* da célula e o recurso solicitado está operacional (*status=0*).

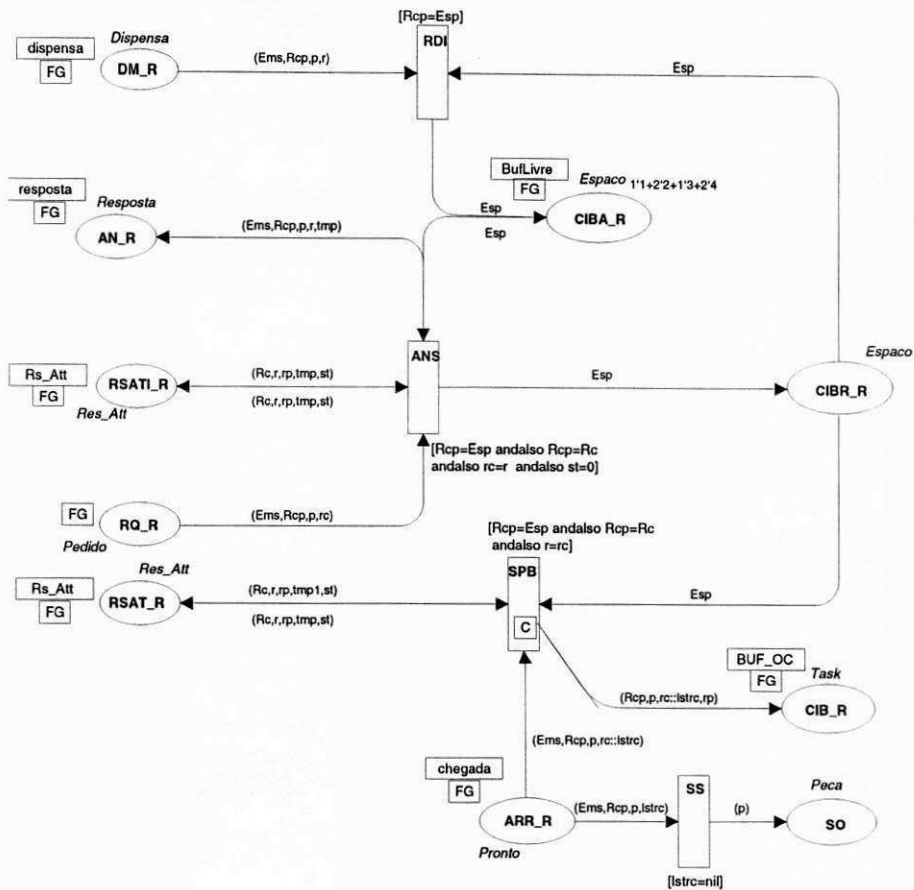


Figura 5.6: Protocolo de Reserva de Recurso: Módulo de Recepção

5.6 Tratamento de Respostas

A página associada à transição de substituição AV, do módulo de Envio do Protocolo de Reserva de Recurso (Figura 4.4), é mostrada na Figura 5.7. Esta rede modela o tratamento das respostas recebidas pelo supervisor solicitante. As alterações acrescentadas ao modelo permitem a reinicialização do protocolo mesmo que nenhum supervisor responda, isto é, mesmo que supervisores solicitados ou recursos do tipo requerido falhem. Foram realizadas as seguintes alterações:

Lugares:

TOI_V é lugar de fusão com TOI_RG da Figura 5.5;

Uma ficha em TOF_V modela um atraso que representa um estado de espera;

Uma ficha em ERRMS_V modela o envio de uma mensagem de erro para o ambiente do sistema.

5.7 Tratamento de Mensagens de Erro

Todas as mensagens de erro produzidas pelo SFM são tratadas no modelo mostrado na Figura 5.8. O lugar ERRMS_F modela o recebimento de todas as mensagens de erro provenientes do SFM as quais podem ser originadas do módulo de Tratamento de Respostas, Figura 5.7, Seção 5.6, ou dos módulos de Verificação da Operacionalidade dos Recursos e Reescalonamento de Peças (transições CRI e CRII do módulo de envio do Protocolo de Reserva de Recurso), Figuras 5.3 e 5.4, Seção 5.3.

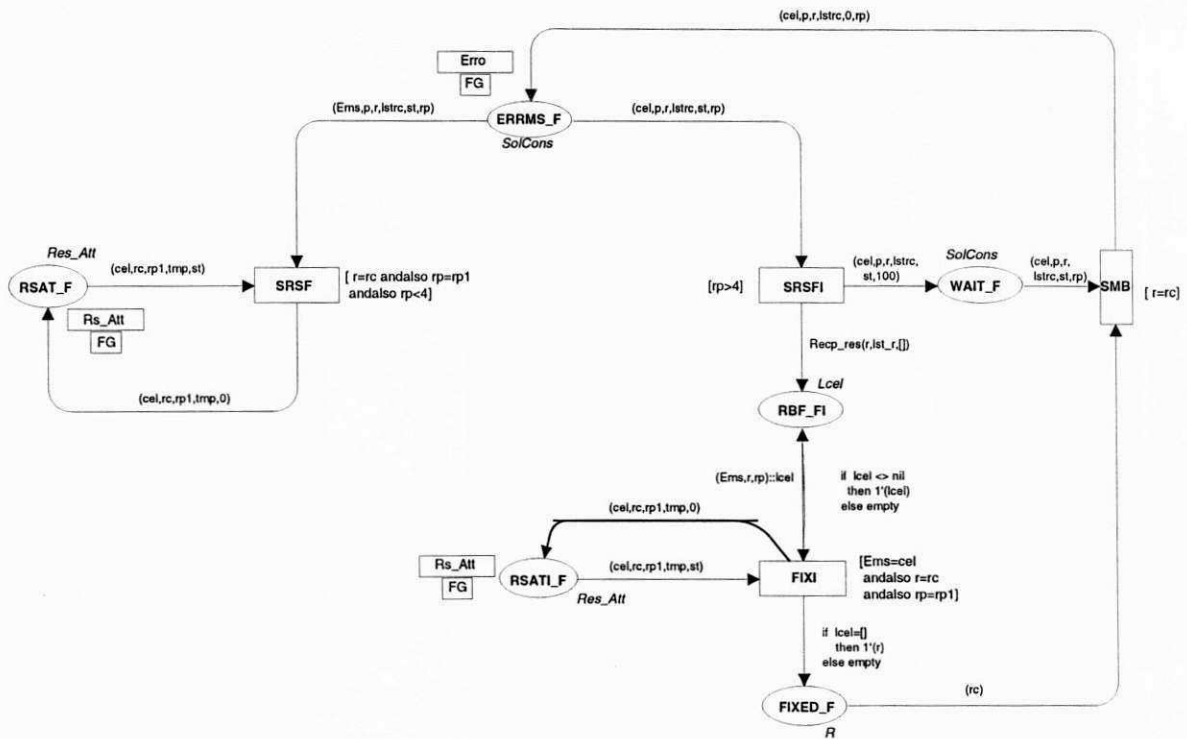


Figura 5.8: Modelo de Tratamento de Mensagens de Erro

O significado dos lugares e transições são os seguintes:

Lugares:

Uma ficha em ERRMS_F modela o recebimento de mensagens de erro do SFM e é lugar de fusão com os lugares ERR_H da Figura 5.4, ERR_K da Figura 5.3 e ERRMS_V da Figura 5.7;

RSAT_F e RSATI_F são lugares de fusão com RS_AT da Figura 5.2;

WAIT_F modela um procedimento de espera pelo reparo de recursos;

RBF_FI modela solicitações de reparo de recursos;

FIXED_F modela o estado quando um recurso foi reparado.

Transições:

SRSFI representa a geração de uma solicitação de reparo para todos os recursos de um determinado tipo;

FIXI representa o reparo de um recurso em decorrência de uma mensagem de erro gerada por um dos módulos associados às transições CRI e CRII, ver Seção 5.3;

SRSF representa o reparo de um recurso em decorrência do recebimento de uma mensagem de erro gerada pelo módulo de Envio do Protocolo de Reserva de Recurso, ver Seção 5.6.

Caso a mensagem seja originada do módulo de Tratamento de Respostas, significa que um supervisor fez uma solicitação de envio de peça e não obteve nenhuma resposta após um certo tempo, impedindo a continuação do processo de manufatura. Isto indica que houve falha em supervisores ou recursos de um determinado tipo no SFM. Neste caso, a ocorrência da transição SRSF representa a inicialização de um procedimento de reparo para os componentes que falharam. A ocorrência da transição FIXI representa o reparo dos componentes em falha. Quando os componentes são restaurados, uma ficha é depositada no lugar FIXED_F, habilitando a transição SMB cuja ocorrência representa o envio de uma mensagem para o módulo de Tratamento de Respostas, Figura 5.7, notificando ao supervisor que o problema foi resolvido.

Se a mensagem for originada dos módulos de Verificação e Operacionalidade de Recursos, Figuras 5.3 e 5.4 Seção 5.3, então somente o reparo do recurso identificado na mensagem de erro recebida é levado em conta. A ocorrência da transição SRSF representa o processo de reparo do recurso.

5.8 Nó de Declaração

A seguir mostramos o nó de declaração que define os conjuntos de cores, as variáveis e funções utilizadas no modelo.

```

val lista_a = [4];
val lista_b = [1,3];
val lista_c = [1,2];
val lista_d = [2,3];
val SmCel = 0;
val OtCel = 1;
color existerec= bool;
color AGV = with Agv;
color Peca = with Belt|Purse|Wallet timed;
color INT = int;
color Emissor = INT;
color Receptor = INT;
color tmpo = INT;
color Espaco = INT;
color Cel = INT;
color Status = INT;
color Replica = INT;
color Code = INT;
color R = with a|b|c|d timed;
color Rp = with S|N;
color Celulas = list Cel;
color Tab = product R*Celulas;
color lstRc = list R;
color ResId = product Cel*R*Replica;
color Lcel = list ResId;
color Inicio = product Peca*lstRc;
color Taskc = product Emissor*Peca*lstRc;
color Espera = product Emissor*Peca*R*lstRc*Code;
color Pedido = product Emissor*Receptor*Peca*R;
color Prep = product Emissor*Peca*R*Celulas;
color PrepR = product Emissor*Receptor*Peca*R*lstRc*tmpo;
color tipo = product Emissor*Receptor*Peca*R*lstRc;
color Task = product Receptor*Peca*lstRc*Replica;
color Taskl = product Receptor*Peca*lstRc*tmpo*Replica;
color Taskll = product Receptor*Peca*R*lstRc;
color Resc = product Receptor*Peca*lstRc*Replica*Replica;
color Busy = product Receptor*Peca*lstRc*Status*Replica;
color FaultRec = product Receptor*Peca*lstRc*Status*Replica;
color SolCons = product Receptor*Peca*R*lstRc*Status*Replica;
color Pronto = product Emissor*Receptor*Peca*lstRc;
color Resposta = product Emissor*Receptor*Peca*R*tmpo;
color TimeOut = product Emissor*Peca*R*lstRc*tmpo;
color lstRp = list Resposta;
color Crmd = product Emissor*Receptor*Peca*R*lstRp*Celulas;
color Dispensa = product Emissor*Receptor*Peca*R;
color Ped_Agv = product Emissor*Receptor*Peca;
color Chegada = product Emissor*Receptor*Peca*AGV*lstRc;
color Agv_Env = product Emissor*Receptor*Peca*AGV;
color Aval_Res = product Cel*R*Replica;
color Aval_Buf = product Cel*R*Replica;
color Res_Att = product Receptor*R*Replica*tmpo*Status;
var lstrp : lstRp;
var tt : Resposta;
var lcel : Lcel;
var tmp,tmp1,tp,Rcp,Rc,cel,st,st1,rp,rp1,rpl,replica : INT;
var cod : Code;
var p,pc,p1 : Peca;
var lstrc,lstrc1: lstRc;
var Esp,Espc : Espaco;
var r,rc,r1 : R;
var Ems,Em,Em1 : Emissor;

```

```

var listCel : Celulas;
var existe_rec : existerec;
var agv : AGV;
val lst_r = [(1,c,0,2),(1,b,0,3),(1,c,0,3),(2,c,0,3),(2,d,0,4),(2,d,1,2),
(3,b,1,2),(3,d,1,2),(4,a,0,2),(4,a,1,3),(4,b,0,4)];

(* Recupera a informação de tempo de processamento para o recurso *)
fun temp(Rcp,r,rp,(Rc,rc,rp1,tp)::y)=
if (Rcp=Rc andalso r=rc andalso rp=rp1)
then tp
else temp(Rcp,r,rp,y);

(* Atualiza a informação de tempo de atendimento para um recurso, adicionando *)
fun calc_temp(Rcp,r,rp,tmp:INT,(Rc,rc,rp1,tp)::y)=
if (Rcp=Rc andalso r=rc andalso rp=rp1)
then tmp+tp
else calc_temp(Rcp,r,rp,tmp,y);

(* Atualiza a informação de tempo de atendimento para um recurso, subtraindo *)
fun calc_temp2(Rcp,r,rp,tmp:INT,(Rc,rc,rp1,tp)::y)=
if (Rcp=Rc andalso r=rc andalso rp=rp1)
then tmp-tp
else calc_temp2(Rcp,r,rp,tmp,y);

(* Verifica a existência de um recurso numa célula *)
fun Existe_rec(Rcp,r::lstrc,[]) = false|
Existe_rec(Rcp,r::lstrc,(Rc,rc,rp,tp)::y) =
if (Rcp=Rc andalso r=rc) then
true
else Existe_rec(Rcp,r::lstrc,y);

(* Recupera a localização de um recurso para fazer o reescalonamento *)
fun Next_rec(Rc,r,rp,(Rcp,rc,rp1,tp)::y)=
if (Rc=Rcp andalso r=rc andalso rp<>rp1)
then 0
else if (Rc<>Rcp andalso r=rc)
then 1
else Next_rec(Rc,r,rp,y);

(* Atualiza a informação da ficha para executar o reescalonamento *)
fun resc(Rc,r,rp,(Rcp,rc,rp1,tp)::y)=
if (Rc=Rcp andalso r=rc andalso rp<>rp1)
then rp1
else resc(Rc,r,rp,y);

(* Recupera a informação da localização de recursos de um determinado
tipo para o ambiente do sistema *)
fun Recp_res(r,[],lcel)=lcel|
Recp_res(r,(Rcp,rc,rp1,tp)::y,lcel)=
if (r=rc)
then Recp_res(r,y,(Rcp,rc,rp1)::lcel)
else
Recp_res(r,y,lcel);

(* Recupera a lista de células que possui um determinado recurso *)
fun acha_lista(rc::lstrc) =
if (rc=a) then
(rc,lista_a)
else if (rc=b) then
(rc,lista_b)
else if (rc=c) then

```

```

(rc,lista_c)
else if (rc=d) then
(rc,lista_d)
else (rc,[]);

(* Verifica se um pedido foi enviado para um determinado supervisor *)
fun sent(Rcp,[]) = false|
sent(Rcp,(Em,Rc,p,r,tmp)::y)=
if Rc = Rc then
true
else sent(Rcp,y);

```

5.9 Análise do Modelo

A análise do modelo foi feita com base no mesmo método discutido no Capítulo 4, considerando os seguintes casos de falhas já discutidos na Seção 3.4 do Capítulo 3.

5.9.1 Situações 1 e 2

- 1 Existem ordens de produção na entrada do sistema ou *buffer* de saída de uma célula e o supervisor solicitado ou uma das réplicas do próximo recurso de produção falharam.
- 2 Existem ordens de produção na entrada do sistema ou *buffer* de saída de uma célula e os supervisores solicitados ou todas as réplicas do próximo recurso solicitado falharam.

Para as situações 1 e 2 geramos um só grafo de ocorrência. A análise foi feita a partir da seguinte marcação inicial:

$$M(B_S) = 1'(Piece1, [a,c,]) + 1'(Piece2,[b,d])$$

Sendo que as réplicas dos recursos dos tipos c, d, b, e todos os recursos do tipo a estavam em estado de falha.

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 537

Arcs: 1104

Secs: 46

Status: Full

Indicando que a geração foi completa com 537 nós e 1104 arcos. Na Figura 5.9 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO (Saída do Sistema, ver Seção 4.5) e B_S (Entrada do Sistema, ver Seção 4.3) indicando duas ordens de produção na entrada do sistema e vazio na saída.

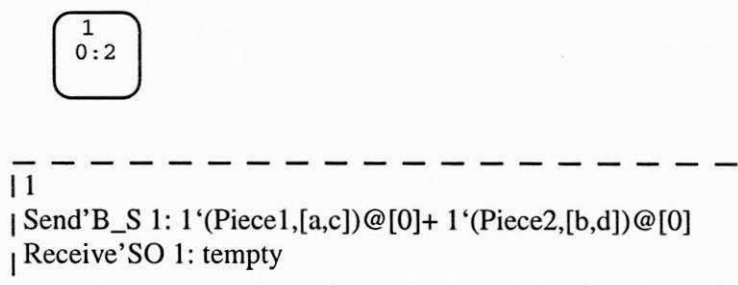


Figura 5.9: Marcação inicial para as situações de falha 1 e 2

Na Figura 5.10 exibimos a lista de marcações mortas encontradas. Podemos também observar na Figura 5.10, que em todos os estados finais encontrados, o conteúdo do lugar SO corresponde às tarefas completadas. Por exemplo, no nó 537 temos a marcação $1'Piece1@[17] + 1'Piece2@[7]$, indicando que as peças Piece1 e Piece2 foram processadas em 17 e 7 unidades de tempo respectivamente.

Podemos observar que as tarefas foram concluídas mesmo para o pior caso (falha de todos os recursos de um mesmo tipo). Entretanto, o tempo levado para manufaturar as peças foi maior em relação aos tempos encontrados na análise feita no Capítulo 4. Isto era esperado, já que para este caso, as tarefas tiveram que ser completadas com apenas parte da capacidade do sistema.

5.9.2 Situação 3

Houve falha no recurso e peças estavam armazenadas no *buffer* da célula aguardando para serem enviadas para o *buffer* do recurso.

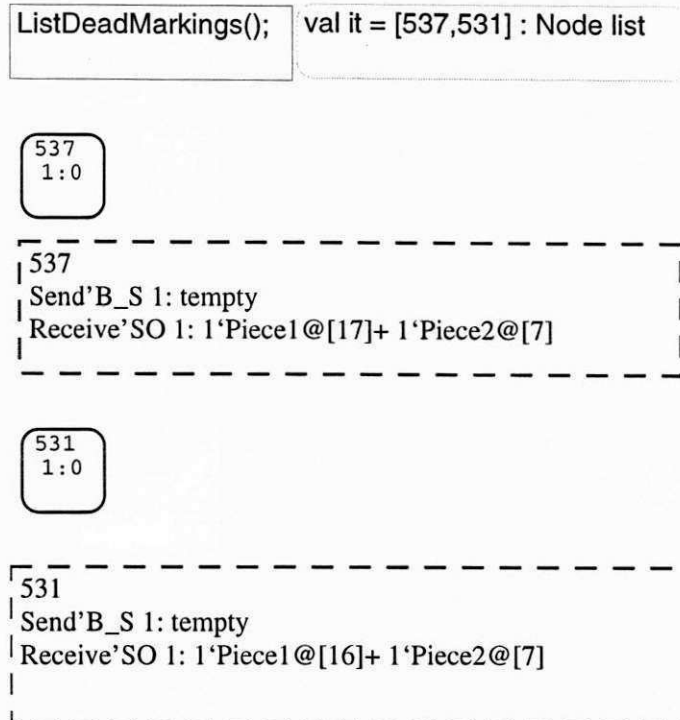


Figura 5.10: Estados finais para as situações de falha 1 e 2

Neste caso, as peças são reescaloadas para a respectiva réplica do recurso que pode estar:

- 3.a) Na mesma célula.
- 3.b) Em outra célula.

Para a situação 3.a, nossa análise foi feita a partir da marcação inicial:

$$M(\text{CIB_C}) = 1'(2, \text{Piece}, [d], 1)$$

Indicando uma peça no *buffer* da célula 2 escalonada para usar um recurso do tipo *d* cuja identificação de réplica é 1. Sendo que este recurso está em estado de falha.

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 19

Arcs: 22

Secs: 1

Status: Full

Indicando que a geração foi completa, com 19 nós e 22 arcos. Na Figura 5.11 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO e B_S, indicando duas ordens de produção na entrada do sistema e vazio na saída.

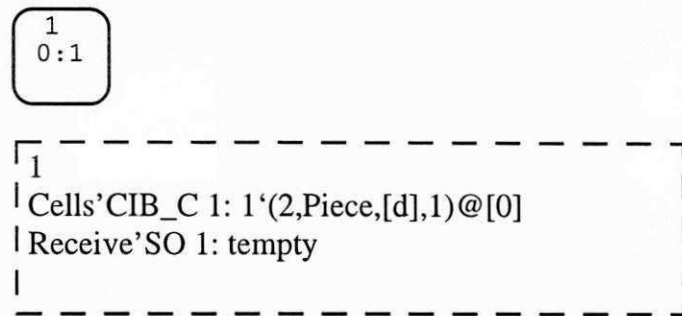


Figura 5.11: Marcação inicial para a situação de falha 3.a

Na Figura 5.12 exibimos a lista de marcações mortas encontradas. Podemos também observar na Figura 5.12, que no estado final encontrado, o conteúdo do lugar SO corresponde à tarefa completada. O nó 19 tem a marcação 1'Piece@[4], indicando que a peça Piece foi processada em 4 unidades de tempo.

É importante observar que para esta situação e para as próximas que serão descritas, o tempo de processamento indicado na marcação final da rede não corresponde ao tempo de processamento total desde a chegada da ordem de produção no sistema até a saída. Isto porque devido ao problema de explosão de estados durante a geração dos grafos de ocorrência para estas situações, a análise foi feita criando-se diversos cenários de falta, colocando-se a marcação inicial, para efeito de geração do grafo de ocorrência, no estado imediatamente anterior à detecção da falta.

No caso da situação 3, as marcações iniciais correspondentes às ordens de produção foram colocadas no lugar COB_C, que corresponde ao *buffer* de entrada da célula.

Para a situação 3.b, nossa análise foi feita a partir da marcação inicial:

$$M(\text{CIB_C}) = 1'(3,\text{Piece},[b],1)$$

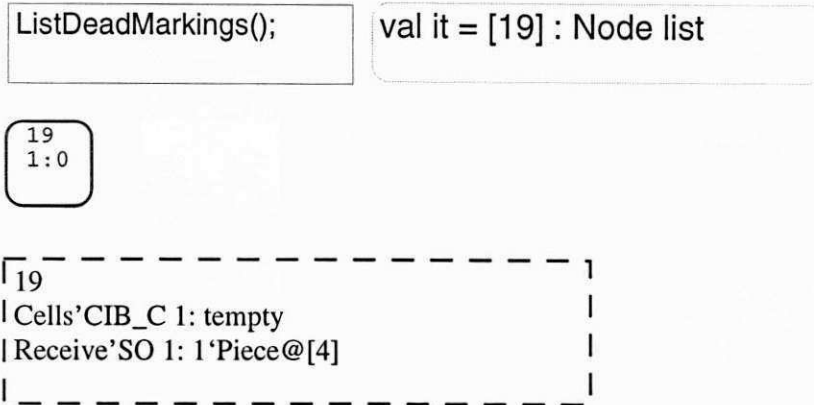


Figura 5.12: Estado final para a situação de falha 3.a

Indicando uma peça no *buffer* da célula 3 escalonada para usar um recurso do tipo b cuja identificação de réplica é 1. Sendo que este recurso está em estado de falha.

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 80

Arcs: 144

Secs: 3

Status: Full

Indicando que a geração foi completa, com 80 nós e 144 arcos. Na Figura 5.13 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO e B_S, indicando duas ordens de produção na entrada do sistema e vazio na saída.

Na Figura 5.14 exibimos a lista de marcações mortas encontradas. Podemos também observar na Figura 5.14, que no estado final encontrado, o conteúdo do lugar SO corresponde à tarefa completada. O nó 80 tem a marcação 1'Piece@[4], indicando que a peça Piece foi processada em 4 unidades de tempo.

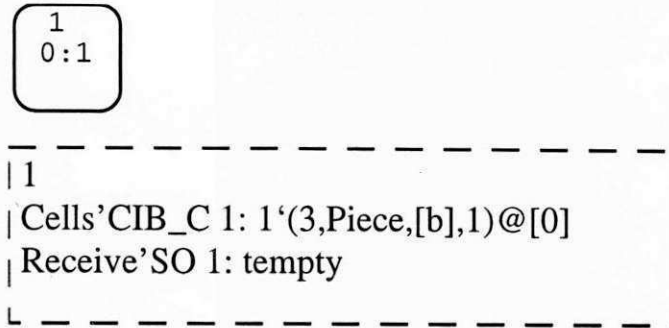


Figura 5.13: Marcação inicial para a situação de falha 3.b

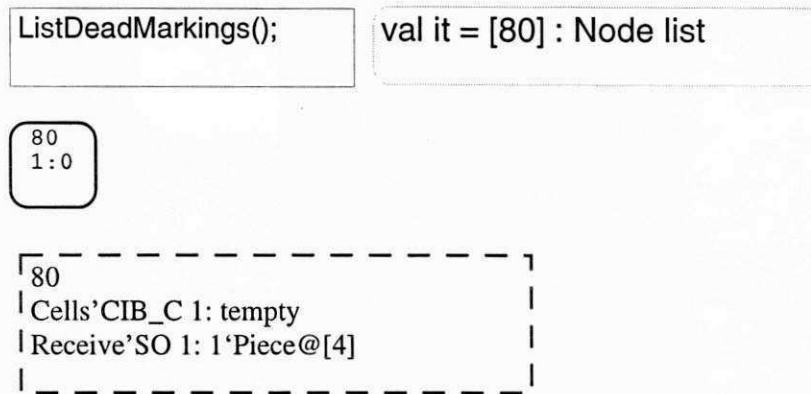


Figura 5.14: Estado final para a situação de falha 3.b

5.9.3 Situação 4

Houve falha no recurso e peças estavam armazenadas no *buffer* deste, aguardando para serem atendidas.

Neste caso, as peças são reescaloadas para a respectiva réplica do recurso que pode estar:

- 4.a) Na mesma célula.
- 4.b) Em outra célula.

Para a situação 4.a, nossa análise foi feita a partir da marcação inicial:

$$M(\text{RIB_C}) = 1'(2, \text{Piece}, [d], 1)$$

Indicando uma peça na célula 2, no *buffer* de um recurso do tipo d cuja identificação de réplica é 1. Sendo que este recurso está em estado de falha.

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 17

Arcs: 19

Secs: 1

Status: Full

Indicando que a geração foi completa, com 17 nós e 19 arcos. Na Figura 5.15 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO e B_S, indicando duas ordens de produção na entrada do sistema e vazio na saída.

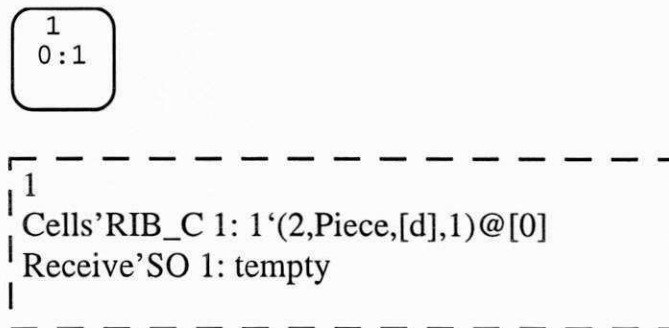


Figura 5.15: Marcação inicial para a situação de falha 4.a

Na Figura 5.16 exibimos a lista de marcações mortas encontradas. Podemos também observar na Figura 5.16, que no estado final encontrado, o conteúdo do lugar SO corresponde à tarefa completada. O nó 17 tem a marcação 1'Piece@[4], indicando que a peça Piece foi processada em 4 unidades de tempo.

O tempo de processamento indicado na marcação final da rede, não corresponde ao tempo de processamento total desde a entrada do sistema até a saída, pelo mesmo motivo já explicado na Seção 5.9.2. Deste modo, para a geração do grafo de ocorrência, às marcações iniciais correspondentes às ordens de produção foram colocadas no lugar ROB_C, que corresponde ao *buffer* de entrada da máquina.

Para a situação 4.b, nossa análise foi feita a partir da marcação inicial:

$$M(\text{RIB_C}) = 1'(3, \text{Piece}, [b], 1)$$

```
ListDeadMarkings();
```

```
val it = [17] : Node list
```

```
17  
1:0
```

```
17  
| Cells'RIB_C 1: tempty  
| Receive'SO 1: 1'Piece@[4]
```

Figura 5.16: Estado final para a situação de falha 4.a

Indicando uma peça na célula 3, no *buffer* de um recurso do tipo b cuja identificação de réplica é 1. Sendo que este recurso está em estado de falha.

Após a geração do grafo, obtivemos as seguintes estatísticas:

Statistics

Occurrence Graph

Nodes: 80

Arcs: 144

Secs: 3

Status: Full

Indicando que a geração foi completa, com 80 nós e 144 arcos. Na Figura 5.17 mostramos o primeiro nó do grafo de ocorrência denotando a marcação inicial dos lugares SO e B_S, indicando duas ordens de produção na entrada do sistema e vazio na saída.

```
1  
0:1
```

```
1  
| Receive'SO 1: tempty  
| Cells'RIB_C 1: 1'(3,Piece,[b],1)@[0]
```

Figura 5.17: Marcação inicial para a situação de falha 4.b

Na Figura 5.18 exibimos a lista de marcações mortas encontradas. Podemos também observar na Figura 5.18, que no estado final encontrado, o conteúdo do lugar SO corresponde à tarefa completada. O nó 80 tem a marcação 1'Piece@[4], indicando que a peça Piece foi processada em 4 unidades de tempo.

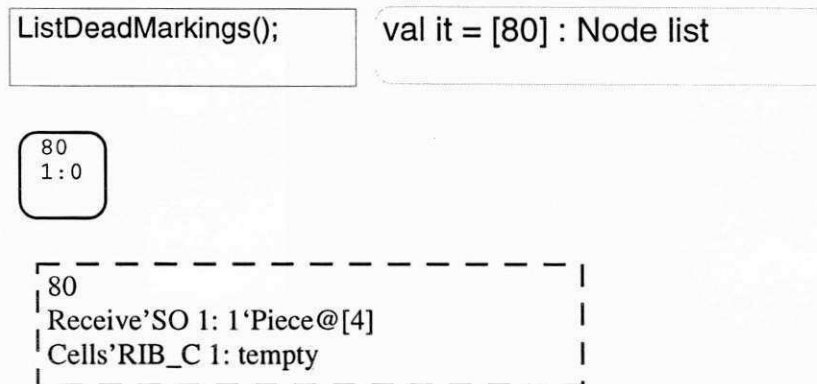


Figura 5.18: Estado final para a situação de falha 4.b

Pudemos observar que para todas as situações de falha consideradas, foi possível para o sistema completar suas tarefas, indicando que o modelo é vivo em todos os estados, com exceção dos nós referentes aos estados finais, e a propriedade de reinicialização foi preservada.

5.10 Conclusão

O objetivo deste capítulo foi apresentar a modelagem, em redes de Petri, para o Supervisor Distribuído Tolerante a Faltas para um SFM. Para tanto, foram adicionados ao modelo, apresentado no Capítulo anterior, as mudanças necessárias para suportar a ocorrência de falhas em recursos.

Iniciamos o capítulo com a descrição da página que define a hierarquia do modelo, seguida das redes de Petri modeladas, com a descrição dos lugares, transições e comportamento do modelo. Por fim mostramos os resultados da análise. Observamos que em todos os casos de falha considerados, o sistema completou suas tarefas preservando as propriedades encontradas no modelo não tolerante a faltas.

Capítulo 6

Conclusão

Neste trabalho, para promover a coordenação de eventos de um SFM de forma distribuída, desenvolvemos modelos para atribuir um supervisor para cada célula do SFM, que coordenam os eventos internos de uma célula à qual são associados, e interagem de modo cooperativo para promover a produção. Vimos que a abordagem de supervisores distribuídos é interessante no sentido que promove uma maior escalabilidade ao sistema, simplifica a gerência de recursos através da coordenação local de eventos, e torna mais natural a adição de tolerância a faltas aos supervisores. Essa abordagem é relevante no contexto de SFM por que remove a atribuição da tarefa de supervisão a um só componente centralizado.

Os modelos que apresentamos foram construídos em redes de Petri coloridas através da utilização da ferramenta Design/CPN [26], que permite a edição, simulação e análise formal de redes de Petri Coloridas, Hierárquicas e Temporizadas.

O trabalho foi desenvolvido em quatro etapas. Primeiramente, estudamos os conceitos básicos referentes a SFM e redes de Petri. Em seguida definimos a solução distribuída para o supervisor a ser modelado. Depois elaboramos os modelos para o supervisor em redes de Petri coloridas. Por fim, acrescentamos as mudanças necessárias nos modelos para suportar tolerância a faltas em recursos.

Os sistemas modelados foram apresentados por completo com as respectivas descrições dos significados dos lugares, transições e marcações iniciais.

No final de cada fase de modelagem, exibimos os resultados obtidos da análise. Para isso, obtivemos os grafos de ocorrência no Design/CPN, e verificamos as propriedades

requeridas nos modelos. Verificamos as seguintes propriedades:

- Os modelos são vivos em todos os estados, não apresentando situações de bloqueio indesejadas;
- Os modelos são reinicializáveis. Isto significa que, após a execução de uma tarefa, o sistema está pronto para executar outra.

O ambiente para suporte à comunicação distribuída não foi discutido. Entretanto, é importante ressaltar a necessidade de um ambiente adequado, como uma rede de comunicação, com seus respectivos protocolos de comunicação e outros componentes para permitir a troca de mensagens entre os supervisores do SFM.

Ao longo do desenvolvimento do trabalho, sentimos uma dificuldade no desenvolvimento dos modelos com a ferramenta Design/CPN, que ainda possui alguns problemas a serem solucionados. Como exemplo, podemos citar a falta de clareza nos manuais de operação e nas mensagens de erro, e as constantes ocorrências de danos nos dados dos arquivos que nos obrigavam a reeditar os modelos. Contudo, devemos ressaltar que, devido ao tamanho dos modelos e dos espaços de estado gerados, teria sido impraticável sua implementação e análise sem a ferramenta Design/CPN.

Considerando as discussões anteriores, em que afirmamos que abordagens descentralizadas têm sido largamente investigadas em projetos de supervisores para SFM, e considerando as redes de Petri, com sua fundamentação matemática consolidada, e adequação para modelar características de sistemas a eventos discretos, podemos afirmar que as pesquisas na área de especificação de supervisores para SFM, utilizando redes de Petri, trazem uma contribuição válida para a atual tendência de evolução desses tipos de supervisores.

Alguns aspectos não foram discutidos neste trabalho. Por exemplo, o comportamento dos AGVs e o transporte de peças dentro das células. Seria interessante em trabalhos futuros, investigar os detalhes do sistema de transporte como o comportamento de robôs para movimentação de peças dentro de células, políticas de escalonamento de AGVs, detalhamento das rotas de passagem, definição de pontos de parada, tratamento de colisão e outros.

Outra possibilidade para investigações futuras, seria considerar estimativas de tempo para todas as operações e avaliar diferentes políticas de escalonamento.

Bibliografia

- [1] J. Angelva and P. Piltonen. Distributed real-time data management in a flexible manufacturing system (FMS). In R. S. Sodhi, editor, *Advances in Manufacturing Systems*, pages 81–86. Elsevier Science B. V., 1994.
- [2] T. C. Barros. Especificação do supervisor de sistemas de produção baseada em redes de petri. Relatório Técnico, Universidade Federal da Paraíba, 1998.
- [3] T.C. Barros and A. Perkusich. Design of supervisors for agile manufacturing systems using colored petri nets. In *Proc. of 15th International Conference on CAD/CAM Robotics & Factories of the Future, CARS & FOF'99*, pages MF1–1–MF1–6, Águas de Lindóia, SP, Brazil, August 1999.
- [4] T.C. Barros, A. Perkusich, and J.C.A. de Figueiredo. Alocação de recursos e tolerância a faltas em sistemas flexíveis de manufatura utilizando redes de petri coloridas. In *Anais do VII Simpósio Brasileiro de Computadores Tolerantes a Falhas*, pages 153–167, Campina Grande, PB, Brasil, July 1997.
- [5] T.C. Barros, A. Perkusich, and J.C.A. de Figueiredo. A coloured petri net based approach for resource allocation and fault tolerance for flexible manufacturing systems. In *Proc. of 2nd Workshop on Manufacturing and Petri Nets, 18th Int. Conference on Application and Theory of Petri Nets*, pages 77–96, Toulouse, France, June 1997.
- [6] T.C. Barros, A. Perkusich, and J.C.A. de Figueiredo. A fault tolerant coloured petri net resource allocation manager for manufacturing systems. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 1210–1215, Orlando, USA, October 1997.

- [7] T.C. Barros, A. Perkusich, and J.C.A. de Figueiredo. Modelagem do controle dos sistemas flexíveis de manufatura baseada em redes de petri coloridas. In *Anais do XII Congresso Brasileiro de Automática*, pages 1403–1408, Uberlândia, Brasil, September 1998.
- [8] T.C. Barros, A. Perkusich, and J. C. A. Figueiredo. A fault tolerant colored petri net model for flexible manufacturing systems. *Journal of the Brazilian Computer Society*, pages 16–30, 1997.
- [9] B. Bona, P. Brandimarte, C. Greco, and G. Menga. Hybrid hierarchical scheduling and control systems in manufacturing. *IEEE Transactions on Robotics and Automation*, 6(6), December 1990.
- [10] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley Publishing Company, 1967.
- [11] A. A. Desrochers and R. Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems, Modeling, Control, and Performance Analysis*. IEEE Press, 1995.
- [12] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman & Hall, 1993.
- [13] R. C. Dorf and A. Kusiak. *Handbook of Design, Manufacturing and Automation*. John Wiley & Sons, Inc., 1994.
- [14] M. P. Fanti, B. Maione, and B. Turchiano. Event control for deadlock avoidance in assembly systems. *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics*, 4:3756–3761, October 1997.
- [15] L. Ferrarini and M. Maroni. A control algorithm for deadlock-free scheduling of manufacturing systems. *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics*, 4:3762–3767, October 1997.
- [16] J. C. A. Figueiredo. *Rede de Petri com Temporização Nebulosa*. Tese de Doutorado, Universidade Federal da Paraíba, 1994.

- [17] N. Gayed, D. H. Jarvis, and J. H. Jarvis. A strategy for the migration of existing manufacturing systems to holonic systems. In *Proc. Of IEEE Int. Conf. On Systems Man and Cybernetics*, pages 319–324, 1998.
- [18] D. D. S. Guerrero. Sistemas de redes de petri modulares baseados em objetos. Dissertação de Mestrado, Universidade Federal da Paraíba, 1997.
- [19] K. Hasegawa, M. Sugisawa, and Z. A. Banaszak. Graphical analysis and synthesis of deadlocks avoidance in flexible manufacturing systems. In *First International Workshop on Manufacturing and Petri Nets*, pages 161–176. Osaka, Japão, June 1996.
- [20] J. S. Huang and T. Murata. Classifications of petri nets transitions and their applications to firing sequence and reachability problems. *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics*, 1:263–268, October 1997.
- [21] P. Jalote. *Fault Tolerance in Distributed Systems*. P T R Prentice Hall, 1994.
- [22] M. D. Jeng and C. S. Lin. Scheduling flexible manufacturing systems containing assembly operations based on petri net structures and dynamics. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 4430–4435, 1997.
- [23] K. Jensen. *Colored Petri Nets, basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer-Verlag, 1992.
- [24] K. Jensen. *Coloured petri nets-Basic Concepts, Analysis Methods and Practical Use*, volume 2. Springer-Verlag, 1997.
- [25] K. Jensen. An introduction to the practical use of coloured petri nets. In *Lectures on Petri Nets II: Applications*. Springer, 1998.
- [26] K. Jensen and et. al. *Design/CPN Manuals*. Meta Software Corporation and Department of Computer Science, University of Aarhus, Denmark. On-line version:<http://www.daimi.aau.dk/designCPN/>.
- [27] N. N. Krishnamurthy, R. Batta, and M. H. Karwan. Developing conflict-free routes for automated guided vehicles. *Operations Research*, pages 1077–1090, 1993.

- [28] D. Y. Lee and F. DiCesare. Petri net-based heuristic scheduling for flexible manufacturing. In MengChu Zhou, editor, *Petri Nets in Flexible and Agile Automation*, pages 149–187. KAP, 1995.
- [29] D. Y. Lee and DiCesare F. Integrated scheduling of flexible manufacturing systems employing automated guided vehicles. *IEEE Transactions on Industrial Electronics*, pages 602–610, 1994.
- [30] S. Levi and A. K. Agrawala. *Fault Tolerant System Design*. McGraw-Hill, 1994.
- [31] S. Li, T. Takamori, and T. Satoshi. Scheduling and re-scheduling of agvs for flexible manufacturing. In MengChu Zhou, editor, *Petri Nets in Flexible and Agile Automation*, pages 189–205. KAP, 1995.
- [32] P. B. Luh, D. J. Hoiomt, E. Max, and Pattipati K. R. Schedule generation and reconfiguration for parallel machines. *IEEE Transactions on Robotics and Automation*, 6(6), December 1990.
- [33] J. E. Luntz and W. Messner. A distributed control system for flexible materials handling. *IEEE Control Systems*, pages 22–28, 1997.
- [34] M. A. Marsan. An introduction to generalized stochastic petri nets. In *Second International Course on Petri Nets for Latin America*, Campina Grande-PB, Novembro 1995.
- [35] T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, 1989.
- [36] S. C. Park, N. Raman, and Shaw M. J. Adaptive scheduling in dynamic flexible manufacturing systems: A dynamic rule selection approach. *IEEE Transactions on Robotics and Automation*, 13(4), August 1997.
- [37] A. Perkusich. *Análise de Sistemas Complexos Baseada na Decomposição de G-Nets*. Tese de Doutorado, Curso de Doutorado em Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, PB, August 1994.

- [38] E. M. Petriu. Automated guided vehicle with absolute encoded guide-path. *IEEE Transactions on Robotics and Automation*, pages 562–565, 1991.
- [39] J. Proth and X. Xie. *Petri Nets-A Tool for Design and Management of Manufacturing Systems*. Wiley, 1996.
- [40] Z.B. de S. Ribeiro, A. Perkusich, and T.C. Barros. Modeling and analysis of a distributed supervisor for manufacturing systems. In *Proc. of 15th International Conference on CAD/CAM Robotics & Factories of the Future, CARS & FOF'99*, pages MT1–7–MT1–12, Águas de Lindóia, SP, Brazil, August 1999.
- [41] Z.B. de S. Ribeiro, A. Perkusich, and T.C. Barros. Supervisores distribuídos tolerantes a faltas para sistemas flexíveis de manufatura. In *Anais do VII Simpósio Brasileiro de Computadores Tolerantes a Falhas*, pages 127–141, Campinas, SP, Brasil, July 1999.
- [42] W. Shen and D. H. Norrie. An agent-based approach for dynamic manufacturing scheduling. *Agent-Based Manufacturing Workshop at Autonomous Agents*, 1998.
- [43] Y. Shen and Lau. L. K. Planning of flow path to minimize expected waiting time for free-ranging automated guided vehicle systems. *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 3738–3743, 1997.
- [44] M. Silva and E. Teruel. Petri nets for the design and operation of manufacturing systems. In *First International Workshop on Manufacturing and Petri Nets*, pages 31–61, Osaka, Japao, Junho 1996.
- [45] M. Silva, E. Teruel, R. Valette, and H. Pinguad. Petri nets and production systems. In *Lectures on Petri Nets II: Applications*. Springer, 1998.
- [46] M. Silva and R. Valette. Petri nets and flexible manufacturing. In *Lecture Notes on Computer Science*. Springer-Verlag, Berlin, 1989.
- [47] T. Sun, C. Cheng, and L. Fu. A petri net based approach to modeling and scheduling for fms and a case study. *IEEE Transactions on Industrial Electronics*, pages 593–601, 1994.

- [48] M. A. J. Tamayo, D. G. Contreras, and A. R. Trevino. Petri net based control for the dynamic scheduling of a flexible manufacturing cell. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 553–557, 1998.
- [49] A. Tharumarajah, A. J. Wells, and L. Nemes. Comparison of emerging manufacturing concepts. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 325–331, 1998.
- [50] M. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers, 1993.
- [51] R. Zurawski and M. C. Zhou. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Electronics*, 41(6):567–583, 1994.