RODRIGO PIMENTA FERREIRA BRAGA

CONTRIBUIÇÕES AO CONTROLE SUPERVISÓRIO DE SISTEMAS MODELADOS POR REDES DE PETRI

FLORIANÓPOLIS
2006

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

CONTRIBUIÇÕES AO CONTROLE SUPERVISÓRIO DE SISTEMAS MODELADOS POR REDES DE PETRI

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

RODRIGO PIMENTA FERREIRA BRAGA

Florianópolis, Fevereiro de 2006.

CONTRIBUIÇÕES AO CONTROLE SUPERVISÓRIO DE SISTEMAS MODELADOS POR REDES DE PETRI

Rodrigo Pimenta Ferreira Braga

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

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

Prof. Alexandre Trofino Neto, Dr. Eng.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Nome do presidente da banca
Presidente

Nome do membro da banca

Nome do membro da banca

Nome do membro da banca

"Sobressai-se em resolver as dificuldades quem as resolve antes que apareçam."

Sun Tzu

AGRADECIMENTOS

A Huguinho, Zezinho e Luisinho, os sobrinhos do Pato Donald.

Aos bravos gauleses, Asterix, Obelix, e a seu chefe, Abracurcix, pilares da resistência contra as legiões de Júlio César.

A Stan, Kyle, Cartman e Kenny (Oh, my God! They killed Kenny! *You bastards!*).

Ao pessoal que trabalhou nesse modelo: Maziero, Antonio, Fabio, Passold, Vallim e Rafael.

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

CONTRIBUIÇÕES AO CONTROLE SUPERVISÓRIO DE SISTEMAS MODELADOS POR REDES DE PETRI

Rodrigo Pimenta Ferreira Braga

Fevereiro/2006

Orientador: Prof. José Eduardo Ribeiro Cury, Dr. d'Etat
Área de Concentração: Automação e Sistemas
Palavras-chave: Controle Supervisório, Redes de Petri, μ-calculus, Teoria das Regiões
Número de Páginas: xiii + 108

Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo.

Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo.

Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo.

Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo.

Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo. Este seria o texto de resumo.

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

CONTRIBUTIONS TO THE SUPERVISORY CONTROL OF SYSTEMS MODELED BY PETRI NETS

Rodrigo Pimenta Ferreira Braga

February/2006

Advisor: Prof. José Eduardo Ribeiro Cury, Dr. d'Etat
Area of Concentration: Automation and Systems
Key words: Supervisory Control, Petri Nets, μ-calculus, Theory of Regions
Number of Pages: xiii + 108

This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text.

This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text.

This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text. This would be an abstract text.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o passar das décadas, os sistemas de manufatura sofreram diversas alterações. A primeira mudança significativa ocorreu na passagem do estágio inicial, no qual era empregado um elevado grau de atividades manuais, para um estágio cujos equipamentos eram operados manualmente. Um segundo marco foi caracterizado pela maior dependência entre os processos e equipamentos, o que resultou no estabelecimento de seqüencias automáticas que deveriam ser respeitadas. Com a evolução tecnológica, representada tanto pelo advento dos computadores (e conseqüentemente pelo aumento do poder computacional), quanto pelo desenvolvimento de processos e equipamentos mais complexos, os sistemas de manufatura passaram a ser caracterizados por um elevado grau de automação. Um sistema de manufatura automatizado geralmente consiste de quatro componentes fundamentais: um conjunto de estações de processamento interconectadas, um sistema de transporte, um sistema de comunicação para integrar todos os aspectos da manufatura e um sistema de controle.

Um Sistema a Eventos Discretos (SED) é um sistema dinâmico cuja mudança de estado o ocorre em pontos discretos no tempo, em decorrência de eventos isolados e instantâneos por natureza. Os SEDs estão presentes em muitas aplicações do cotidiano, desde sistemas de manufatura a protocolos de comunicação, sistemas de supervisão de tráfego e sistemas computacionais. Alguns exemplos de eventos são o início (ou término) de uma tarefa, a percepção de uma mudança de estado em um sensor ou a recepção de uma mensagem proveniente de outro sub-sistema.

A natureza discreta dos SEDs inviabiliza a utilização de abordagens utilizadas para os sistemas de variáveis contínuas, modelados por equações diferenciais e tratados, dentre diversas técnicas, pela Teoria de Controle Clássica [30]. Existe uma intensa atividade de pesquisa voltada à busca de modelos matemáticos adequados à sua representação, sem que se tenha conseguido até agora encontrar uma alternativa concisa matematicamente e extremamente eficiente em termos computacionais. Dentre estes modelos, duas estruturas bastante utilizadas são as redes de Petri e os autômatos. Também é importante mencionar que a resolução de diferentes problemas relacionados aos SEDs deram origem a duas classes de técnicas com finalidades bem distintas: a síntese de controladores e a verificação formal.

Dentre as técnicas que visam a síntese automática de controladores (supervisores), a Teoria de Controle Supervisório proposta por Ramadge and Wonham [82] [1989] é uma das abordagens mais

aceitas e exploradas na literatura. A abordagem RW se baseia na modelagem da dinâmica do sistema em malha aberta (planta) e da especificação do comportamento desejado em malha fechada por autômatos que reconhecem linguagens controláveis. O supervisor possui a função de exercer uma ação de controle restritiva sobre a planta, de modo com que seu comportamento em malha fechada esteja de acordo com o especificado. Além disso, quando a especificação não pode ser cumprida, tem-se a garantia de que é possível construir um supervisor que satisfaz o comportamento desejado de forma minimamente restritiva. Sua grande vantagem é permitir a síntese automática de supervisores de forma pouco iterativa com o projetista, além de exigir menor experiência do processo a ser controlado. Como desvantagem, o processo de síntese pode se tornar inviável em sistemas complexos, devido à ocorrência de explosão de estados. Embora a abordagem RW possua complexidade de ordem polinomial em função do número de estados de um sistema, a modelagem resultante da composição de diversos subsistemas normalmente acarreta em crescimento exponencial destes estados [83]. Algumas abordagens vêm sendo desenvolvidas para amenizar este esforço computacional, tais como controle modular [95] e controle modular local [80].

Apesar de não existir um consenso sobre qual seja o melhor modelo [80] [28], outra estrutura bastante utilizada para a modelagem de SEDs consiste nas redes de Petri [76]. Uma rede de Petri consiste em uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes. Embora comprovado que a rede de Petri possui um formalismo mais poderoso e flexível do que os autômatos, esta estrutura é mais utilizada com a finalidade de análise. Isto ocorre, pois a modelagem normalmente já associa a lógica de controle ao comportamento dinâmico da planta [19]. No entanto, existem abordagens alternativas que utilizam redes de Petri como estruturas fundamentais no processo de síntese automática de supervisores [39] [48]. Em linhas gerais, estas abordagens possuem duas grandes vertentes: uma que utiliza as propriedades estruturais da rede de Petri que modela o comportamento da planta e outra baseada em linguagens controladas. Em outras palavras, a primeira vertente faz uso de propriedades que naturalmente são utilizadas na análise de redes de Petri, enquanto a segunda vertente faz uso de alguma técnica semelhante à abordagem RW.

As técnicas de verificação formal, por sua vez, permitem aos projetistas verificar se uma dada especificação é válida para um controlador. De forma a realizar esta análise, o projetista deve construir um modelo para o sistema controlado e elaborar uma expressão lógica para cada propriedade a ser verificada. A lógica temporal CTL* [32] e μ-calculus [14] são duas possibilidades de expressar estas expressões lógicas. Os algoritmos de *model checking* [55] [18], por sua vez, possuem a finalidade de confirmar se cada propriedade é satisfeita. Caso contrário, é fornecido um contra-exemplo e o projetista deve alterar o modelo do sistema controlado e repetir todo o procedimento até uma configuração satisfatória ser obtida. Desta forma, a utilização de verificação formal se distingue da maioria das técnicas de síntese de supervisores, pois é necessário a atuação iterativa do projetista para corrigir eventuais falhas na estrutura do sistema controlado. Embora também sofram do problema de explosão de estados, as técnicas de verificação formal são mais utilizadas em sistemas complexos. Isto se justifica pelo fato destas técnicas serem utilizadas sobre um sistema controlado final, e não durante o processo de síntese, onde a explosão de estados é mais evidenciada. Conseqüentemente, até o presente momento, estas técnicas possuem maior aceitação em grandes projetos [24] [49], comparadas

às técnicas de síntese de supervisores.

Apesar de inicialmente serem destinadas a classes distintas de problemas relacionados aos SEDs, o presente trabalho possui o objetivo de utilizar alguns princípios de verificação formal em conjunto com a síntese de supervisores, de forma a explorar as características peculiares de cada técnica. Dentre os objetivos secundários, deseja-se que esta metodologia seja computacionalmente eficiente, genérica e flexível. Com relação à eficiência computacional, seria aceitável que esta contribuição possuísse, no pior caso, uma ordem de complexidade semelhante àquela obtida pela abordagem RW. Por generalidade e flexibilidade, entende-se que esta metodologia seja aplicável a uma ampla classe de problemas e que possa ser estendida a outras abordagens existentes na literatura, respectivamente. Com o intuito de contribuir ainda mais para as técnicas de síntese de supervisores, utilizou-se sistemas modelados por redes de Petri, por serem estruturas com menor objeto de estudo nesta área do conhecimento. Dentro deste contexto, dois trabalhos recentes formam os pilares da metodologia proposta neste documento:

- Ghaffari et al. [35] utilizam como base sistemas modelados por redes de Petri e fazem uso da abordagem RW e da Teoria das Regiões [6], uma teoria inicialmente desenvolvida para a síntese de redes de Petri a partir de estruturas em grafo.
- Ziller and Schneider [100] também se baseiam na abordagem RW, mas resolvem a síntese de supervisores utilizando-se sistemas modelados por autômatos e expressões lógicas expressas em μ-calculus.

Ao conciliar diversas características dos trabalhos supracitados com outros aspectos de modelagem de SEDs por redes de Petri, buscou-se atender os objetivos já mencionados. Inicialmente, foram contemplados SEDs sem considerar explicitamente aspectos temporais. A principal contribuição foi a proposta de uma nova metodologia e conseqüente resolução de alguns problemas encontrados na literatura. Como principais ferramentas de apoio, foram utilizados o *software* Tina (modelagem e análise de redes de Petri) [12], o *model checker* MEC [4] e alguns programas auxiliares desenvolvidos em linguagem C++. Como parte da aplicação da Teoria das Regiões, os modelos de programação linear inteira foram todos gerados em linguagem AMPL, podendo ser resolvidos por qualquer ferramenta de otimização que reconheça tal linguagem. Por último, foi apresentado uma das extensões temporais das redes de Petri e foram discutidos aspectos relativos a utilização de uma metodologia mais semelhante possível à proposta para o caso não temporal.

Esta dissertação é estruturada da seguinte forma: o Capítulo 2 introduz todos os formalismos necessários ao entendimento do restante deste documento. Inicialmente são apresentadas algumas características das redes de Petri, realçando definições básicas e descrição de algumas de suas propriedades. Em seguida, é feito um breve resumo da Teoria de Linguagens [50], assunto introdutório para a descrição da Teoria de Controle Supervisório, apresentada logo em seguida. Foram sumarizados os principais conceitos relativos ao controle monolítico, ou seja, aquele caracterizado pela existência de um único supervisor. Por último, são explicitados alguns aspectos referentes à verificação de sistemas, incluindo as propriedades normalmente verificadas, lógica temporal CTL* e μ-calculus.

O Capítulo 3 discute diversas abordagens alternativas ao controle supervisório de SEDs que utilizam estruturas em redes de Petri e algumas de suas extensões como modelos. Após uma caracterização em ordem cronológica de diversos trabalhos, são apresentados as metodologias propostas por Ghaffari et al. [35] e Ziller and Schneider [100]. Estes trabalhos foram apresentados de forma minuciosa, pois formam os pilares para as contribuições apresentadas neste documento.

O Capítulo 4, por sua vez, propõe uma metodologia para a síntese automática de supervisores em SEDs modelados por redes de Petri. Após uma apresentação dos princípios gerais que regem esta metodologia, é feita uma breve descrição das duas principais ferramentas computacionais utilizadas, Tina e MEC. Em seguida, é apresentado o algoritmo proposto, com uma descrição detalhada de cada um de seus passos, inclusive dos programas auxiliares desenvolvidos em linguagem C++.

Baseado diretamente no algoritmo proposto no Capítulo 4, o Capítulo 5 consiste na resolução de alguns problemas constantemente citados na literatura, bem como do problema da célula de fabricação apresentada originalmente por Ghaffari et al. [35]. Nos primeiros exemplos, verificou-se que a rede de Petri supervisionada resultante possui um comportamento isomórfico ao autômato do supervisor obtido pela abordagem RW. No último caso, buscou-se ressaltar as principais diferenças entre a abordagem original e a presente neste documento. Este capítulo se encerra com uma análise da aplicabilidade da metodologia proposta.

O Capítulo 6 visa considerar explicitamente aspectos temporais, de forma a enriquecer o processo de síntese com tais informações. Dentre as duas extensões mais comuns na literatura, optou-se pela rede de Petri temporal [65], devido ao seu maior poder de modelagem. Após uma breve apresentação dos aspectos mais importantes da rede de Petri temporal e seu comportamento dinâmico associado, o restante deste capítulo discorre sobre perspectivas de utilização deste modelo. Embora não chegue a um resultado tão expressivo quanto ao apresentado no Capítulo 4, são trilhados caminhos por onde se deve e não se deve percorrer. O desfecho se faz por um possível algoritmo.

Finalmente, o Capítulo 7 apresenta as conclusões desta pesquisa, que consistem nas contribuições ao controle supervisório, bem como em sugestões para trabalhos futuros.

Capítulo 2

Formalismos de base

Este capítulo apresenta os principais conceitos sobre os quais este documento se baseia. Ao invés de fazer uma apresentação exaustiva de cada tema, buscou-se resumir os formalismos necessários e suficientes para o entendimento dos demais capítulos.

É importante ressaltar que os SEDs considerados são deterministas, ou seja, são sistemas que, estando em um determinado estado, sempre reagem da mesma forma à ocorrência de um mesmo evento. Ademais, as representações matemáticas apresentadas não admitem a ocorrência simultânea de dois ou mais eventos distintos, o que faz com que os sistemas modelados apresentam comportamento dinâmico seqüencial. Neste momento, a modelagem também não considera os instantes de tempo em que os eventos ocorrem, mas apenas a ordem em que acontecem.

A primeira seção apresenta a rede de Petri [19], uma ferramenta flexível e poderosa para modelagem de SEDs. Em seguida, é introduzida a Teoria de Linguagens, base para a Teoria de Controle Supervisório proposta por Ramadge e Wonham. A terceira seção apresenta os principais conceitos da estrutura do controle monolítico, ou seja, aquele caracterizado pela existência de um único supervisor. Detalhes aprofundados sobre esta abordagem podem ser encontrados em [20], [28] e [95]. Por último, é feita uma pequena discussão sobre as técnicas de verificação, dando ênfase para as propriedades normalmente verificadas, lógica temporal CTL* e μ-calculus.

2.1 Rede de Petri

Com origem na tese de Petri [76], rede de Petri consiste em uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes. Embora inicialmente utilizada para descrever as relações casuais entre condições e eventos em sistemas computacionais, rede de Petri representa um formalismo poderoso e flexível para a modelagem de SEDs em geral. Além de informações estruturais, uma rede de Petri fornece o comportamento dinâmico do sistema modelado. Dentre outras coisas, é possível verificar se as transições são paralelas ou conflitantes, se uma transição está ou não sensibilizada, bem como disparar uma transição e fazer evoluir a rede.

2.1.1 Definições

Uma rede de Petri é uma quádrupla

$$R =< P, T, Pre, Post > \tag{2.1}$$

onde P é um conjunto finito de lugares, T é um conjunto finito de transições, $Pre : (P \times T) \rightarrow \mathbb{N}$ é uma função[1] que define arcos dos lugares às transições e $Post : (P \times T) \rightarrow \mathbb{N}$ é uma função[2] que define arcos das transições aos lugares. Se $Pre(p,t) = k$ (ou $Post(p,t) = k$), então existem k arcos direcionados conectando o lugar p à transição t (ou transição t ao lugar p).

Graficamente, uma rede de Petri é constituída por um grafo com três tipos de objetos: lugares, transições e arcos direcionados conectando lugares a transições e transições a lugares. Normalmente, os lugares são representados por círculos, enquanto as transições são representadas por barras estreitas. Visando simplificar a representação gráfica, K arcos conectando um lugar a uma mesma transição (ou uma transição a um mesmo lugar) são representados por um arco simples com peso[3] k.

Uma rede é dita pura se e somente se (sse) $\forall p \in P$ e $\forall t \in T$, $Pre(p,t) \times Post(p,t) = 0$, ou seja, se nenhum lugar é entrada e saída de uma mesma transição.

Uma marcação é um mapeamento $M : P \rightarrow \mathbb{N}$ que associa um número inteiro de fichas[4] a cada lugar. Conseqüentemente, $M(p)$ indica a quantidade de fichas presente no lugar p. Uma rede marcada é uma dupla

$$N =< R, M_0 > \tag{2.2}$$

onde $R =< P, T, Pre, Post >$ e M_0 é a marcação inicial. A partir de agora, rede de Petri marcada será denotada por RdP.

O fluxo de fichas é dado pela habilitação e disparo de transições. Uma transição t está habilitada em uma marcação M sse $M \geq Pre(.,t)$, ou seja, quando cada um de seus lugares de entrada p contém pelo menos um número de fichas igual ao peso do arco direcionado conectando p a t. Esta habilitação será denotada por $M[t \rightarrow$.

O disparo de uma transição t remove de cada lugar p o número de fichas igual ao peso do arco conectando p a t. Além disso, este disparo deposita em cada lugar de saída o número de fichas igual ao peso do arco conectando t a p. Matematicamente, o disparo da transição t em uma marcação M leva a uma nova marcação $M' = M - Pre(.,t) + Post(.,t)$.

[1] Alguns nomes encontrados na literatura são: função de incidência anterior, função de lugares precedentes ou função de aplicação de entrada

[2] Alguns nomes encontrados na literatura são: função de incidência posterior, função de lugares seguintes ou função de aplicação de saída

[3] O peso de um arco também é chamado de multiplicidade

[4] As fichas também são chamadas de *tokens* ou marcas

Qualquer marcação M alcançável a partir da marcação inicial M_0 pelo disparo de uma seqüencia $\sigma = t_1 t_2 ... t_n$ satisfaz a equação fundamental da RdP

$$M = M_0 + C\vec{\sigma} \tag{2.3}$$

onde $\vec{\sigma} : T \rightarrow \mathbb{N}$ é chamado vetor de contagem e $C(p,t) = Post(p,t) - Pre(p,t)$ é denominada matriz de incidência. Como pode-se observar, a matriz de incidência fornece o balanço das fichas quando ocorre o disparo de transições.

O conjunto de todas marcações alcançáveis a partir de uma marcação M em N é denominado $A(N,M)$. O comportamento de uma RdP pode ser descrito por um grafo de alcançabilidade[5] $G_A(N,M_0)$, cujos nodos correspondem às marcações alcançáveis[6]. Neste grafo, existe um arco com etiqueta t do nodo M ao nodo M' sse $M[t \rightarrow M'$.

Duas categorias de RdP com características peculiares são o grafo de eventos [7] e o grafo de estados. Um grafo de eventos é uma RdP em que cada lugar possui exatamente um arco de entrada e um arco de saída, enquanto um grafo de estados é uma RdP em que cada transição possui unicamente um arco de entrada e um arco de saída. Estruturas em grafos marcados podem modelar a sincronização de processos concorrentes, pois as fichas em lugares que compartilham uma transição de saída devem progredir de forma síncrona. Entretanto, estas estruturas não podem representar a possibilidade de escolha no modelo da planta, pois existe somente uma alternativa de remoção de uma ficha de um dado lugar. Conseqüentemente, todas as transições sensibilizadas são persistentes, o que significa que, uma vez sensibilizadas, permanecem neste estado até serem disparadas. Estruturas em grafos marcados, por sua vez, apresentam características opostas em relação às estruturas em grafos marcados. Uma estrutura em grafo de estados com uma única ficha é análoga a um autômato de estados finitos: cada lugar corresponde a um estado do autômato e a localização da ficha indica seu estado atual. Grafos de estados com múltiplas fichas podem representar uma categoria restrita de concorrência, pois mais de uma transição pode estar habilitada em uma dada marcação. Estas estruturas são ilustradas na figura 2.1, juntamente com uma RdP genérica.

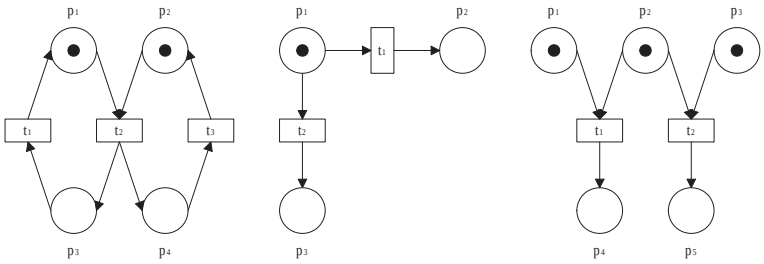

Figura 2.1: Exemplo de grafo de eventos, grafo de estados e RdP genérica

Como ferramenta matemática, uma rede de Petri permite que o projetista identifique a presença

[5] O grafo de alcançabilidade também é chamado de grafo de marcações alcançáveis, grafo de acessibilidade ou grafo de marcações acessíveis

[6] Marcações alcançáveis também são chamadas de marcações acessíveis

[7] Um grafo de eventos é também chamado de grafo marcado

ou ausência de propriedades funcionais específicas do sistema modelado. Estas propriedades são divididas em dois grupos: propriedades comportamentais e estruturais. As definições das propriedades comportamentais implicam considerações sobre o conjunto de marcações acessíveis a partir da marcação inicial, enquanto as propriedades estruturais dependem apenas da estrutura da rede [19].

2.1.2 Propriedades comportamentais

As definições das propriedades comportamentais implicam considerações sobre o conjunto de marcações acessíveis a partir da marcação inicial. As propriedades limitabilidade, reiniciabilidade e vivacidade são normalmente reagrupadas sob o nome genérico de boas propriedades.

Limitabilidade

Um lugar p de uma RdP N é k-limitado sse $\forall M' \in A(R,M_0)$, $M'(p) \leq k$. Se $k = 1$, o lugar é denominado binário[8]. Uma RdP N é k-limitada (binária) sse todos seus lugares são k-limitados (binários).

Uma RdP limitada corresponde ao fato de que um sistema físico é sempre limitado. Entretanto, uma RdP não limitada pode ser utilizada quando se quer avaliar o desempenho de um sistema independentemente dos limites de seus elementos de armazenamento intermediários.

Reiniciabilidade

Uma RdP N é reiniciável[9] sse $\forall M' \in A(R,M_0)$, $\exists \sigma |\ M'[\sigma \rightarrow M_0$. Desta forma, é possível, a partir de qualquer marcação acessível M' de $G_A(R,M_0)$, encontrar uma seqüencia de disparo σ que leve a rede de volta à marcação inicial M_0. A reiniciabilidade é característica de sistemas físicos que possuem comportamentos repetitivos.

Vivacidade

Uma transição t de uma RdP N é viva sse $\forall M' \in A(R,M_0)$, $\exists \sigma |\ M'[\sigma t \rightarrow$, ou seja, se uma transição é viva, ela pode ser sensibilizada a partir de qualquer marcação M' do grafo de alcançabilidade, através de uma seqüencia de disparos σ.

Existe uma definição menos rígida denominada quase-vivacidade. Uma transição t de uma RdP N é quase viva sse $\exists \sigma |\ M_0[\sigma \rightarrow M'\ e\ M'[t \rightarrow$, ou seja, uma transição t é quase viva sse é possível sensibilizá-la por uma marcação do grafo de alcançabilidade, obtida através de uma seqüencia de disparos σ a partir da marcação inicial M_0.

[8] Um lugar binário também é chamado de lugar seguro
[9] Uma RdP reiniciável também é chamada de RdP própria

Uma RdP N é viva (semi-viva) sse todas suas transições são vivas (semi-vivas). A vivacidade garante que nenhum bloqueio pode ser provocado pela estrutura da rede.

2.1.3 Propriedades estruturais

As propriedades estruturais são propriedades derivadas diretamente da estrutura de rede de Petri e que não dependem da sua marcação inicial. Estas propriedades constituem os componentes conservativos de lugar e os componentes repetitivos estacionários e podem ser facilmente deduzidas a partir da equação fundamental da RdP [19]. Em conjunto com informações de marcação, são definidos os invariantes de lugar e de transição, que fornecem informações adicionais sobre o comportamento dinâmico da rede.

Componentes conservativos e invariantes de lugar

Um invariante linear de lugar é uma função linear da marcação dos lugares cujo valor é uma constante que depende apenas da marcação inicial da RdP. Ele corresponde a uma restrição sobre os estados e as atividades do sistema que será sempre verificada, quaisquer que sejam suas evoluções.

Um componente conservativo de uma rede de Petri é o conjunto de lugares $p \in P$ correspondentes aos elementos não nulos do vetor coluna $\vec{f}$, solução da equação $\vec{f}^T C = 0 \;\; com \;\; \vec{f} > 0$. Uma rede de Petri é conservativa se todos seus lugares pertencem a um componente conservativo.

Se $\vec{f}$ é solução da equação anterior, então a função linear $\vec{f}^T M = \vec{f}^T M_0, \;\; \forall M \in A(R, M_0)$ é o invariante linear de lugar correspondente. Observa-se que o invariante de lugar depende da marcação inicial, enquanto que o componente conservativo é completamente independente.

Componentes repetitivos e invariantes de transição

Um invariante de transição corresponde a uma seqüencia cíclica de eventos que pode ser repetitiva indefinidamente sem modificar a marcação da rede.

Um componente repetitivo estacionário de uma rede de Petri, dado pelo vetor coluna $\vec{\sigma}$, corresponde a uma solução da equação $C\vec{\sigma} = 0$. Uma rede de Petri é repetitiva se todas suas transições pertencem a um componente repetitivo estacionário.

A seqüencia σ associada ao vetor $\vec{\sigma}$ é dita invariante de transição. Como é observado, o componente repetitivo estacionário não depende da marcação inicial, ao contrário do invariante de transição correspondente de quem sua existência depende.

2.1.4 Exemplo

Considera-se a linha de transferência com retrabalho de peças rejeitadas ilustrada pela figura 2.2. Este sistema é composto por uma máquina M_1, um *buffer* unitário B_1, uma máquina M_2, um *buffer*

unitário B_2 e uma máquina de medição M_3 que entrega as peças boas na saída da linha e retorna as peças rejeitadas no buffer B_1 para serem retrabalhadas.

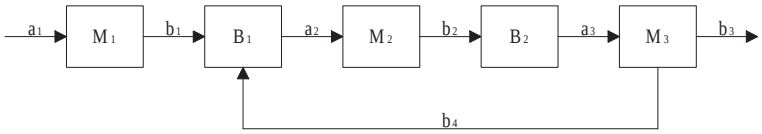

Figura 2.2: Linha de transferência com retrabalho de peças

Para $i = 1,2,3$, consideram-se α_i como o início de operação da máquina M_i com a retirada de uma peça de B_{i-1} (quando existir), e β_i como o fim de operação da máquina M_i com o depósito de uma peça em B_i (quando existir). Já o evento β_4 representa o retorno de uma peça para ser retrabalhada.

Considerando-se que os *buffers* possuem capacidade unitária, este sistema pode ser modelado pela RdP representada na figura 2.3. Ao analisar o grafo de alcançabilidade correspondente, ilustrado na figura 2.4, observa-se facilmente que esta rede é k-limitada, viva e reiniciável. A marcação inicial M_0 corresponde ao estado da RdP com fichas nos lugares p_1, p_7 e p_9. As demais marcações marcações podem ser obtidas pela equação fundamental da RdP.

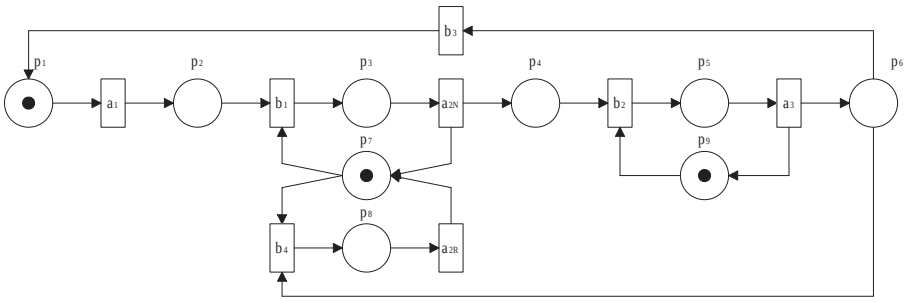

Figura 2.3: RdP da linha de transferência com retrabalho de peças

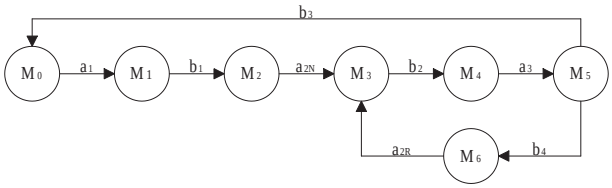

Figura 2.4: Grafo de alcançabilidade da RdP da linha de transferência com retrabalho de peças

Os invariantes de lugar e de transição são enunciados na tabela 2.1.

2.2 Teoria de Linguagens

Na Teoria de Linguagens, o comportamento de um SED pode ser modelado por um conjunto de cadeias de símbolos, que representam todas as possíveis seqüencias de eventos admitidas pelo sistema. Se o comportamento lógico de um SED pode ser descrito por uma expressão regular, então existe um autômato capaz de gerar esta mesma linguagem, de acordo com regras bem definidas [50].

Invariantes de lugar	**Interpretação**
$M(p_3)+M(p_7)+M(p_8)=1$	O *buffer B*1 possui dimensão unitária, nunca ocorrendo *underflow*, nem *overflow*
$M(p_5)+M(p_9)=1$	O *buffer B*2 possui dimensão unitária, nunca ocorrendo *underflow*, nem *overflow*
$M(p_1)+M(p_2)+M(p_3)+M(p_4)+M(p_5)+$ $M(p_6)+M(p_8)=1$	Há sempre uma, e somente uma, peça sendo produzida ou retrabalhada
Invariantes de transição	**Interpretação**
$\beta_2\ \alpha_3\ \beta_4\ \alpha_{2R}$	Uma peça poderá ser retrabalhada diversas vezes
$\alpha_1\ \beta_1\ \alpha_{2N}\ \beta_2\ \alpha_3\ \beta_3$	Após uma peça ser fabricada com sucesso, o ciclo de produção de outra peça é iniciado

Tabela 2.1: Invariantes de lugar e de transição da RdP da linha de transferência com retrabalho de peças

2.2.1 Linguagens Formais

Seja Σ o conjunto finito de todos os eventos que ocorrem em um SED. Seja Σ^* o conjunto de todas as cadeias finitas constituídas de elementos presentes em Σ, incluindo a cadeia vazia ε. Uma linguagem L definida sobre o alfabeto Σ é um subconjunto de Σ^*.

O comportamento lógico de um SED de alfabeto Σ pode ser modelado por um par de linguagens $L \subseteq \Sigma^*$ e $L_m \subseteq L$. L representa todas as cadeias finitas de eventos que o SED pode gerar, ou seja, todas cadeias que são fisicamente realizáveis. Por sua vez, L_m descreve todas as tarefas completas possíveis de serem realizadas. Devido às suas características, L e L_m são denominadas linguagem gerada e linguagem marcada.

Uma cadeia $u \in \Sigma^*$ é um prefixo de $v \in \Sigma^*$ se, para alguma cadeia $s \in \Sigma^*$, $v = us$. O prefixo fechamento, ou simplesmente fechamento, de uma linguagem L é dado por $\bar{L} = \{u \in \Sigma^* | \exists v \in \Sigma^* \wedge uv \in L\}$. Uma linguagem é prefixo-fechada sempre que $L = \bar{L}$.

A descrição de uma linguagem nem sempre é realizada por meio das cadeias que a definem. Existem estruturas compactas, concisas e sem ambigüidades que são utilizadas para tal finalidade. Duas destas estruturas são as expressões regulares e os autômatos.

2.2.2 Expressões Regulares

Expressões regulares fornecem um meio de descrição de linguagens. Dado um alfabeto Σ, uma expressão regular é definida recursivamente da seguinte forma:

1.
   - $\emptyset$ é uma expressão regular que representa a linguagem vazia,
   - ε é uma expressão regular denotando a linguagem $\{\varepsilon\}$
   - σ é uma expressão regular denotando $\{\sigma\}\ \forall e \in \Sigma$
2. Se r e s são expressões regulares, então rs, r^*, s^*, $(r+s)$ são expressões regulares;
3. Toda expressão regular é obtida pela aplicação das regras 1 e 2 um número finito de vezes

O operador * presente na regra 2 é denominado fechamento Kleene. O fechamento Kleene de uma linguagem $L \subseteq \Sigma$ é definido por $L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup ....$ Sendo assim, uma cadeia de L^* é formada pela concatenação de um número finito de cadeias de L, incluindo a cadeia vazia ε.

2.2.3 Autômatos e Geradores

Um autômato de estados finitos é uma quíntupla

$$A =< \Sigma, X, \delta, x_0, X_M > \tag{2.4}$$

onde Σ é um alfabeto de eventos, X é um conjunto de estados, $x_0 \in X$ é o estado inicial, $X_M \in X$ é o conjunto de estados marcados [10] e $\delta : (\Sigma \times X) \rightarrow X$ é uma função de transição de estados.

Gerador é um autômato $G =< \Sigma, Q, \delta, q_0, Q_M >$ em que $\delta : (\Sigma \times X) \rightarrow X$ é uma função parcial, ou seja, δ está definida em cada estado $q \in Q$ somente para um subconjunto de elementos $\sigma \in \Sigma$. A notação $\delta(\sigma, q)!$ representa que $\delta(\sigma, q)$ está definida.

A função de transição δ pode ser naturalmente estendida para cadeias de eventos. Assim, defini-se a função de transição estendida como a função $\hat{\delta} : (\Sigma^* \times Q) \rightarrow Q$ tal que, para $q \in Q$, $s \in \Sigma^*$ e $\sigma \in \Sigma$, $\hat{\delta}(\varepsilon, q) = q$ e $\hat{\delta}(s\sigma, q) = \delta(\sigma, \hat{\delta}(s, q))$, sempre que $q' = \hat{\delta}(s, q)!$ e $\delta(\sigma, q')!$.

Um estado $q \in Q$ é acessível se $\exists s \in \Sigma^* \mid q = \delta(s, q_0)$. Um estado é $q \in Q$ é co-acessível se $\exists s \in \Sigma^* \mid \delta(s, q) \in Q_M$. Um gerador G é acessível (co-acessível) se todos seus estados forem acessíveis (co-acessíveis) [11]. G é trim caso seja acessível e co-acessível simultaneamente.

Os autômatos normalmente são ilustrados por diagramas de transição de estados, que são grafos direcionados onde os nós representam os estados e os ramos representam os eventos. Neste diagramas, o estado inicial é identificado por uma seta e os estados marcados são caracterizados por nós desenhados com linhas duplas.

2.3 Teoria de Controle Supervisório

Desenvolvida por Ramadge and Wonham [82] [1989], a Teoria de Controle Supervisório (TCS) de SEDs possui algumas terminologias semelhantes àquelas utilizadas na Teoria de Controle clássica de sistemas contínuos. Esta semelhança explica o uso dos termos planta, supervisor [12] e sistema em malha fechada. A planta possui todos os comportamentos possíveis do sistema a ser controlado, incluindo algumas situações indesejáveis, tais como bloqueios na linha de produção ou colisão entre robôs autônomos. Além disso, a planta é vista como um sistema que gera eventos e que possui entradas de controle, através das quais alguns eventos podem ser desabilitados em determinados estados.

[10] Os estados marcados também são chamados de estados finais

[11] As propriedades acessível e co-acessível também são chamadas de alcançável e co-alcançável, respectivamente.

[12] Na TCS, supervisor é o nome mais usual para controlador

O supervisor é um agente externo que possui habilidade de observar os eventos gerados pela planta e influenciar seu comportamento através da entrada de controle. Em malha fechada, a ação de controle do supervisor garante que o funcionamento da planta esteja de acordo com uma dada especificação, funcionamento este que não poderia ser obtido em malha aberta.

2.3.1 Controle Monolítico

No controle monolítico de SEDs, assume-se a existência de um único supervisor para restringir o comportamento global de uma planta. A figura 2.5 ilustra a malha fechada para este caso.

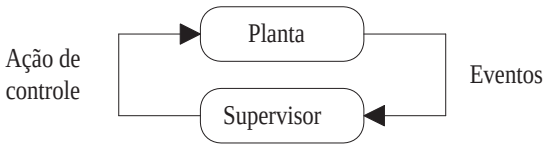

Figura 2.5: Modelo Ramadge-Wonham

Os problemas de controle são formulados utilizando Teoria de Linguagens e geradores. Desta forma, a planta a ser controlada é modelada por um autônomo $G =< \Sigma, Q, \delta, q_0, Q_M >$, cujo comportamento é caracterizado por dois subconjuntos de Σ^*: o comportamento gerado, denotado por $L(G)$ e o comportamento marcado, denotado por $L_m(G)$. Formalmente, estas linguagens são definidas como $L(G) = \{s \in \Sigma^* \mid \hat{\delta}(s, q_0)!\}$ e $L_m(G) = \{s \in \Sigma^* \mid \hat{\delta}(s, q_0) \in Q_M\}$.

Para as próximas notações, é conveniente definir o conjunto de eventos ativos $act_A(q)$ como o subconjunto de eventos possíveis de ocorrerem no estado q. Formalmente, dados um autônomo $G =< \Sigma, Q, \delta, q_0, Q_M >$ e um estado $q \in Q$, o conjunto de eventos ativos é definido como $act_A = \{\sigma \in \Sigma \mid \delta(s, q)!\}$.

Quando uma planta e seu supervisor são representados por geradores G e S, o comportamento do sistema em malha fechada é representado por um produto síncrono. O produto síncrono dos geradores [13] $G =< \Sigma, Q_G, \delta_G, q_G^0, Q_G^M >$ e $S =< \Sigma, Q_S, \delta_S, q_S^0, Q_S^M >$ é o autônomo $G \times S =< \Sigma, Q_G \times Q_S, \delta_{G\times S}, (q_G^0, q_S^0), Q_G^M \times Q_S^M >$, onde $\delta_{G\times S}((p,q),\sigma)! \Leftrightarrow \delta_G(p,\sigma)! \wedge \delta_S(q,\sigma)!$. Com o produto síncrono, se uma transição está presente em somente um dos estados p ou q, ela não estará presente no estado (p,q), ou seja, $act_{G\times S}((p,q)) = act_G(p) \cap act_S(q)$. A ação de controle do supervisor habilita somente os eventos em $act_{G\times S}$. Sendo assim, para desabilitar a ocorrência de um evento σ quando $G \times S$ está no estado (p,q), é suficiente omitir σ no estado q de S.

Normalmente, os SEDs contém alguns eventos que naturalmente não podem ser desabilitados, tais como a quebra de um equipamento ou a sinalização de um sensor. Desta forma, a abordagem Ramadge-Wonham particiona o conjunto Σ em eventos controláveis Σ_c e eventos não controláveis $\Sigma_u = \Sigma - \Sigma_c$. Os eventos controláveis correspondem àqueles que o supervisor pode desabilitar, enquanto os eventos não controláveis são aqueles cuja ocorrência não pode ser evitada. Esta característica corresponde a uma restrição na existência de supervisores, pois nem sempre é possível construir

[13] O produto síncrono pode ser estendido para mais de dois geradores, sendo denotado por $\|$

um supervisor que restrinja o comportamento da planta a uma linguagem de especificação de forma exata.

Uma linguagem $K \subseteq \Sigma^*$ é uma linguagem controlável de $L(G)$ se $\overline{K}\Sigma_u \cap L(G) = \overline{K}$, ou seja, se $\forall s \in \overline{K}, s\Sigma_u \subset L(G) \Rightarrow s\Sigma_u \subset \overline{K}$. Desta forma, uma especificação dada por um autônomo E pode ser implementada por um supervisor sse $\forall (p,q) \in G \times E$, $act_E(p) - act_{G\times E}((p,q)) \subseteq \Sigma_c$. Especificações que não atendem este requerimento são chamadas de não controláveis, pois permitem que a planta alcance um estado cujos eventos não controláveis podem ocorrer e, ao mesmo tempo, tentam proibir a ocorrência de um ou mais destes eventos neste estado. Formalmente, isto significa que o produto $G \times E$ possui um ou mais maus estados, que são estados (p,q) que falham ao satisfazer a seguinte condição:

$$act_{G\times E}((p,q)) \supseteq act_G(p) \cap \Sigma_u. \tag{2.5}$$

Dado um gerador especificação E, a linguagem $K = L_m(E)$ é controlável se e somente se E não possui maus estados.

A linguagem marcada de uma planta G sob controle de um supervisor S é denotada por $L_m(S/G) = L(S/G) \cap L_m(G)$. Ramadge and Wonham [82] [1989] mostraram que, para qualquer planta G e qualquer linguagem de especificação $K \subseteq L_m(G)$, existe uma máxima linguagem controlável de K, denotada por $supC(K)$. Desta forma, dada uma especificação K não-controlável, é possível calcular $supC(K)$ e construir um supervisor S tal que $L_m(S/G) = supC(K)$. Esta linguagem pode substituir a especificação original, pois atende às especificações de forma minimamente restritiva.

Outra restrição imposta ao supervisor é que sua ação de controle sempre permita que o sistema complete uma tarefa. Uma tarefa completa é representada nos geradores por meio de estados marcados. Existem duas situações que caracterizam um bloqueio:

- *Deadlock*: o SED alcança um estado cuja nenhuma tarefa é terminada e nenhum outro evento pode ocorrer.
- *Livelock*: o comportamento do SED permanece infinitamente dentro de um subconjunto de estados, nenhum dos quais correspondendo a uma tarefa terminada

A figura 2.6 ilustra estes conceitos. Um supervisor que evita estas situações é chamado de não-bloqueante, o que é indicado formalmente por $\overline{L_m(S/G)} = L(S/G)$. Graficamente, esta condição é representada por um gerador trim.

Como se pode ver controlabilidade e ausência de bloqueio estão fortemente relacionadas no Problema de Controle Supervisório (PCS). Dada uma planta G, uma linguagem de especificação $K \subseteq L_m(G)$ representando o comportamento desejado de G sob supervisão e um comportamento minimamente aceitável $A_{min} \subseteq K$, o PCS consiste em encontrar um supervisor não-bloqueante S tal que $A_{min} \subseteq L_m(S/G) \subseteq K$.

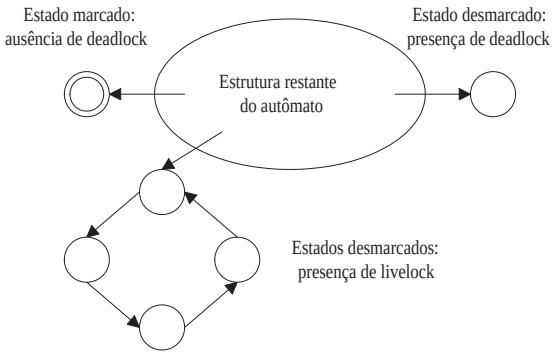

Figura 2.6: Exemplo de *deadlock* e *livelock*

PCS é calculável sse $supC(K) \supseteq A_{min}$, sendo $supC(K)$ sua solução menos restritiva possível [82] [83]. Um gerador coacessível S, cuja linguagem marcada é igual a $supC(K)$ pode ser calculado a partir dos geradores G e E, com E construído de tal forma que $L_m(E) = K$. Devido ao fato do gerador final ser um supervisor, este cálculo é normalmente chamado de síntese do supervisor.

A complexidade do cálculo da máxima linguagem controlável é dada por $O(|Q|^2 \times |\Sigma|)$ [83]. Na prática, é comum modelar SEDs complexos através do produto síncrono de componentes mais simples. Na pior das hipóteses, cada vez que um componente de N estados é adicionado, o número de estados do gerador resultante é multiplicado por N. Desta forma, o tamanho do modelo aumenta exponencialmente com o tamanho dos componentes individuais, o que limita a resolução do controle monolítico para SEDs de grande porte. Algumas abordagens vêm sendo desenvolvidas para amenizar este esforço computacional, tais como controle modular [95] e controle modular local [80].

2.3.2 Exemplo

Considera-se a linha de transferência ilustrada na figura 2.7. Este sistema é composto por três máquinas (M_1, M_2 e M_3) e dois *buffers* intermediários de capacidade unitária (B_1 e B_2). Para $i = 1,2,3$, consideram-se α_i como o início de operação da máquina M_i com a retirada de uma peça de B_{i-1} (quando existir), e β_i como o fim de operação da máquina M_i com o depósito de uma peça em B_i (quando existir). Sabe-se também que $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3\}$ e $\Sigma_u = \{\beta_1, \beta_2, \beta_3\}$.

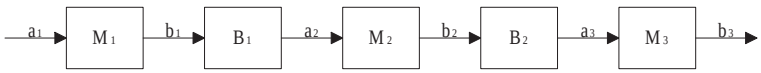

Figura 2.7: Linha de transferência com três máquinas

O comportamento de cada máquina M_i pode ser modelado individualmente pelo gerador G_i apresentado na figura 2.8. Este gerador possui o estado inicial de repouso e um estado que representa o processamento de uma peça. O comportamento em malha aberta da planta é obtido pelo produto síncrono dos geradores das três máquinas, ou seja, $G = \mathop{\|}_{i=1}^{3} G_i$. O gerador G é apresentado na figura 2.9.

Como restrições de controle, deseja-se que não ocorra nem *underflow*, nem *overflow* em nenhum

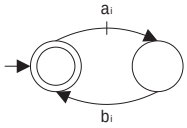

Figura 2.8: Gerador G_i para a máquina i

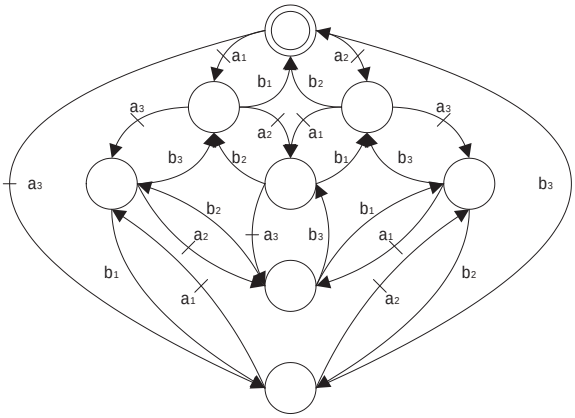

Figura 2.9: Gerador G para o sistema em malha aberta

dos *buffers* intermediários. Para $i = 1,2$, a restrição R_i é satisfeita alternando-se os eventos β_i e α_{i+1}. Os geradores que expressam R_i são ilustrados na figura 2.10.

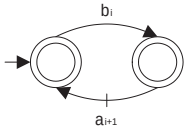

Figura 2.10: Gerador para a restrição de controle R_i

A especificação E é obtida pelo produto síncrono das restrições de controle R_1 e R_2 com a planta G e o gerador correspondente possui 32 estados. Como a linguagem da especificação é não controlável em relação à G, é preciso calcular a máxima linguagem controlável $SupC(E,G)$. O supervisor que implementa a especificação de forma minimamente restritiva é ilustrado na figura 2.11.

2.4 Verificação de Sistemas

Devido à crescente magnitude e complexidade dos SEDs (tais como protocolos de comunicação e sistemas de manufatura), cada vez é mais exigido a utilização de técnicas e ferramentas que facilitem a análise automática de determinadas propriedades. Desta forma, a verificação de sistemas, ou seja, o processo de determinação da conformidade de especificações e produtos, está se tornando uma atividade cada vez mais necessária.

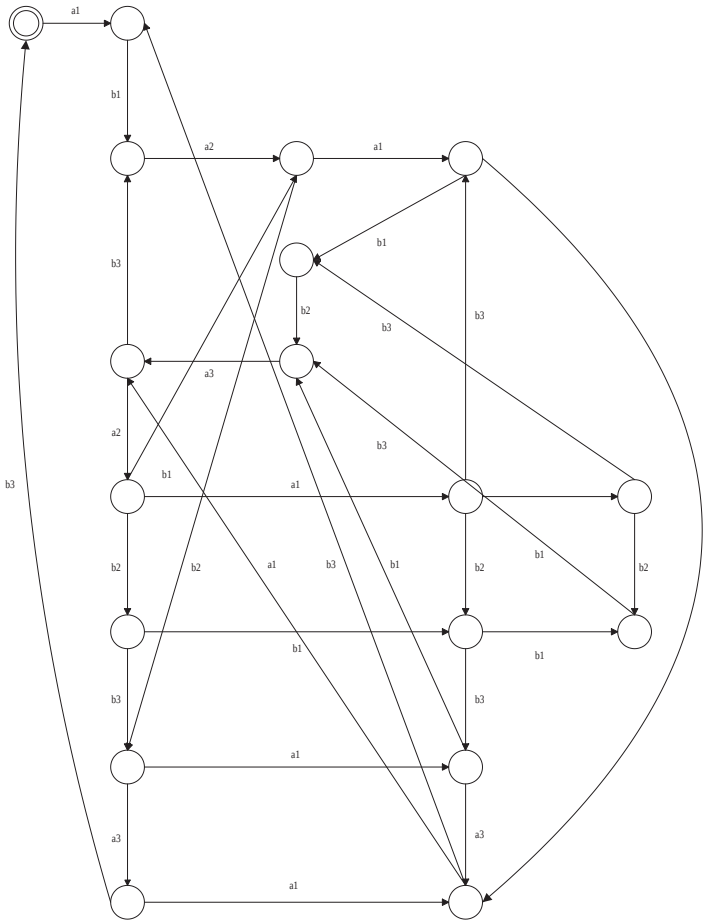

Figura 2.11: Gerador para $SupC(E,G)$

2.4.1 Técnicas de Verificação

Algumas importantes técnicas baseadas em métodos formais incluem simulação, testes, verificação formal e *model checking* [55]. A simulação é uma técnica que faz uso de cenários gerados randômicamente por um usuário ou ferramenta. Normalmente é útil para uma classificação inicial da qualidade de um sistema, mas é pouco aconselhada para detecção de falhas sutis. Isto se justifica pela falta de capacidade (ou até impossibilidade) de simular todos os cenários representativos.

Os testes consistem na forma mais peculiar de detecção de erros. O princípio é semelhante ao da simulação, com a diferença de utilizar sistemas reais ao invés de modelos. Por possuir a mesma restrição da técnica anterior, os testes permitem apenas mostrar a presença de erros, nunca sua ausência.

A verificação formal é uma técnica que requer um modelo do sistema, um método de especificação para expressar os requerimentos de modo formal e um conjunto de regras de prova para determinar se um modelo satisfaz os requerimentos declarados. A idéia básica é provar a conformidade de programas em um nível sintático. Uma das maiores dificuldades nesta abordagem é encontrar invariantes apropriados, o que dificulta a automação destas provas.

Model Checking é uma categoria específica de verificação formal que possui menor interação

com o usuário durante o processo. Originalmente cunhada por Clarke and Emerson [22], *model checking* é uma técnica automática que, dados um modelo em estados finitos de um sistema e uma propriedade lógica, inspeciona sistematicamente se esta propriedade é verdadeira para aquele modelo. Normalmente é realizada uma busca exaustiva em espaço de estados que possui garantia de término se o modelo é finito. Quando um erro é reconhecido, a ferramenta proporciona um contra-exemplo, consistindo de um cenário no qual o modelo se comportou de forma indesejada. Duas alternativas bastante utilizadas para expressar as propriedades lógicas são lógica temporal CTL* e μ-calculus.

O principal obstáculo encontrado pelos algoritmos de *model checking* é o problema de explosão de estados. Muitas vezes esta ferramenta se torna inviável, pois o número de estados excede a quantidade de memória disponível no computador. Alguns métodos têm sido desenvolvidos para combater este problema. Entre eles, pode ser citados [55]:

- representação simbólica do espaço de estados usando diagramas de decisão binárias;
- estratégias eficientes de gerenciamento de memória;
- redução de ordem parcial;
- redução através de equivalências e pré-ordem;
- verificação composicional;
- interpretação abstrata.

Devido ao crescimento da utilização de *model checking* na indústria, algumas companhias têm desenvolvido suas próprias ferramentas ou iniciaram grupos de pesquisa (Siemens, Lucent Technologies, Intel, etc). Casos de estudo de dois grandes projetos que se beneficiaram desta técnica de verificação são descritos em [24] e [49]. Alguns exemplos de ferramentas são SMV, SPIN, DESIGN-CPN, UPPAAL, KRONOS, HYTECH, MEC, Acheck e Averest. Uma breve comparação entre algumas técnicas e ferramentas utilizadas na detecção de deadlock pode ser vista em [8].

2.4.2 Propriedades Normalmente Verificadas

Existe uma classificação hierárquica clássica das propriedades normalmente verificadas em um SED. Tal classificação divide as propriedades em quatro grandes famílias: propriedades de alcançabilidade (*reachability*), de segurança (*safety*), de vivacidade (*liveness*) e de justiça (*fairness*) [18]. Estas famílias levam a técnicas e heurísticas particulares para serem resolvidas, o que ressalta a importância do projetista em saber classificá-las. Segue adiante a definição de cada tipo de propriedade:

- Alcançabilidade: alguma situação particular será alcançada.
- Segurança: sob certas condições, algo nunca ocorre.
- Vivacidade: sob certas condições, algo virá por acontecer.

- Justiça: sob certas condições, algo irá (ou não) ocorrer com freqüência infinita.

É importante ressaltar algumas propriedades podem possuir características de mais de uma família. Por exemplo, pode ser facilmente provado que a negação de uma propriedade de alcançabilidade é uma propriedade de segurança, e vice-versa [18]. Outra propriedade de extrema importância é a ausência de *deadlock*. Tal propriedade expressa que o sistema nunca poderá estar em uma situação na qual nenhum progresso é possível, ou seja, independentemente do estado alcançado, sempre existirá um estado sucessor imediato. Embora possa parecer, a ausência de *deadlock* não é uma propriedade de segurança. Umas das justificativas para tal é que a remoção de transições de um autômato preserva as propriedades de segurança, o mesmo não ocorrendo com a ausência de *deadlock* [18].

2.4.3 Lógica Temporal

Sugerida por Pnueli [77], lógica temporal é uma forma de lógica designada para afirmações que envolvem a noção de ordem no tempo. Embora seja mais clara e simples que outras formas de lógica, alguma experiência é requerida para escrever afirmações em lógica temporal, e mais experiência ainda é necessária para interpretar afirmações escritas por terceiros. Como toda linguagem, possui uma sintaxe e uma semântica.

Sintaxe da CTL*

Introduzida por Emerson and Halpern [32], CTL* (*Computation Tree Logic*) é uma forma de lógica temporal amplamente utilizada para verificação de propriedades relativas a execuções (seqüencia de estados) de um sistema. A lógica temporal CTL* é composta basicamente de proposições atômicas, operadores booleanos, combinadores temporais e quantificadores de caminho.

Proposições atômicas são utilizadas para realizar afirmações sobre os estados. Estas proposições são afirmações elementais que possuem um valor verdadeiro bem definido em cada estado.

Os operadores booleanos consistem nas constantes *verdadeiro* e *falso*, bem como nos tradicionais operadores $\neg$ (negação), $\wedge$ (conjunção, *e*), $\vee$ (disjunção, *ou*), $\Rightarrow$ (implicação lógica, *se...então*) e $\Leftrightarrow$ (dupla implicação, *se e somente se*). Ao envolver várias sub-fórmulas, estes operadores permitem a construção de complexas afirmações.

Os combinadores temporais são utilizados para expressar propriedades durante uma execução, ao invés de propriedades individuais dos estados. Os principais combinadores temporais são explicados em função das propriedades atômicas φ e ψ:

- X (*next*): $X\varphi$ expressa que o próximo estado satisfaz φ.
- F (*future*): $F\varphi$ expressa que um estado futuro satisfaz φ, sem especificar qual estado. Informalmente, F pode ser lido como *algum dia* ou *pelo menos uma vez*.

- G (*global*): $G\varphi$ expressa que todos os estados futuros satisfazem φ. Informalmente, G pode ser lido como *será sempre*.
- U (*until*): $\varphi U \psi$ expressa que φ será verificado até ψ ser verificado.
- W (*weak until*): $\varphi W \psi$ expressa a mesma idéia de U, porém sem obrigar a ocorrência de ψ. Informalmente, $\varphi W \psi$ expressa φ *ocorre enquanto não ocorrer* ψ .

Algumas equivalências são facilmente observadas, tais como: $G\varphi \equiv \neg F \neg \varphi$, $F\varphi \equiv trueU\varphi$ e $\varphi W \psi \equiv (\varphi U \psi) \vee G\varphi$. Além disso, a concatenação de F e G normalmente é usada para expressar propriedades de repetição. $GF\varphi$ (ou $\overset{\infty}{F}\varphi$)[14] indica que φ ocorre freqüentemente durante uma execução. Informalmente, $\overset{\infty}{F}$ pode ser interpretado como *sempre, em algum dia, existirá um estado que satisfaça* φ. Já $FG\varphi$ (ou $\overset{\infty}{G}\varphi$) indica que a partir de um determinado momento, φ será satisfeita por todos os estados, ou seja, φ ocorrerá em cada instante de tempo, possivelmente excluindo um número finito de instantes anteriores.

Completando a sintaxe, a lógica CTL* também possui operadores para quantificar sobre conjuntos de execuções. Os dois quantificadores de caminho são explicados em função da propriedade atômica φ:

- A (*always*): $A\varphi$ expressa que todas as execuções a partir do estado atual satisfazem φ.
- E (*exists*): $E\varphi$ expressa que, a partir do estado atual, existe uma execução que satisfaz φ.

Estes quantificadores são duais, o que pode ser comprovado pela equivalência $A\varphi \equiv \neg E \neg \varphi$. É importante ressaltar que A e E se referem a caminhos, enquanto F e G se referem a posições em um dado caminho. A figura 2.12, retirada de [18], ilustra claramente este conceito.

As propriedades normalmente verificadas em um SED podem ser traduzidas em expressões CTL*, dada uma proposição atômica φ.

- Alcançabilidade: $EF\varphi$.
- Segurança: $AG\neg\varphi$.
- Vivacidade: $F\varphi$.
- Justiça: $\overset{\infty}{F}\varphi$.
- Ausência de deadlock: $AGEXtrue$.

[14] Na literatura, $\overset{\infty}{F}$ é conhecido como *infinitely often*

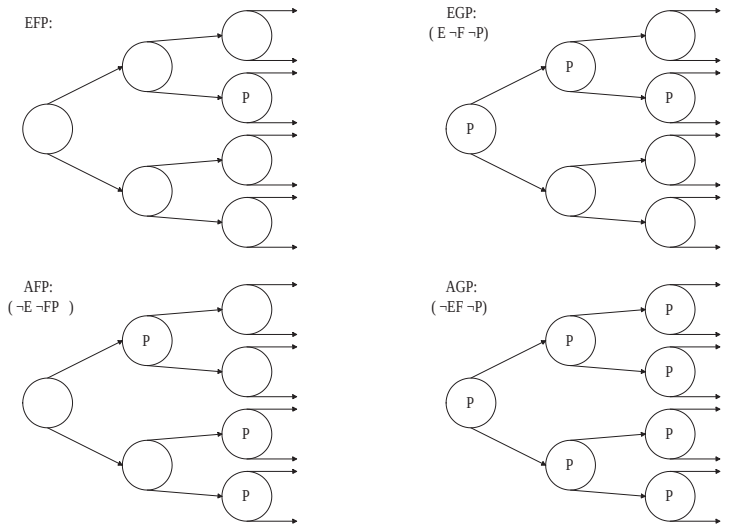

Figura 2.12: Combinações entre os quantificadores de caminho e combinadores temporais da lógica CTL*

Semântica da CTL*

Os modelos utilizados na lógica temporal são denominados estruturas de Kripke. Uma estrutura de Kripke é uma categoria de autômato na qual as proposições que etiquetam os estados possuem maior importância sobre as etiquetas das transições [15]. Uma estrutura de Kripke sobre um conjunto de propriedades atômicas P é uma quádrupla

$$K =< S, s_0, R, \mathcal{L} > \tag{2.6}$$

onde S é um conjunto finito de estados, $s_0 \in S$ é o estado inicial, $R \subseteq (Q \times Q)$ é um conjunto finito de relações de transição e $\mathcal{L} : S \rightarrow P$ é uma função que atribui o conjunto $\mathcal{L}(s)$ de propriedades atômicas a cada um dos estados $s \in S$. Na estrutura de Kripke, tanto a estrutura do autômato, quanto as proposições que etiquetam seus estados são elaboradas simultaneamente, como parte do mesmo processo de modelagem.

A idéia de que uma fórmula é satisfeita em uma dada situação é denotada pela expressão $K, \sigma, i \models \phi$. Esta expressão é interpretada como ϕ é verdadeiro no instante i da execução σ[16]. Para indicar que ϕ não é satisfeita no instante i por σ, escreve-se $K, \sigma, i \not\models \phi$. Normalmente, o contexto da estrutura de Kripke é omitido.

Uma estrutura K satisfaz ϕ sse, $\forall \sigma \in K$, $\sigma, 0 \models \phi$. Esta afirmação e sua negação são denotadas por $K \models \phi$ e $K \not\models \phi$, respectivamente. No entanto, $K \not\models \phi$ não implica necessariamente em $K \models \neg\phi$,

[15] Esta definição está de acordo com o fato da lógica CTL* ser baseada em estados. Também existe variantes da CTL* baseadas em ações e que são utilizadas em autômatos nos quais as etiquetas das transições são mais relevantes. De Nicola and Vaandrager [29] provaram que os dois pontos de vista são bastante similares. A ferramenta MEC4, utilizada na metodologia a ser apresentada, trabalha simultaneamente com ambas [4]

[16] Na CTL*, o tempo é discreto e, portanto, nada existe entre i e $i+1$. Além disso, $\sigma(i)$ indica o i-ésimo estado da execução σ

embora $\sigma, i \not\models \phi$ seja equivalente a $\sigma, i \not\models \neg\phi$ [18]. Em palavras, a primeira expressão é verdadeira quando uma (ou mais execuções) não satisfaz ϕ, enquanto a última expressa que todas execuções de K não satisfazem ϕ.

PLTL e CTL

Na prática, somente alguns fragmentos da lógica CTL* são implementados pelas ferramentas disponíveis. Dois dos fragmentos bastante utilizados são PLTL (*Propositional Linear Temporal Logic*) [78] e CTL(*Computation Tree Logic*) [22] [32].

PLTL é o fragmento obtido da CTL* ao retirar os quantificadores A e E. Desta forma, PLTL somente trata com o conjunto de execuções e não com o modo no qual estes estão organizados em uma árvore. Por este motivo, suas expressões são denominadas fórmulas de caminho.

Por outro lado, CTL é o fragmento obtido da CTL* pela imposição de que o uso de um combinador temporal esteja no escopo imediato de um quantificador de caminho, o que impossibilita o uso de expressões como FG e GF. Na CTL é impossível concatenar diversos combinadores temporais e continuar se referindo a uma mesma execução. Como conseqüência, suas as expressões são denominadas fórmulas de estado.

Ambas as lógicas possuem vantagens e desvantagens ao serem utilizadas. Na prática, as restrições impostas a sintaxe CTL deixam-a mais difícil para expressar propriedades. Porém, *model checking* em CTL é muito mais eficiente que *model checking* em PLTL [18]. *Model checking* '$K, s_0 \models \phi$' para uma fórmula PLTL ϕ pode ser realizado em tempo $O(|K| \times 2^{|\phi|})$ [63] [92] [93], enquanto *model checking* '$K, s_0 \models \phi$' para uma fórmula CTL ϕ pode ser realizado em tempo $O(|K| \times |\phi|)$ [79] [23]

2.4.4 μ-calculus

μ-calculus é uma lógica de especificação cujos algoritmos formam a base do estado-de-arte das atuais ferramentas de verificação [87]. Bhat and Cleaveland [14] provaram que a lógica CTL* pode ser eficientemente traduzida para expressões em μ-calculus. Para explicar a sintaxe e a semântica desta lógica, é necessário rever alguns conceitos básicos do cálculo de pontos fixos.

Cálculo de pontos fixos

Os pontos fixos são bastante úteis para descrever propriedades de entidades matemáticas discretas. Notações para pontos fixos extremos de operadores monotônicos foram introduzidas por diversos autores [59].

Um operador $f : 2^X \rightarrow 2^X$ sobre o conjunto potência 2^X é dito monotônico se $\forall x_i, x_j \subseteq x$, $x_i \subseteq x_j \Rightarrow f(x_i) \subseteq f(x_j)$. Tal operador possui pontos fixos mínimo e máximo, que são as soluções de $x \stackrel{\mu}{=} f(x)$ e $x \stackrel{\nu}{=} f(x)$, respectivamente. Os símbolos $\stackrel{\mu}{=}$ e $\stackrel{\nu}{=}$ indicam que a busca de menor e maior valores de x satisfazem estas equações. As soluções, denotadas por $\mu x.f(x)$ e $\nu x.f(x)$, satisfazem

$$\mu x.f(x) = \cap\{x \subseteq X : x = f(x)\}, \tag{2.7}$$

$$\nu x.f(x) = \cup\{x \subseteq X : x = f(x)\}. \tag{2.8}$$

Como X é finito e f é monotônico, o ponto fixo mínimo pode ser encontrado por uma iteração iniciando-se com $x_0 = \emptyset$ e calculando $x_{i+1} = f(x_i)$ até convergir para algum valor. O ponto fixo máximo pode ser obtido de forma análoga, porém iniciando-se com $x_0 = X$.

Sintaxe do μ-calculus

Dado um conjunto de propriedade atômicas P, o conjunto de fórmulas em μ-calculus sobre P é definido como o menor conjunto $\mathcal{L}_\mu$ que satisfaz as seguintes regras:

- $P \cup \{0,1\} \subseteq \mathcal{L}_\mu$,
- $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi \in \mathcal{L}_\mu$, desde que $\varphi, \psi \in \mathcal{L}_\mu$,
- $EX\varphi$, $EX^{-1}\varphi$, $AX\varphi$, $AX^{-1}\varphi \in \mathcal{L}_\mu$ [17], desde que $\varphi \in \mathcal{L}_\mu$,
- $\mu x.\varphi \in \mathcal{L}_\mu$, desde que $\varphi \in \mathcal{L}_\mu$,
- $\nu x.\varphi \in \mathcal{L}_\mu$, desde que $\varphi \in \mathcal{L}_\mu$.

Os operadores booleanos, combinadores temporais e quantificadores de caminho são os mesmos descritos na sintaxe da CTL*, com a exceção do combinador temporal X^{-1}. Este operador expressa idéia oposta a X, ou seja, $X^{-1}\varphi$ expressa que o estado anterior satisfaz φ.

Semântica do μ-calculus

Assim como na CTL*, a semântica do μ-calculus também é definida sobre estruturas de Kripke. Dada uma estrutura de Kripke $K =< S, s_0, R, \mathcal{L} >$ sobre um conjunto de propriedades atômicas P, associa-se a cada fórmula $\Phi \in \mathcal{L}_\mu$ um conjunto de estados $[[\Phi]]_K \subseteq S$ por meio das seguintes regras:

- $[[0]]_K = \{\}$
- $[[1]]_K = S$
- $[[p]]_K = \{s \in S \mid p \in \mathcal{L}(s)\}$ para todas propriedades atômicas $p \in P$

[17] Em algumas referências [102] [71], foram encontrados outros símbolos para representar os operadores modais (EX, EX^{-1}, AX e EX^{-1}), tais como $\Diamond$ e $\Box$. Devido à falta de padronização, optou-se por não utilizar nenhum destes símbolos neste documento.

- $[[\neg\varphi]]_K = S - [[\varphi]]_K$
- $[[\varphi \wedge \psi]]_K := [[\varphi]]_K \cap [[\psi]]_K$
- $[[\varphi \vee \psi]]_K = [[\varphi]]_K \cup [[\psi]]_K$
- $[[EX\varphi]]_K = \{s \in S \mid \exists s' \in S.R(s,s') \wedge s' \in [[\varphi]]_K\}$
- $[[EX^{-1}\varphi]]_K = \{s' \in S \mid \exists s \in S.R(s,s') \wedge s \in [[\varphi]]_K\}$
- $[[AX\varphi]]_K = \{s \in S \mid \forall s' \in S.R(s,s') \Rightarrow s' \in [[\varphi]]_K\}$
- $[[AX^{-1}\varphi]]_K = \{s' \in S \mid \forall s \in S.R(s,s') \Rightarrow s \in [[\varphi]]_K\}$
- $[[\mu p.\varphi]]_K = \cap\{Q \subseteq S \mid [[\varphi]]_{K_p^Q} \subseteq Q\}$, onde K_p^Q é obtido pela troca da função $\mathcal{L}$ de K por outra função cujos estados em Q sejam etiquetados com p
- $[[\nu p.\varphi]]_K = \cup\{Q \subseteq S \mid [[\varphi]]_{K_p^Q} \subseteq Q\}$, onde K_p^Q é definido como acima

Dada uma fórmula φ e um estado $s \in [[\varphi]]_K$, conclui-se que s satisfaz φ, denotado por $s \models \varphi$. Analogamente, uma estrutura de Kripke K satisfaz φ, denotado por $K \models \varphi$, quando $\forall s \in s_0.s \models \varphi$.

$[[EX\varphi]]$ corresponde ao conjunto de estados que possuem pelo menos uma transição de saída que aponte para algum estado em $[[\varphi]]$. Tais estados são chamados de predecessores existenciais de $[[\varphi]]$. Analogamente, os estados em $[[EX^{-1}\varphi]]$ são chamados sucessores existenciais de $[[\varphi]]$.

$[[AX\varphi]]$ corresponde ao conjunto de estados cujas todas transições de saída apontam para algum estado em $[[\varphi]]$. Tais estados são chamados de predecessores universais de $[[\varphi]]$. Analogamente, os estados em $[[AX^{-1}\varphi]]$ são chamados sucessores universais de $[[\varphi]]$.

Caso φ seja uma função monotônica em p, então $\mu p.\varphi$ ($\nu p.\varphi$) é o seu menor (maior) ponto fixo [91]. Assim como na CTL* algumas equivalências são observadas, tais como $AX\varphi \equiv \neg EX\neg\varphi$, $AX^{-1}\varphi \equiv \neg EX^{-1}\neg\varphi$ e $\mu p.\varphi(p) \equiv \neg\nu p.\neg\varphi(\neg p)$.

Quando uma expressão ou sistema de equações envolvendo múltiplos operadores μ e ν é avaliada, cada grupo de sucessivos operadores do mesmo tipo (μ ou ν) pode ser calculado dentro de um loop simples. Entretanto, quando pontos fixos de tipos diferentes são concatenados de forma a causar dependência mútua, as iterações também são concatenadas, o que acarreta em aumento de complexidade. Uma noção que captura este grau de dependência é a profundidade de alternação (*alternation depth*) de uma expressão ou sistema de equações em pontos fixos. Intuitivamente falando [102], a profundidade de alternação de uma expressão com operadores μ e ν concatenados é dada pelo número de pontos fixos mutuamente dependentes, com a exceção de que pontos fixos sucessivos do mesmo tipo são contados somente uma vez. Quando uma expressão é transformada para a forma de sistemas de equações, cada equação possui somente um ponto fixo. A ordem na qual as equações aparecem no sistema está de acordo com a forma como os operadores μ e ν estão concatenados na expressão. Neste caso, a profundidade de alternação é igual ao número de equações mutuamente dependentes, com a exceção de que equações sucessivas com o mesmo tipo de ponto fixo são contados somente

uma vez. Expressões e sistemas de equações com um único tipo de ponto fixo possuem profundidade de alternação unitária e são chamados de sem-alternação (*alternation-free*). Definições formais podem ser vistas em [31] e [72].

Dada uma estrutura de Kripke $K =< S, s_0, R, \mathcal{L} >$ e uma expressão (ou sistema de equações) E de profundidade de alternação l e comprimento $|E|$ sobre K, *model checking* em μ-calculus pode ser realizado em tempo $O((\frac{|E||S|}{l})^{l-1}|R||E|)$ [25] [87]. O comprimento |E| de uma expressão é dada pelo número de símbolos que ela contém. Já o comprimento de um sistema de equações é obtido pela adição dos comprimentos individuais de todas as equações constituintes.

2.5 Discussão

Foram apresentados todos os formalismos necessários para o entendimento dos capítulos seguintes. Com relação às redes de Petri, foram descritas suas principais características. Existem diversas extensões na literatura, que consideram tanto aspectos relevantes durante a síntese de supervisores (ou simplesmente leis de controle), quanto aspectos temporais. Dentro do primeiro contexto, o Capítulo 3 apresenta resumidamente duas extensões bastante encontradas na literatura, as redes de Petri etiquetadas e as redes de Petri controladas. Também será visto que os invariantes de lugar também podem ser utilizados para a síntese de supervisores, indo de encontro à sua finalidade original como informação para análise. O Capítulo 6, por sua vez, apresenta os aspectos relevantes das redes de Petri temporais.

Os principais conceitos da Teoria de Linguagens e da Teoria de Controle Supervisório são apresentados de forma resumida, porém suficiente para a compreensão do controle monolítico. Caso alguns aspectos não foram satisfatoriamente descritos, estes serão melhor explicitados no capítulo a seguir.

Por último, talvez foi dada uma atenção maior do que a necessária à verificação de sistemas. Embora a metodologia proposta nos Capítulo 4 faça uso direto de sistemas de equações escritas em μ-calculus, optou-se por descrever com mais detalhes aspectos referentes às propriedades normalmente verificadas e à lógica temporal CTL*. O principal objetivo desta extensão foi deixar claro como as técnicas de verificação são inicialmente utilizadas para a resolução de problemas distintos daqueles tratados pelas técnicas de síntese de supervisores em SEDs.

Capítulo 3

Diferentes Abordagens ao Controle Supervisório

Este capítulo discute diversas abordagens alternativas ao PCS de SEDs que utilizam como modelos estruturas em RdP e algumas de suas extensões. Em um primeiro momento, são apresentadas duas formas distintas de se classificar as diversas metodologias, uma que leva em consideração até quando o comportamento em malha fechada deve ser restrito [39] e outra que considera a forma como a realimentação é realizada [48]. Posteriormente, alguns trabalhos são apresentados em ordem cronológica.

As seções seguintes apresentam duas recentes metodologias que formam os pilares para as contribuições apresentadas neste documento. Ghaffari et al. [35] utilizam como base sistemas modelados por RdP e fazem uso da abordagem Ramadge and Wonham [82] [1989] e da Teoria das Regiões [6]. Ziller and Schneider [100], por sua vez, também se baseiam na abordagem Ramadge and Wonham [82] [1989], mas resolvem o PCS utilizando-se sistemas modelados por autômatos e expressões lógicas expressas em μ-calculus. Estas duas metodologias são apresentadas em profundidade, pois o entendimento das mesmas se faz necessário para a compreensão da metodologia proposta neste trabalho.

3.1 Abordagens utilizando sistemas modelados por redes de Petri

Existem diversas classificações sobre as abordagens de resolução do PCS em sistemas modelados por RdP. Dentre estas classificações, duas merecem destaque. Em [48], as abordagens são divididas em controle por realimentação de estados e controle por realimentação de eventos. Já em [39], as políticas de controle são divididas em políticas maximamente permissivas e políticas mais restritivas, porém calculadas mais facilmente.

O controle por realimentação de estados foi inicialmente estudado para um modelo denominado RdP controlada, introduzido por Krough [56] e Ichikawa and Hiraishi [51], com pequenas diferenças

nas definições envolvidas. RdP controladas são uma classe de RdP com condições de habilitação externas denominadas entradas de controle [1], que permitem um controlador externo influenciar a progressão das fichas na rede. Formalmente, uma RdP controlada é uma tripla:

$$N_c =< N, C, \beta > \tag{3.1}$$

onde N é uma RdP, C é um conjunto finito de entradas de controle e $\beta \subseteq (C \times T)$ é uma função que define arcos das entradas de controle às transições. Neste contexto, os lugares da RdP N são denominados de lugares de estado, enquanto uma transição é dita controlada caso exista pelo menos um arco ligando esta transição a uma entrada de controle. Nesta abordagem, as especificações são dadas como um conjunto de marcações admissíveis do SED a ser controlado e a intenção de controle é restringir seu comportamento tal forma que somente estas marcações sejam alcançadas. Após ter caracterizado a política de realimentação de estados maximamente permissiva, uma questão que surge é como calcular o conjunto de controles admissíveis para cada marcação da RdP controlada. Uma abordagem é simplesmente criar um autômato equivalente à RdP controlada, através da generalização do grafo de alcançabilidade da estrutura da RdP com a informação de controle associada, e então aplicar algoritmos padrões da abordagem Ramadge and Wonham [82] para resolver o PCS. Como o grafo de alcançabilidade pode crescer exponencialmente com relação às dimensões do modelo em RdP controlada, foram desenvolvidas abordagens eficientes que utilizam a estrutura do modelo em RdP controlada diretamente. Duas destas alternativas são a abordagem de programação linear inteira, que tira vantagem da estrutura linear das equações de transição de estado da RdP controlada, e a abordagem de algoritmos baseados em caminho, que tira vantagem da estrutura gráfica dos modelos em RdP controlada. Algumas políticas de realimentação de estados em RdP controladas foram investigadas em [45], [44], [61] e [43].

O controle por realimentação de eventos foi principalmente considerado no contexto de linguagens formais e o modelo envolvido é denominado RdP etiquetado [75] [54]. Em uma RdP etiquetada [2], o disparo de uma transição corresponde a um evento na terminologia usual dos SEDs e o conjunto de todas seqüências de eventos admissíveis é chamada linguagem da RdP. A noção de linguagem de uma rede é útil na especificação e análise de problemas de controle seqüencial. Formalmente, uma RdP etiquetada é uma quíntupla:

$$N_e =< N, \Sigma, l, m_0, F > \tag{3.2}$$

onde N é uma RdP, Σ é um alfabeto de eventos, $l : T \rightarrow \Sigma$ é uma função que associa um evento (incluindo a cadeia vazia ε) a cada transição, $m_0 \in M$ é a marcação inicial e $F \subset M$ é um conjunto finito de marcações finais. As duas principais linguagens associadas à RdP etiquetada são as linguagens do *tipo-P* e do *tipo-L*, que correspondem respectivamente às linguagens gerada e marcada no contexto do

[1] As entradas de controle também são chamadas de lugares de controle. Neste documento, optou-se por utilizar a primeira terminologia, pois o termo lugares de controle será utilizado para outras finalidades.

[2] A RdP etiquetada também é chamada de gerador em RdP

controle supervisório. Também é definida uma linguagem do *tipo-G* (também chamada de linguagem fraca) que define uma noção diferente do comportamento final. Nestas redes, somente um evento pode ocorrer a cada instante e existem casos onde nenhuma destas linguagens é regular, ou seja, não pode ser gerada por um autômato de estados finitos [75]. Dentre as diversas abordagens de realimentação de estados, três são bastante citadas na literatura. Em Guia and DiCesare [40] discutem a teoria de linguagens em RdP e estendem algumas das idéias da abordagem Ramadge and Wonham [82], desenvolvidas inicialmente sobre autômatos. Guia [39] apresenta diversas propriedades das linguagens das RdP etiquetadas e deriva importantes resultados sobre as classes de problemas de controle que podem ser efetivamente solucionadas. A existência e projeto de supervisores para RdP etiquetadas sujeitas a comportamentos regulares admissíveis foram discutidas por Kumar and Holloway [58].

As abordagens de resolução do PCS em sistemas modelados por RdP também podem ser classificadas de acordo com as medidas tomadas para a redução da complexidade computacional [39]. Por um lado, pode-se considerar estruturas especiais de RdP cujas políticas de controle maximamente permissivas podem ser calculadas e implementadas. Existem diversas abordagens interessantes neste sentido, podendo ser citados os trabalhos de Holloway and Krogh [45] [1991], Guia et al. [42] e Li and Wonham [61] [1994].

Por outro lado, pode-se desistir da requisição de que a política de controle seja maximamente permissiva e escolher, portanto, uma política de controle mais restritiva, mas que seja facilmente calculada. Dentre outros autores, esta abordagem foi seguida por Moody and Antsanklis [69], Moody et al. [68] e Yamalidou et al. [98]. Nesta abordagem, a idéia geral é utilizar supervisores bem simples na forma de lugares de controle [3] que são adicionados ao modelo da planta em RdP e que restringem somente transições controláveis.

3.1.1 Apresentação cronológica de algumas abordagens

Holloway and Krogh [45] [1991] comparam a abordagem por invariantes de lugar [4] com o tipo de restrição mais genérica que pode ser definida sobre o conjunto de marcações de um sistema, a restrição de marcações proibidas. Foi observado que nem sempre é possível existir um invariante de lugar equivalente a uma restrição de marcações proibidas. Os SEDs são modelados como grafo marcados controlados cíclicos e seguros e a marcação das entradas de controle é calculada por um agente externo em uma função da marcação dos lugares de estado. Comparando-se a uma RdP convencional, existem duas diferenças com relação à política de disparo: duas transições habilitadas podem disparar simultâneamente e as entradas de controle permitem interpretar paralelismo, ou seja, se uma entrada de controle conectada a duas ou mais transições está marcada [5], então todas suas transições de saída podem disparar caso elas também estejam habilitadas pela marcação dos lugares de estado da RdP

[3] Lugares de controle também são chamados de monitores. Na literatura, esta terminologia é utilizada tanto para uma classe específica de lugares das RdP controladas, quanto para os lugares que são adicionados a uma RdP convencional com o objetivo de supervisão.

[4] Na resolução do PCS, os invariantes de lugar não são utilizados para analisar, e sim para sintetizar controladores que garantam restrições lineares no comportamento das marcações da RdP que será controlada.

[5] Assume-se que todas as entradas de controle são binárias

controlada. A lei de controle é calculada em duas etapas e a abordagem é extremamente eficiente, devido à grande simplicidade dos cálculos envolvidos. Entretanto, como o o controlador é dado por uma lei de realimentação, não é possível construir um modelo em RdP do comportamento em malha fechada. Esta abordagem recebeu uma atenção considerável na literatura e foi também estendida para outras classes de redes além de grafos marcados. Condições necessárias e suficientes para vivacidade sob controle foram apresentadas em [47].

Yamalidou and Kantor [97] e Yamalidou et al. [96] apresentaram uma formulação baseada em otimização linear, cujo objetivo era encontrar a seqüencia de disparo que levasse o sistema de um estado inicial a um estado final especificado, ao mesmo tempo que minimizando uma função de custo. A função de custo era relacionada com a marcação e com os vetores de disparo em cada etapa. As restrições incluíam dois conjuntos: restrições dinâmicas e operacionais. O primeiro conjunto era imposto pela dinâmica do modelo em RdP, enquanto o último era derivado de expressões booleanas sobre as restrições operacionais do sistema.

Guia et al. [41] [1993] definiram restrição generalizada de exclusão mútua (RGEM) como uma condição que limita a soma ponderada de fichas contidas em um subconjunto de lugares [6] e realizaram suas considerações sobre grafos marcados com lugares seguros de controle. Uma transição t pertence ao conjunto de transições de controle A_p de um lugar p sse t é controlável e existe um caminho de t a p que não contem transições controláveis. Um lugar p é seguro de controle a quantidade de fichas não excede o valor unitário em pelo menos um caminho de cada $t \in A_p$ a p. Uma transição $t \in A_p$ é dita ser restritiva em uma marcação M se não existe um caminho de t a p que não está marcado em M. Para esta classe de redes e restrições, Guia et al. [42] mostram que uma lei de controle maximamente permissiva sempre pode ser implementada pela adição de um conjunto de lugares de controle ao modelo da planta em RdP. Em [41], também é provado que, na presença de transições não controláveis, um problema de exclusão mútua é transformado em um problema genérico de marcações proibidas. Neste caso, arcos dos lugares de controle às transições não-controláveis não são permitidos, desde que o efeito de tais arcos poderia ser de prevenir o disparo da transição quando o lugar de controle não estiver marcado. Em particular, foi mostrado que se o conjunto L de marcações admissíveis é expresso por um conjunto de restrições em inequações lineares e se L é controlável, então a solução baseada em RdP existe e é maximamente permissiva. Outro importante resultado é que, para as RdPs seguras, existe um conjunto de RGEM equivalente a qualquer restrição de marcações proibidas. Por último, é também observado que para sistemas seguros e conservativos, uma RGEM pode ser satisfeita por um conjunto de lugares de controle mesmo se algumas transições são não-controláveis.

Em uma abordagem inicial, Li and Wonham [61] [1994] consideraram SEDs de Vetor, um modelo que é conhecido ser equivalente a uma RdP [75], e fizeram uso da análise da matriz de incidência para calcular a lei de controle que garantisse as restrições, denominadas predicados lineares. Eles desenvolveram um problema de programação inteira que deve ser solucionado como parte de um algoritmo para assegurar uma dada restrição em redes cuja sub-rede não-controlável é acíclica. Esta condição

[6] Assim como foi definida, uma restrição generalizada de exclusão mútua corrreponde a um invariante de lugar de uma RdP. Esta é uma forma natural de expressar a utilização concorrente de um número finito de recursos, compartilhados entre diferentes processos. O uso de pesos permite atribuir diferentes unidades de recursos aos vários processos. A utilização de pesos negativos permite expressar restrições de justiça na alocação de recursos compartilhados.

não é muito restritiva, no sentido em que é válida em muitas aplicações práticas. Entretanto, a grande desvantagem é que o controlador deve resolver em tempo real diversos problemas de programação inteira em cada mudança de estado da rede, o que torna a abordagem infactível em problemas de grandes dimensões. Isto levou os autores a estudar outras classes de redes na quais o controlador poderia ser representado por uma RdP. Na segunda abordagem, uma importante restrição foi a imposição de que a rede não-controlável fosse composta de componentes não conectados e que dois ou mais lugares envolvidos na restrição não pertencessem ao mesmo componente (independência mútua). O método pôde então ser aplicado a redes cujas transições não controláveis formam estruturas em árvore de dois tipos, definidos em [62]. No caso onde cada transição possui um único arco de entrada, foi mostrado que existe um lugar de controle que não possui arcos a transições não-controláveis. No caso onde cada transição possui um único arco de saída, mostrou-se que a restrição original é equivalente à disjunção de um conjunto de restrições e determinou-se como calcular um lugar de controle que garante que a solução para cada restrição não possua arcos a transições não-controláveis. No entanto, o comportamento global dos lugares de controle não pode ser representado por uma rede, pois a adição de cada lugar equivale a uma operação de conjunção. Para garantir uma disjunção de restrições também foi definido em [62] uma estrutura denominada sistema generalizado de adição de vetores.

Em [15] e [16], as especificações foram expressas em termos de estados proibidos, mais especificamente os problemas de alocação de recursos e prevenção de *deadlock*. O motivo da lei de controle era encontrar uma RdP a ser adicionada à rede original do SED, de forma com que o modelo geral em RdP pudesse prover o comportamento ótimo do sistema. A maior contribuição destes trabalhos foi a resolução do PCS utilizando a técnica de *simulated annealing*.

Sreenivas [88] [1996] [1996] explorou a decidabilidade e a existência de estruturas de controle para a satisfação de diversas classes de especificações sobre RdP que não podem ser facilmente expressas como linguagens. Em [88], condições necessárias e suficientes foram derivadas para a existência de uma política supervisória que satisfaz a vivacidade em RdP parcialmente controladas. Esta condição é decidível se todas as transições são controláveis ou se a RdP é limitada. Entretanto, para redes não-limitadas com transições não-controláveis, a condição se torna não-decidível. Em [89] políticas supervisórias que satisfazem vivacidade em redes *free-choice* não-vivas foram definidas. Mostrou-se também que, se existe uma política supervisória que satisfaz vivacidade, então também existe uma política minimamente restritiva. Por último [90], foram definidos procedimentos decidíveis com dois objetivos principais: verificar a existência de uma política supervisória que garanta justiça global e justiça limitada em RdP parcialmente controladas e verificar a existência de uma política minimamente restritiva que garanta tais condições em RdP limitadas. A classe de políticas não é fechada sob disjunção, e, portanto, uma política minimamente restritiva pode não existir.

Dentro do contexto do controle de coordenação de plantas em batelada, Gu et al. [38] definiram três tipos de predicados para especificar as tarefas envolvidas: predicados seqüenciais, predicados de sincronização e predicados de compartilhamento de recursos. O controlador de coordenação era constituído por outro modelo em RdP e pôde ser projetado automaticamente de acordo com os predicados e com as regras de síntese apresentadas. Ambos os modelos em RdP da planta e do controlador de coordenação são utilizados na implementação do controle de coordenação da planta global em batelada.

Yamalidou et al. [98] utilizaram invariantes de lugar para calcular, através de multiplicação de matrizes, um controlador por realimentação de estados. Este método de controle pode ser aplicado a sistemas cujas restrições são expressas como inequações algébricas ou expressões lógicas que contém elementos dos vetores de marcação e/ou disparo. Para isto, aproveita-se um resultado apresentado em [97], onde foi mostrado como restrições escritas como expressões booleanas podem ser transformadas em conjuntos de inequações lineares envolvendo vetores de disparo e de marcações. Além disso, as restrições que envolvem o vetor de disparo devem ser transformadas em restrições que envolvem somente elementos do vetor de marcações, o que pode ser feito de duas formas diferentes. A primeira forma é baseada em manipulações algébricas nas restrições e pode ser aplicada a transições controláveis e não-controláveis. Em certos casos, a transformação de uma restrição pode gerar diversas outras com a forma apropriada, o que acarreta no aumento do tamanho do controlador. Em outros casos, esta transformação é somente aplicada a RdP seguras. A segunda forma é baseada em uma transformação gráfica e conserva a quantidade de restrições. Sua desvantagem é que ela não pode ser aplicada a transições não controláveis. O controlador resultante em RdP consiste de lugares e arcos a serem adicionados à RdP da planta e seu tamanho é proporcional à quantidade de restrições. Quando todas as transições são controláveis, prova-se que este controlador é maximamente permissivo, embora não seja ótimo em termos de minimização da quantidade de lugares envolvidos. Se as restrições sobre a performance da rede são escritas em termos do vetor de disparo, então existem situações nas quais a máxima permissividade do controlador só é garantida caso a rede seja segura. Além da eficiência computacional, este método não realiza a enumeração de estados durante o processo de síntese. Ao contrário de diversas outras abordagens, também não há restrições quanto a RdP cíclicas ou limitadas. Conseqüentemente, este método de controle é genérico e possui grande potencial para aplicação prática em SEDs complexos e de largas dimensões.

Moody et al. [70] generalizaram o conceito de invariantes de lugar a RdP que possuem transições não controláveis e/ou não-observáveis, mas não garantem a máxima permissividade para o caso geral. Nesta metodologia, não devem existir arcos conectando os lugares de controle às transições não-controláveis, ao mesmo tempo que não devem existir arcos das transições não-observáveis aos lugares de controle. A restrição sobre as transições não observáveis se aplica, pois, como a ocorrência de tais transições não é percebida, elas não podem ser utilizadas por um lugar de controle para atualizar sua respectiva lei. Esta metodologia é apresentada de forma mais aprofundada em Moody and Antsaklis [67].

Em [66], a síntese de supervisores em RdP foi realizada utilizando-se a teoria dos tubos (*siphons*). O método proposto parte do princípio que, para evitar os bloqueios, é necessário prevenir que os tubos sejam esvaziados. Desta forma, uma restrição linear é imposta a cada tubo, obrigando que a quantidade de fichas em seu interior seja no mínimo unitária. Para determinar um lugar de controle para cada restrição linear, o método baseado nos invariantes de lugar proposto por Moody et al. [70] é utilizado em seqüencia. Esta abordagem possui uma séria desvantagem, pois a adição de lugares de controle visando prevenir o esvaziamento dos tubos pode induzir novos bloqueios na RdP [33].

Em [27], a lógica temporal CTL é utilizada para representar o comportamento da planta em malha aberta e para especificar o comportamento desejado em malha fechada. O SED é inicialmente modelado por uma RdP multiplexada, definida em [26]. As RdP multiplexadas possuem a mesma ex-

pressividade de uma RdP e facilitam a utilização da lógica temporal. Comparando-se à definição de uma RdP etiquetada, as diferenças consistem na não existência do conjunto de marcações finais e na inclusão de uma função que associa eventos aos arcos e de uma função binária de habilitação das transições. As funções associadas às transições de uma RdP multiplexada são transformadas em fórmulas CTL e são definidas condições para a validação das especificações de comportamento. São propostas duas alternativas para a realização da síntese do supervisor. A primeira consiste em efetuar a verificação e validação da fórmula que expressa o comportamento desejado em relação à fórmula que descreve o comportamento do SED, enquanto a segunda consiste em construir a árvore de alcançabilidade da RdP multiplexada da planta e verificar nessa árvore a validade da fórmula que expressa o comportamento desejado. Efetuada a verificação e a validação, as sub-fórmulas que constituem o comportamento desejado e que não são factíveis em relação ao comportamento da planta em malha aberta devem ser eliminadas. O supervisor resultante possui a mesma estrutura do modelo do SED, diferenciando-se apenas nas fórmulas CTL associadas às transições.

Iordache and Antsaklis [52] generalizaram os resultados de Yamalidou et al. [98] e Moody et al. [70], ao incluir o vetor de Parikh como um novo termo no conjunto das restrições a serem satisfeitas. A grosso modo, este vetor conta qual a freqüência uma dada transição foi disparada desde a inicialização do sistema. É bem provável que uma das suas principais funções seja descrever requerimentos de justiça, tais como a restrição de que a diferença entre o número de disparo de duas transições seja no máximo unitária. Esta abordagem também pode ser naturalmente estendida para a garantia de restrições envolvendo simultâneamente conjunções e disjunções de inequações lineares. Este trabalho mostrou como estas restrições mais expressivas poderiam ser satisfeitas tão eficientemente quanto a abordagem presente em [98]. Outro importante resultado foi aprimorar a técnica presente em Moody et al. [70] para satisfazer as restrições de vetor de disparos na presença de transições não-controláveis e não-observáveis.

Iordache et al. [53] introduziram um novo procedimento de prevenção de deadlock, mas também utilizarm como ponto de partida os resultados de Yamalidou et al. [98] e Moody et al. [70]. Quando a RdP é supervisionada de tal forma que sua marcação satisfaça um conjunto de inequações lineares, prova-se que esta rede é livre de deadlocks para todas as marcações iniciais que satisfaçam as restrições de supervisão. Embora o procedimento de síntese seja computacionalmente custoso, o supervisor resultante requer poucos recursos computacionais em tempo de execução. O procedimento pode ser utilizado em RdP não-repetitivas, ilimitadas e que possuem transições não-controláveis e/ou não-observáveis, sendo também apropriado para utilização em redes que podem não ser estruturalmente vivas. Quando o procedimento é aplicado em RdP repetitivas, vivacidade pode ser o resultado. É mostrado que neste caso, sob premissas sempre satisfeitas por RdP controláveis e observáveis, o supervisor que garante vivacidade também é menos restritivo possível. Outra importante característica é que o método é flexível o suficiente para incorporar especificações desejadas em termos de restrições das marcações da planta.

Em trabalho recente, Lima and Dórea [64] também se basearam no conceito de invariantes de lugar em RdP ao apresentar os resultados de uma aplicação prática de controle supervisório em uma célula de manufatura. O programa implementado em CLP é escrito na linguagem STL (Statement

List), de uma forma a permitir uma tradução direta para LDR (Ladder Relay Logic) [7]. Ao contrário da maioria das implementações apresentadas nas literatura [34] [74], o modelo em RdP não integra os sinais de baixo nível do CLP, devendo estes ser tratados pelos procedimentos operacionais. O tamanho do programa possui relação direta com o número de lugares e transições do modelo em RdP da planta controlada, o que representa uma prova de sua viabilidade prática comparada a abordagem por autômatos. Além disso, se a RdP é segura, as variáveis envolvidas são todas booleanas, o que reduz consideravelmente a memória utilizada do CLP. Por último, foi verificado na prática que a abordagem baseada em invariantes de lugar garante a obtenção de um supervisor maximamente permissivo para plantas que possuem a mesma estrutura seqüencial, embora seja provado na literatura que esta propriedade não é válida para o caso geral.

3.2 Abordagem Ghaffari et al.

Ghaffari et al. [35] [2003] propõe um método de síntese de supervisores baseados em RdP, para resolver o problema de estados proibidos na presença de transições não-controláveis e satisfazendo a requisição de vivacidade. A metodologia proposta combina a abordagem Ramadge and Wonham [83] e a teoria das regiões, uma teoria recentemente proposta para a síntese de RdP a partir de grafos de alcançabilidade. Após o espaço de estados admissível ser determinado, o supervisor resultante é então obtido pela adição de lugares à RdP original.

3.2.1 Definições

Na abordagem Ghaffari et al. [35], o SED é modelado por uma RdP, enquanto as especificações são modeladas sob a forma de marcações proibidas que correspondem a estados de bloqueio e/ou contradizem restrições de controle. O objetivo é determinar um conjunto de lugares de controle, de tal forma que a RdP em malha fechada não alcance tais estados proibidos.

Dada uma RdP N, uma restrição de controle do tipo estado proibido é uma função $Q : R(N,M_0) \rightarrow \{0,1\}$, onde o valor 0 indica que o estado viola a restrição. O conjunto dos estados proibidos é dado então por $M_F = \{M \in R(N,M_0) \mid Q(M) = 0\}$.

O conjunto das transições é dividido em T_c, o conjunto das transições controláveis, e T_u, o conjunto das transições não-controláveis. Analogamente à abordagem Ramadge and Wonham [83], para impedir que o SED atinja os estados proibidos, é necessário proibir as seqüências de transições não-controláveis que levam a tais estados. Seja N uma RdP, T_u um conjunto de transições não controláveis e M_F um conjunto de estados proibidos, uma marcação acessível M é perigosa sse ela é proibida ou existe uma seqüência de transições não-controláveis de M a qualquer marcação proibida. Formalmente, o conjunto das marcações perigosas é representado por $M_D = \{M \in R(N,M_0) \mid \exists M' \in M_F \wedge \sigma \in T_u^*, M[\sigma \rightarrow M'\}$.

[7] Para maiores detalhes sobre estas linguagens, consultar [60]

A restrição de coacessibilidade também é considerada, mas somente a marcação inicial M_0 é considerada marcada. Desta forma, o conjunto das marcações admissíveis M_A corresponde ao maior conjunto de marcações alcançáveis tal que:

- $M_A \cap M_D = \varnothing$.
- É possível alcançar a marcação inicial a partir de qualquer estado admissível.
- A passagem à uma marcação não-admissível a partir de uma marcação admissível se faz pelo disparo de uma transição controlável.

Caso M_A exista, ele representará o comportamento menos restritivo possível em relação às especificações. O conjunto $M_B = R(N,M_0) - M_D - M_A$ é denominado conjunto das marcações bloqueantes. O grafo de alcançabilidade constituído apenas pelas marcações admissíveis é denominado R_C.

Para resolver o PCS, é necessário determinar o conjunto de transições controláveis que levam à marcações não-admissíveis. O conjunto de transições que o supervisor deve inibir é dado por $\Omega = \{(M,t) \mid M[t \rightarrow \wedge M \in M_A \wedge t \in T_{cF}\}$, onde $T_{cF} = \{t \in T_c \mid \exists (M,M') \in M_A \times (M_D \cup M_B) \wedge M[t \rightarrow M'\}$. Um supervisor é menos restritivo possível sse toda marcação admissível de M_A é alcançável sob supervisão e todas as transições de estados Ω são proibidas. Tal supervisor é dito ótimo.

3.2.2 Metodologia de síntese

Dados um modelo em RdP e um comportamento desejado para o SED, a metodologia de síntese consiste basicamente de duas etapas. A primeira etapa é baseada no grafo de alcançabilidade da RdP e possui como objetivo a determinação do conjunto M_A. A segunda etapa, por sua vez, utiliza a teoria das regiões e possui como objetivo a determinação dos lugares de controle que serão adicionados a RdP original, de forma a assegurar o comportamento desejado.

Utilização da abordagem Ramadge and Wonham

O algoritmo para a determinação do conjunto M_A segue basicamente a abordagem Ramadge and Wonham [83]. Dada uma RdP N, ele pode ser sintetizado da seguinte forma [35] [2003]:

1. Defina o conjunto das marcações proibidas M_F.
2. Gere o grafo de alcançabilidade parcial $R_P(N,M_0)$, cuja construção não considera os sucessores das marcações proibidas.
3. Identifique o conjunto das marcações perigosas M_D pela análise retrógrada do grafo $R_P(N,M_0)$ a partir das marcações em M_F. Nesta etapa, marcações que levam a marcações proibidas pelo disparo de transições não-controláveis são identificadas. Caso o estado inicial M_0 seja uma marcação perigosa, o PCS não possui solução.

4. Gere o grafo de alcançabilidade R_C, que consiste no grafo derivado de $R_P(N,M_0)$ após remoção das marcações em M_D. Para garantir a coacessibilidade em relação à marcação inicial, R_C deve incluir M_0 e ser fortemente conexo. Caso R_C seja fortemente conexo, vá para o passo 7. Caso contrário, calcule o componente fortemente conexo (CFC) do grafo R_C que inclui M_0.

5. Calcule o conjunto de marcações bloqueantes M_B, que consistem em marcações que levam a marcações fora da CFC através do disparo de transições não-controláveis. Formalmente, o conjunto de marcações bloqueantes é dado por $M_B = \{M \in CFC \mid \exists t \in T_u, M[t \rightarrow M' \wedge \exists M' \notin CFC\}$.

6. Remova as marcações bloqueantes de R_C e volte ao passo 4.

7. R_C é o comportamento desejado, enquanto M_A é seu respectivo conjunto de marcações.

O algoritmo acima fornece o comportamento desejado R_C, o conjunto de marcações admissíveis M_A e o conjunto de transições Ω que levam a estados fora de R_C. Ghaffari et al. [35] provam que o grafo de alcançabilidade obtido corresponde ao comportamento minimamente restritivo do sistema em malha fechada. O supervisor deve ser capaz de restringir o comportamento do SED à R_C, inibindo toda transição controlável que leve a uma marcação bloqueante ou perigosa, ou seja, todas as transições que pertencem ao conjunto T_{cF}.

Utilização da Teoria das Regiões

Dado um grafo de alcançabilidade que representa um comportamento minimamente restritivo, Ghaffari et al. [35] [2003] utilizam a teoria das regiões para sintetizar os lugares de controle que são adicionados à RdP original. A teoria das regiões foi proposta inicialmente por Badouel et al. [6] para a síntese de RdP puras a partir de sistemas de transições finitos. Como o trabalho original foi baseado sobre um ponto de vista teórico da ciência da computação, inicialmente foi necessário dar uma nova interpretação à teoria das regiões, tornando-a mais apropriada às terminologias adotadas na síntese de supervisores.

Dado um grafo de alcançabilidade G_A, a teoria das regiões permite determinar uma RdP N, tal que G_A seja seu grafo de marcações. Seja p um lugar qualquer da RdP que se deseja obter. Considerando-se que a RdP é pura, p pode ser completamente descrito pelo seu vetor de incidência correspondente $C(p,.)$. Como já definido anteriormente, para qualquer transição t habilitada em uma marcação M, tem-se

$$M'(p) = M(p) + C(p,t), \quad \forall (M,M') \in G_A \text{ e } M[t \rightarrow M'. \tag{3.3}$$

Considere qualquer ciclo não orientado γ do grafo de alcançabilidade G_A. Ao realizar a soma da equação 3.3 aplicada a cada um dos estados em γ, chega-se em

$$\sum_{t \in T} C(p,t).\vec{\gamma}[t] = 0, \quad \forall \gamma \in S, \tag{3.4}$$

onde $\vec{\gamma}[t]$ representa a soma algébrica de todas ocorrências de t em γ e S é o conjunto de todos ciclos não orientados do grafo. A equação 3.4 é denominada equação de ciclo.

Considere cada marcação M do grafo de alcançabilidade G_A. Pelo análise do grafo, é possível encontrar um caminho não orientado Γ_M da marcação inicial M_0 a M. Aplicando a equação 3.3 ao longo do caminho chega-se a $M(p) = M_0(p) + C(p,.)\vec{\Gamma}_M$, onde $\vec{\Gamma}_M$ denota a soma algébrica das ocorrências de t em Γ_M. Embora possam existir diversos caminhos de M_0 a M, somente um desses caminhos precisa ser considerado. Isto se justifica, pois, sob as equações de ciclo, o produto $C(p,.)\vec{\Gamma}_M$ é o mesmo para todos os caminhos. A alcançabilidade de qualquer marcação M em G_A implica em

$$M_0(p) + C(p,.)\vec{\Gamma}_M \geq 0, \quad \forall M \in G_A. \tag{3.5}$$

A equação 3.5 é denominada condição de alcançabilidade.

Como as equações de ciclo e as condições de alcançabilidade não são suficientes para a obtenção de uma RdP a partir de um grafo de alcançabilidade, outras condições se fazem necessárias. A primeira condição diz respeito à não habilitação de uma transição t em uma marcação M, o que pode ser satisfeita por algum lugar p. Como a RdP é pura, t é proibida de disparar em M por um lugar p sse

$$M_0(p) + C(p,.)\vec{\Gamma}_M + C(p,t) \leq -1. \tag{3.6}$$

A equação 3.6 é denominada condição de separação de evento de (M,t). O conjunto de todos os pares (M,t) onde M é uma marcação alcançável e t não está habilitada em M é denominado conjunto de instâncias de separação de evento.

A última condição para obtenção de uma RdP com grafo de alcançabilidade idêntico ao grafo fornecido consiste em assegurar que as marcações são diferentes umas das outras. Formalmente, esta condição é dada por

$$\forall M, M' \in G, \exists p \texttt{ tal que } C(p,.)\vec{\Gamma}_M \neq C(p,.)\vec{\Gamma}_{M'}. \tag{3.7}$$

A equação 3.7 é denominada condição de separação de estados.

Inicialmente como foi concebida, a teoria das regiões diz que existe uma RdP N com grafo de alcançabilidade G_A sse existe um conjunto P de lugares $(M_0(p), C(p,.))$ tal que:

1. cada lugar p satisfaz as equações de ciclo (equação 3.4) e as condições de alcançabilidade (equação 3.5),
2. o conjunto de lugares P satisfaz as condições de separação de estado (equação 3.7) e

3. para cada transição t não habilitada em uma marcação M, existe um lugar p que satisfaça a condição de separação de evento (equação 3.6) de (M,t).

Na resolução do PCS proposta por Ghaffari et al. [35] [2003], ao invés de reconstruir uma RdP inteira para o SED a ser controlado, é suficiente calcular um conjunto de lugares $\{p_c\}$ que serão adicionados ao modelo original do SED, constituindo o supervisor. Na metodologia proposta, tanto as equações de ciclo (equação 3.4), quanto as condições de alcançabilidade (equação 3.5) são utilizadas exatamente como foram propostas originalmente pela Teoria das Regiões.

Com relação às instâncias de separação de evento (equação 3.6) a serem proibidas, somente as transições que pertencem ao conjunto Ω precisam ser consideradas. Em outras palavras, ao invés de considerar todas aos transições não disparáveis em uma marcação, como foi proposto originalmente pela Teoria das Regiões, somente transições controláveis que levam a estados fora de R_C são utilizadas durante o processo de síntese. Conseqüentemente, espera-se que a magnitude deste conjunto seja bastante inferior à quantidade total de transições não habilitadas em uma marcação. Portanto, cada elemento em Ω deve ser solucionado pela adição de um lugar p_c, o que representa que a quantidade de lugares a serem adicionados é, no máximo, igual ao número de transições de estado a serem inibidas. Na prática, este número é bastante inferior, pois um mesmo lugar de controle pode ser solução de diferentes instâncias de separação de evento. É importante ressaltar que este problema consiste na resolução de diversos modelos de programação inteira cujo objetivo se resume em encontrar apenas uma solução viável. Por último, as condições de separação de estados (equação 3.7) não são necessárias para a síntese de supervisores, pois elas já são naturalmente satisfeitas pelos lugares presentes no modelo em RdP do SED a ser controlado.

Em suma, a teoria das regiões pode ser utilizada para restringir o comportamento de uma RdP N ao grafo de alcançabilidade R_C sse existe um conjunto P de lugares $(M_0(p_c),C(p_c))$ tal que [35] [2003]:

1. cada lugar p_c satisfaz as equações de ciclo (equação 3.4) e as condições de alcançabilidade (equação 3.5),
2. para cada transição t_{cF} não habilitada em uma marcação M, existe um lugar p_c que satisfaça a condição de separação de evento (equação 3.6) de (M,t_{cF}).

O algoritmo para a síntese de lugares de controle a partir de R_C e Ω consiste em cinco etapas [36]:

1. Gere a árvore de espalhamento de R_C e determine seus ciclos básicos [8]
2. Selecione as equações de ciclo (equação 3.4) independentes utilizando o método de eliminação Gaussiano.

[8] De acordo com resultados da teoria de grafos, as equações de ciclo podem ser reduzidas a equações de ciclos básicos, que podem ser determinados em tempo polinomial. Com esta simplificação, o número de equações de ciclo mutualmente independentes é no máximo $O(T)$ [7]

3. Gere a condição de alcançabilidade (equação 3.5) para cada marcação M em R_C, utilizando o caminho único de M_0 a M na árvore de espalhamento.

4. Enquanto $\Omega \neq \emptyset$ faça

   (a) Gere a condição de separação de evento (equação 3.6) para todos os elementos (M,t) de Ω.

   (b) Resolva as equações 3.4 a 3.6. Caso a solução não exista, o comportamento minimamente restritivo não pode ser satisfeito pela adição de lugares de controle à RdP original.

   (c) Remova de Ω todas as instâncias de separação que podem ser resolvidas pela atual solução.

5. Remova os lugares de controle redundantes para obter a RdP controlada reduzida.

Ghaffari et al. [36] provam que o PCS pode ser resolvido de forma ótima pela adição de um conjunto de lugares de controle à RdP da planta sse existe uma solução $(M_0(p_c),C(p_c))$ que satisfaça as equações 3.4 a 3.6 para cada instância de separação de evento (M,t) em Ω. Além disso, prova-se que a complexidade do algoritmo para a síntese de lugares de controle é polinomial em função dos nodos em R_C.

3.2.3 Exemplo

Considera-se a célula de fabricação ilustrada na figura 3.1 [35]. Este sistema é composto por uma máquina M, um *buffer* intermediário de capacidade unitária B e dois veículos autônomos AGV_1 e AGV_2. A máquina M é considerada pouco confiável e pode apresentar defeitos durante a fabricação de alguma peça. Independentemente da peça produzida ser classificada como boa ou defeituosa, ela é depositada no buffer B. O veículo autônomo AGV_1 é responsável por descarregar uma peça classificada como boa, enquanto o AGV_2 descarrega peças defeituosas. O acesso ao buffer B corresponde a uma seção crítica, pois não pode ser feito simultâneamente pelos dois veículos. Este sistema pode ser modelado pela RdP ilustrada na figura 3.2.

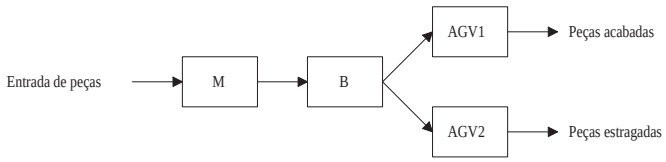

Figura 3.1: Célula de fabricação

A interpretação dos lugares e transições é dada na tabela 3.1. Considera-se $T_u = \{t_2,t_6\}$ e $T_c = T - T_u$.

O SED admite dois estados de bloqueio, identificados pelas marcações $M_1 = (0,0,0,0,1,1,$ $0,0,1,0,0,0,1)$ e $M_2 = (0,1,0,1,0,0,0,0,0,1,0,0,1)$. Na marcação M_1, o veículo AGV_1 possui o acesso ao buffer B, enquanto ocorre uma falha na máquina M. A peça defeituosa espera ser carregada

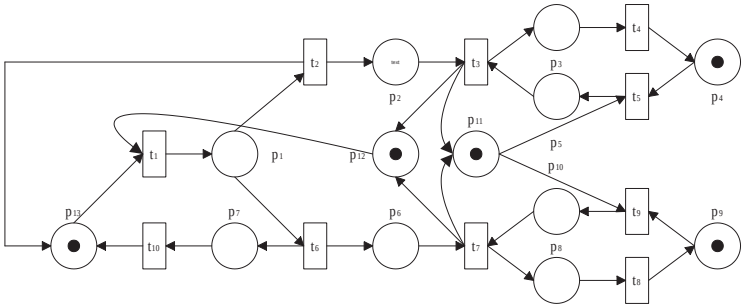

Figura 3.2: RdP da célula de fabricação

Lugar	**Interpretação**
$p1$	M ocupada; peça em fabricação
$p2$	Peça classificada como boa foi produzida e pode ser descarregada
$p3$	Peça sendo transferida pelo AGV_1
$p4$	AGV_1 em repouso
$p5$	Acesso do AGV_1 ao *buffer B*
$p6$	Peça defeituosa foi produzida e pode ser descarregada
$p7$	M com defeito; esperando reparo
$p8$	Peça sendo transferida pelo AGV_2
$p9$	AGV_2 em repouso
$p10$	Acesso do AGV_2 ao *buffer B*
$p11$	Seção crítica de acesso ao *buffer*
$p12$	Capacidade de fabricação de peças simultâneas de M
$p13$	M em repouso
Transição	**Interpretação**
$t1$	Uma peça começa a ser produzida por M
$t2$	Uma peça classificada como boa é produzida
$t3$	AGV_1 carrega uma peça classificada como boa
$t4$	AGV_1 descarrega uma peça classificada como boa
$t5$	AGV_1 ocupa a seção crítica de acesso ao *buffer*
$t6$	Ocorre uma falha em M; uma peça defeituosa é produzida
$t7$	AGV_2 carrega uma peça defeituosa
$t8$	AGV_2 descarrega uma peça defeituosa
$t9$	AGV_2 ocupa a seção crítica de acesso ao *buffer*
$t10$	M é reparada

Tabela 3.1: Interpretação para os lugares e as transições para a RdP da célula de fabricação

pelo AGV_2, que por sua vez espera o AGV_1 desocupar a seção crítica. Isto provoca um bloqueio, pois o AGV_1 não desocupa a seção crítica enquanto nenhuma peça classificada como boa for produzida. Interpretação análoga pode ser feita com relação à marcação M_2.

O modelo em RdP possui um grafo de alcançabilidade com 48 estados. Sob supervisão, seu comportamento fica limitado a 30 estados. A resolução deste problema com a abordagem proposta permite a determinação de um supervisor sob a forma de dois lugares, pc_1 e pc_2, definidos por seus coeficientes na matriz de incidência e por suas marcações iniciais. É importante ressaltar que esta não é a única solução possível. A RdP em malha fechada está ilustrada na figura 3.3.

$$C(pc_1,.) = (0,1,0,0,-1,0,0,0,0,0), M_0(pc_1) = 0$$
$$C(pc_2,.) = (0,0,0,0,0,1,0,0,-1,0), M_0(pc_2) = 0.$$

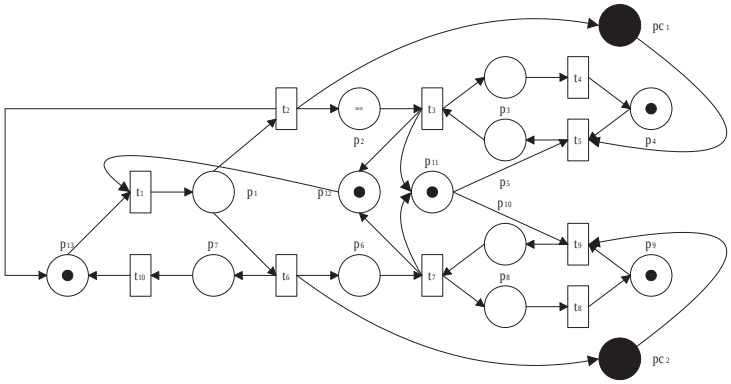

Figura 3.3: RdP da célula de fabricação sob supervisão

3.3 Abordagem Ziller and Schneider

Ziller and Schneider [100] propõe uma nova formulação para o PCS utilizando sistemas de equações inicialmente concebidas a problemas de verificação. A síntese de supervisores é então realizada por meio de equações em μ-calculus que expressam os requisitos de controlabilidade e coacessibilidade. Em trabalhos posteriores, também é descrito como a complexidade da abordagem proposta pode ser reduzida [101], bem como o problema pode ser generalizado, de forma com que diversas outras propriedades possam ser consideradas durante o processo de síntese [99].

3.3.1 Definições

A abordagem de Ziller and Schneider [100] para o PCS utiliza uma estrutura de Kripke construída a partir de um gerador. Dado um gerador $G =< \Sigma, Q, \delta, q_0, Q_M >$ representando o produto síncrono entre uma planta e uma especificação de coordenação, defini-se uma estrutura de Kripke $K_G =< S, I, R, \mathcal{L} >$ sobre as propriedades atômicas $P_G = \{x_q \mid q \in Q\} \cup \{x_b, x_m, x_u\}$ da seguinte forma:

- $S = Q \times \{0,1\}$
- $I = \{(q_0,0),(q_0,1)\}$
- $R((q,0),(q',0))! \Leftrightarrow \exists \sigma \in \Sigma_u \mid \delta(q,\sigma) = q'$
- $R((q,1),(q',1))! \Leftrightarrow \exists \sigma \in \Sigma \mid \delta(q,\sigma) = q'$
- $\mathcal{L}((q,0)) = \{x_q, x_u\} \cup \begin{cases} \{x_b\} & \text{se } q \text{ é mau estado inicial,} \\ \{\} & \text{caso contrário} \end{cases}$
- $\mathcal{L}((q,1)) = \{x_q\} \cup \begin{cases} \{x_m\} & \text{se } q \in Q_M, \\ \{\} & \text{caso contrário.} \end{cases}$

onde S é um conjunto de lugares, I é o conjunto inicial de lugares [9] e $R \subseteq S \times S$ relaciona os estados $(q,0)$ e $(q',0)$ sse existe algum evento não-controlável que leva de q a q' em G e relaciona os estados

[9] Ao contrário da definição 2.6, esta estrutura de Kripke possui dois estados iniciais.

$(q,1)$ e $(q',1)$ sse existe qualquer evento que leva de q a q' em G. Por último, $\mathcal{L}$ atribui propriedades atômicas aos estados. Esta estrutura de Kripke pode ser construída a partir do gerador em tempo $O(|Q| \times |\Sigma|)$.

Para entender esta definição, considera-se o gerador ilustrado na figura 3.4 [100]. Admite-se que os estados 3 e 5 são maus estados iniciais, $\Sigma_c = \{\alpha, \lambda\}$ e $\Sigma_u = \{\beta, \gamma\}$. A estrutura de Kripke correspondente é ilustrada na figura 3.5. De fato, são observadas duas sub-estruturas disconexas, cujos estados que as constituem (duas cópias dos estados originais em G) são identificados pela presença ou ausência da propriedade x_u. A sub-estrutura da esquerda possui somente transições não-controláveis, enquanto a sub-estrutura da direita possui todas as transições de G. Cada estado (q,i) está etiquetado com a propriedade atômica x_q. Adicionalmente, todos os estados que correspondem a estados marcados (maus estados) no gerador possuem a propriedade x_m (x_b). Percebe-se também que a sub-estrutura da esquerda (direita) desconhece os estados marcados (maus estados).

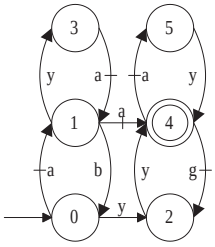

Figura 3.4: Exemplo de gerador

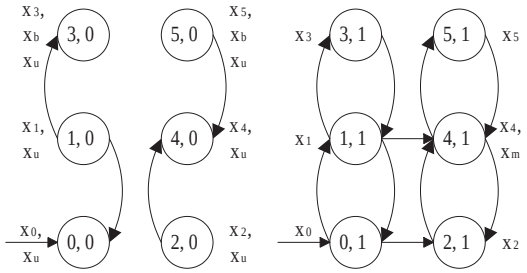

Figura 3.5: Estrutura de Kripke associada

A sintaxe e semântica em μ-calculus são ligeiramente diferentes daquelas encontradas na literatura. A sintaxe inclui a fórmula $\kappa_\pi(\varphi)$, que permite a utilização da função monotônica de transformação de estados $\pi(Q) = \{(q, \neg i) \mid (q,i) \in Q\}$. A semântica, por sua vez, inclui a regra $[[\kappa_\pi(\varphi)]]_K = \pi([[\varphi]]_K)$. Assim definida, a fórmula $\kappa_\pi(\varphi)$ inverte a propriedade x_u que identifica a qual sub-estrutura de Kripke um estado pertence, permitindo assim alternar de uma sub-estrutura para a outra. Desta forma, tem-se:

- $[[x_b]]_{K_G} = \{(q,0)\} \ \ \forall q$ mau estado inicial
- $[[x_m]]_{K_G} = \{(q,1)\} \ \ \forall q \in Q_M$
- $[[x_q]]_{K_G} = \{q\} \times \{0,1\} \ \ \forall q \in Q$
- $[[x_u]]_{K_G} = \{(q,0)\} \ \ \forall q \in Q$

- $[[\neg x_u]]_{K_G} = \{(q,1)\} \ \ \forall q \in Q$

Também é definida uma projeção para mapear estados da estrutura de Kripke para o gerador original. Dados um gerador G, sua estrutura de Kripke associada K_G e uma fórmula Φ em μ-calculus sobre as propriedades atômicas P_G, a projeção de $[[\Phi]]_{K_G}$ em estados de G é dada por $[[\Phi]]_G = \{q \in Q \mid (q,0) \in [[\Phi]]_{K_G} \vee (q,1) \in [[\Phi]]_{K_G}\}$. Com esta projeção, é possível construir um gerador que satisfaça Φ pela restrição do conjunto dos estados do gerador original G a $[[\Phi]]_G$.

As equações em μ-calculus para as propriedades de acessibilidade e coacessibilidade possuem fundamental importância nesta abordagem. O conjunto dos estados de $\tau \subseteq \delta$ que são acessíveis somente através de estados relacionados à τ é dado por $[[x_{Ac}]]_G$, com

$$x_{Ac} \stackrel{\mu}{=} \Phi_\tau \wedge (EX^{-1} x_{Ac} \vee x_{q_0}), \tag{3.8}$$

onde Φ_τ representa os estados de τ. De forma similar, o conjunto dos estados de $\tau \subseteq \delta$ que são coacessíveis somente através de estados relacionados à τ é dado por $[[x_{Co}]]_G$, com

$$x_{Co} \stackrel{\mu}{=} \Phi_\tau \wedge (EX x_{Co} \vee x_m), \tag{3.9}$$

.

Outro importante resultado é a prova de que qualquer conjunto de equações de tamanho constante e profundidade de alternação l formulada para uma estrutura de Kripke associada a um gerador pode ser resolvida em tempo $O(|Q|^l \times |\Sigma|)$ [100].

3.3.2 PCS utilizando μ-calculus

Existem basicamente duas abordagens para resolver o PCS. A abordagem inicial, apresentada por Ramadge and Wonham [82], leva em consideração os geradores da planta e do produto síncrono entre a planta e a especificação de coordenação para encontrar os maus estados iniciais. Após tais estados serem removidos, é obtida a componente *trim* do gerador resultante. Como a remoção de maus estados pode destruir coacessibilidade e remoção de estados não coacessíveis pode expor novos maus estados, o algoritmo é reinicializado com a componente trim substituindo o gerador do produto síncrono inicial e este processo se repete até um ponto fixo ser alcançado. A outra abordagem, apresentada por Kumar and Garg [57], não elimina maus estados ou estados não coacessíveis imediatamente, postergando a eliminação até um ponto fixo ser alcançado. Um estado é considerado mau estado se possui uma transição não-controlável em direção a algum estado já classificado como mau estado ou não coacessível, enquanto a operação de *trimming* é substituída pela identificação dos estados não coacessíveis e pela obtenção da componente acessível do autômato após um ponto fixo ser alcançado. Ao contrário da abordagem Ramadge and Wonham [82], a determinação dos maus estados iniciais é calculada somente no início do processo. Devido às características apresentadas, a segunda abordagem é mais apropriada para a metodologia apresentada por Ziller and Schneider [100].

O PCS utilizando μ-calculus utiliza a estrutura de Kripke K_G associada a um gerador G que representa o produto síncrono entre uma planta e uma especificação de coordenação. Os maus estados, representados inicialmente pelo conjunto x_b [10], são coletados no conjunto x_B, enquanto os estados não coacessíveis são coletados no conjunto x_N. Este processo utiliza as duas sub-estruturas de Kripke, uma durante a verificação de controlabilidade e a outra durante verificação de coacessibilidade. A estrutura cujos estados são identificados por x_u é ideal para a coleta dos maus estados, pois são analisadas somente as transições não-controláveis dos estados predecessores de maus estados ou estados não coacessíveis. Por outro lado, como a verificação de coacessibilidade considera todas as transições, é utilizada a sub-estrutura cujos estados são identificadas por $\neg x_u$. Quando os estados não coacessíveis são considerados no cálculo de x_B, é utilizada a função $\kappa_\pi(x_N)$. Exatamente como foi definido, o conjunto x_B é obtido por meio da equação 3.12.

Para derivar uma expressão para x_N, inicialmente considera-se a negação da equação 3.9, assumindo-se $\Phi_\tau = \neg\kappa_\pi(x_B)$. Estas operações levam à expressão

$$\neg x_{C_o} \stackrel{\nu}{=} \kappa_\pi(x_B) \vee (AX\neg x_{C_o} \wedge \neg x_m), \tag{3.10}$$

onde $\neg x_{C_o} \equiv x_N$. Como a equação 3.10 é uma expressão de ponto fixo máximo, é necessário restringi-la a $\neg x_u$, de forma a eliminar os estados indesejados. O conjunto x_N é então obtido por meio da equação 3.11. Desta forma, chega-se ao sistema em μ-calculus que descreve a solução para o PCS.

$$\begin{cases} x_N \stackrel{\nu}{=} \kappa_\pi(x_B) \vee (AXx_N \wedge \neg x_m \wedge \neg x_u) & (3.11) \\ x_B \stackrel{\mu}{=} EX(x_B \vee \kappa_\pi(x_N)) \vee x_b & (3.12) \end{cases}$$

O componente acessível após a eliminação de todos estados pertencentes a x_B e x_N pode ser obtido através da equação 3.8, assumindo-se $\Phi_\tau = \neg(\kappa_\pi(x_B) \vee x_N)$ e também restringindo o resultado a $\neg x_u$

$$x_{A_c} \stackrel{\mu}{=} \neg(\kappa_\pi(x_B) \vee x_N) \wedge (EX^{-1}x_{A_c} \vee x_{q_0}) \wedge \neg x_u. \tag{3.13}$$

A solução do PCS é dada então pela restrição do gerador G aos estados pertencentes ao conjunto x_{A_c}. Como o sistema de equações em μ-calculus constituído pelas equações 3.11 e 3.12 possui profundidade de alternação $l = 2$, a complexidade do algoritmo é dada por $O(|Q|^2 \times |\Sigma|)$ [100], o que equivale à complexidade obtida pela abordagem Ramadge and Wonham [83]. Também é importante ressaltar que, devido a alternação entre os operadores ν e μ, as equações 3.11 e 3.12 requerem as inicializações $x_N = 1$ (ponto fixo máximo) e $x_B = 0$ (ponto fixo mínimo) em cada computação alternada.

[10] A partir deste momento, o 'conjunto $[[x_b]]_G$' será denotado por o 'conjunto x_b', o que não altera a clareza do seu significado dentro do contexto. O mesmo é válido para os demais conjuntos de estados

3.3.3 Redução da complexidade do PCS

Visando melhorar a complexidade do algoritmo para resolução do PCS, [101] sugerem uma maneira alternativa para coletar estados não coacessíveis. O objetivo é fazer com que a equação em μ-calculus correspondente leve a um ponto fixo mínimo ao invés de um máximo. Desta forma, este sistema possuirá profundidade de alternação $l = 1$, resultando em um algoritmo de complexidade $O(|Q| \times |\Sigma|)$.

A alternativa constituída pelo sistema

$$\begin{cases} x_N \stackrel{\mu}{=} (\neg x_m \wedge AX(\kappa_\pi(x_B) \vee x_N)) \vee x_n & (3.14) \\ x_B \stackrel{\mu}{=} EX(x_B \vee \kappa_\pi(x_N)) \vee x_b, & (3.15) \end{cases}$$

onde x_n é dado por $x_n \stackrel{\nu}{=} \neg x_u \wedge AXx_n \wedge \neg x_m$, possui profundidade de alternação $l = 1$. No entanto, esta solução é válida somente para geradores que não possuem componente escondido de *livelock* (CEL) [101]. Um CEL corresponde a um sub-conjunto de estados de um gerador que satisfazem simultâneamente as seguintes condições:

- são inicialmente coacessíveis, mas se tornam não coacessíveis após alguns estados coletados em x_B serem removidos,
- nenhum se torna mau estado,
- formam um componente conexo.

Um exemplo de CEL está presente no gerador ilustrado na figura 3.6, onde as etiquetas dos eventos são irrelevantes [101]. Suponha que todas transições são controláveis e que o estado 5 seja um mau estado inicial. Embora os estados 2, 3 e 5 sejam não coacessíveis após a remoção de 5, eles não serão detectados pelas equações 3.14 e 3.15. Portanto, estes estados formam um CEL.

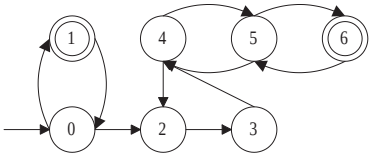

Figura 3.6: Exemplo de componente escondido de *livelock*

Já a alternativa constituída pelo sistema

$$\begin{cases} x_H \stackrel{\nu}{=} (\kappa_\pi(x_B) \vee x_N) \vee (\neg x_u \wedge AXx_H \wedge \neg x_m) & (3.16) \\ x_N \stackrel{\mu}{=} AX(\kappa_\pi(x_B) \vee x_N) \vee x_H & (3.17) \\ x_B \stackrel{\mu}{=} EX(x_B \vee \kappa_\pi(x_N)) \vee x_b & (3.18) \end{cases}$$

substitui as equações 3.12 e 3.11 sem perdas de generalidade. Este sistema possui profundidade de alternação $l = 2$, o que representa uma complexidade $O(|Q|^2 \times |\Sigma|)$ no pior caso. No entanto, esta complexidade é quadrática somente quando gerador possui CELs, sendo linear nos outros casos. Isto porque x_N e x_B são calculados sem a necessidade de reinicialização de seus valores para 0, pois ambos são pontos fixos mínimos. Chega-se também à conclusão que, mesmo na presença de CELs, a nova solução será mais eficiente que a anterior, porque somente os estados não coacessíveis que pertencem a um CEL serão coletados em tempo quadrático. O caso especial de linguagens prefixo-fechadas também é resolvido em tempo linear, pois, como todos os estados são marcados, não existem CELs [101].

3.3.4 Generalização do PCS

Observando que controlabilidade representa uma propriedade de segurança, enquanto coacessibilidade representa uma propriedade de vivacidade, Ziller and Schneider [99] propõe uma generalização ao PCS proposto por Ramadge and Wonham [82] [1989], de forma com que outras propriedades possam ser tratadas. Além disso, a complexidade do algoritmo resultante poderá ser facilmente calculada através da determinação da profundidade de alternação do sistema de equações em μ-calculus envolvido.

Nesta generalização, o raciocínio lógico para a determinação das equações do PCS é invertido, pois as mesmas são descritas em termos dos estados a serem preservados, ao invés daqueles a serem eliminados. Os bons estados são coletados no conjunto x_G, enquanto os estados coacessíveis são coletados no conjunto x_C. As equações 3.11, 3.12 e 3.13 são substituídas respectivamente por

$$\begin{cases} x_C \overset{\mu}{=} \kappa_\pi(x_G) \wedge (EXx_C \vee x_m) & (3.19) \\ x_G \overset{\nu}{=} x_u \wedge AX(x_G \wedge \kappa_\pi(x_C)) \wedge \neg x_b & (3.20) \end{cases}$$

e

$$x_{A_c} \overset{\mu}{=} x_C \wedge (EX^{-1}x_{A_c} \vee x_{q_0}) \wedge \neg x_u, \quad (3.21)$$

sem perdas de generalidade ou aumento de complexidade [99].

Como controlabilidade (segurança) é uma propriedade requerida em qualquer processo de síntese, a equação 3.20 é mantida na generalização. Já a equação 3.19 pode ser substituída, pois não é difícil imaginar problemas em que a requisição de coacessibilidade (vivacidade) se torna inadequada para especificar um comportamento desejado [81]. Por exemplo, suponha que o SED a ser controlado consiste em uma célula de manufatura projetada para produzir um número de peças distintas e que o supervisor resultante seja capaz de permitir que o sistema sempre produza qualquer uma destas peças. Esta condição é uma propriedade de justiça que não pode ser satisfeita diretamente pela abordagem Ramadge e Wonham: se a produção de cada peça é modelada como uma tarefa terminada,

um supervisor não nulo garantirá que o SED alcance pelo menos um dos estados marcados, mas não haverá nenhuma garantia de que todos estados marcados serão sempre alcançáveis. Visando aumentar a abrangência da metodologia proposta, o PCS é generalizado pela substituição do requerimento de um supervisor não bloqueante por qualquer condição expressa em lógica temporal. Ao substituir a equação 3.19 por outra expressão, os estados que satisfazem o novo requerimento são calculados utilizando a mesma sub-estrutura de Kripke que antes era utilizada para o cálculo da coacessibilidade [99].

Dadas uma planta G e uma especificação representada simultaneamente por uma linguagem $K \subseteq L_m(G)$ e por uma propriedade em lógica temporal Ω, o Problema Geral de Controle Supervisório (PCSG) consiste em encontrar um supervisor S para G tal que $L_m(S/G) \subseteq K$ e $K_{G\times S}|_{\neg x_u} \models AG\Omega$ [99]. Esta propriedade em lógica temporal pode ser expressa como um sistema de equações em μ-calculus cuja solução é um conjunto de estados desejados.

Diversos exemplos de especificações em lógica temporal são descritos em [87] e [102]. Formalmente, todos os sistemas de equações possuem a forma geral

$$\begin{cases} x_1 \overset{\sigma_1}{=} & \Phi_1 \\ & \vdots \\ x_n \overset{\sigma_n}{=} & \Phi_n \\ z \overset{\sigma_{n+1}}{=} & \Psi, \end{cases}$$

onde z representa o conjunto de estados que satisfazem a propriedade em lógica temporal Ω. Para aplicar esta condição à sub-estrutura de K_G originalmente utilizada para o cálculo dos estados coacessíveis, tosas as equações deste sistema devem ser restringidas a $\neg x_u$. Ao mesmo tempo, os conjuntos de estados $x_1, ..., x_n$ e z devem ser restringidos à $\kappa_\pi(x_G)$, pois a busca por um supervisor requer que todos os caminhos na solução contenham somente bons estados. Conseqüentemente, o sistema em μ-calculus que descreve a solução para o PCSG possui a forma geral

$$\begin{cases} x_1 \overset{\sigma_1}{=} & \kappa_\pi(x_G) \wedge \neg x_u \wedge \Phi_1 \\ & \vdots \\ x_n \overset{\sigma_n}{=} & \kappa_\pi(x_G) \wedge \neg x_u \wedge \Phi_n \qquad (3.22) \\ z \overset{\sigma_{n+1}}{=} & \kappa_\pi(x_G) \wedge \neg x_u \wedge \Psi \\ x_G \overset{\nu}{=} & x_u \wedge AX(x_G \wedge \kappa_\pi(z)) \wedge \neg x_b \qquad (3.23) \end{cases}$$

Os estados acessíveis podem ser obtidos através da equação 3.8, assumindo-se $\Phi_\tau = z$ e também restringindo o resultado a $\neg x_u$

$$x_{A_c} \overset{\mu}{=} z \wedge (EX^{-1} x_{A_c} \vee x_{q_0}) \wedge \neg x_u. \qquad (3.24)$$

A solução do PCSG é dada então pela restrição do gerador da especificação G aos estados pertencentes ao conjunto X_{A_c}.

Como mencionado anteriormente, o PCS pode não ser suficiente para satisfazer uma propriedade de justiça. No entanto, este problema é facilmente tratado pelo PCSG. Seja φ_1, ..., φ_n as propriedades atômicas referentes aos estados que devem ser alcançados com freqüência infinita, uma expressão em lógica temporal que formaliza este problema é $E[\bigwedge_{i=1}^{n} GF\varphi_i]$. De acordo com Ziller and Schneider [99], o sistema de equações para este PCSG seria

$$\begin{cases} x_1 \overset{\mu}{=} \kappa_\pi(x_G) \wedge \neg x_u \wedge ((z \wedge \varphi_1) \vee EXx_1) \\ \quad \vdots \\ x_n \overset{\mu}{=} \kappa_\pi(x_G) \wedge \neg x_u \wedge ((z \wedge \varphi_n) \vee EXx_n) \\ z \overset{\nu}{=} \kappa_\pi(x_G) \wedge \neg x_u \wedge \bigwedge_{i=1}^{n} EXx_i \\ x_G \overset{\nu}{=} x_u \wedge AX(x_G \wedge \kappa_\pi(z)) \wedge \neg x_b. \end{cases}$$

O algoritmo correspondente possui complexidade $O(|Q|^2 \times |\Sigma|)$, dado que seu sistema de equações possui profundidade de alternação $l = 2$.

3.3.5 Exemplo

Considera-se o gerador ilustrado pela figura 3.4. Admite-se novamente que os estados 3 e 5 são maus estados iniciais, $\Sigma_c = \{\alpha, \lambda\}$ e $\Sigma_u = \{\beta, \gamma\}$. A estrutura de Kripke correspondente está ilustrada na figura 3.5. Este problema será resolvido pelas equações inicialmente propostas em [100]. Em cada computação alternada, a solução para as equações 3.11 e 3.12 são inicializadas com $x_B^0 = 0$ e $x_N^0 = 1$. Os resultados de cada iteração são descritos por meio dos estados da estrutura de Kripke. A notação $[[x_N]]_{K_G}^{i,j}$ denotará o conjunto-resultado da j-ésima iteração para a i-ésima computação de x_N, o mesmo valendo para x_B. O símbolo $^\infty$ será utilizado para representar que a atual computação convergiu para algum conjunto de estados.

$[[x_N]]_{K_G}^{1,0} = \{(0,1),(1,1),(2,1),(3,1),(5,1)\}$
$[[x_N]]_{K_G}^{1,1} = \{(0,1),(3,1)\}$
$[[x_N]]_{K_G}^{1,2} = \{\} = [[x_N]]_{K_G}^{1,\infty}$

$[[x_B]]_{K_G}^{1,0} = \{(3,0),(5,0)\}$
$[[x_B]]_{K_G}^{1,1} = \{(1,0),(3,0),(5,0)\} = [[x_B]]_{K_G}^{1,\infty}$

$[[x_N]]_{K_G}^{2,0} = \{(0,1),(1,1),(2,1),(3,1),(5,1)\}$
$[[x_N]]_{K_G}^{2,1} = \{(0,1),(1,1),(3,1),(5,1)\}$
$[[x_N]]_{K_G}^{2,2} = \{(1,1),(3,1),(5,1)\} = [[x_N]]_{K_G}^{2,\infty}$

$[[x_B]]^{2,0}_{K_G} = \{(1,0),(3,0),(5,0)\} = [[x_B]]^{2,\infty}_{K_G}$

Os pontos fixos são $[[x_N]]^{2,2}_{K_G}$ e $[[x_B]]^{2,0}_{K_G}$. A equação 3.13 pode agora ser utilizada com $[[\neg(\kappa_\pi(x_B) \vee x_N) \wedge \neg x_u]]_{K_G} = \{(0,1),(2,1),(4,1)\}$ e $[[x_{q_0}]]_{K_G} = \{(0,0),(0,1)\}$, o que leva às seguintes iterações

$[[x_{A_c}]]^{1,0}_{K_G} = \{(0,1)\}$
$[[x_{A_c}]]^{1,1}_{K_G} = \{(0,1),(2,1)\}$
$[[x_{A_c}]]^{1,2}_{K_G} = \{(0,1),(2,1),(4,1)\} = [[x_{A_c}]]^{1,\infty}_{K_G}$

Finalmente, a solução é dada restringindo o gerador original aos estados dados pela projeção $[[x_{A_c}]]_G = \{0,2,4\}$.

3.4 Discussão

Foram apresentadas, em ordem cronológica, diversas abordagens ao Controle Supervisório de SEDs modelados por RdP e algumas de suas extensões. Com a apresentação dos resultados de diversos autores, buscou-se situar o estado da arte nesta área do conhecimento. Optou-se por não apresentar trabalhos envolvendo extensões de RdP que levassem em consideração aspectos temporais explicitamente, com o intuito de focar no principal resultado desta pesquisa.

Os resultados provenientes dos trabalhos de Ghaffari et al. [35] e Ziller and Schneider [100] foram minunciamente detalhados, o que facilitará a apresentação da metodologia proposta na Capítulo 4. É importante ressaltar que ambos trabalhos utilizam, em maior ou menor grau, a abordagem Ramadge and Wonham [82]. Conseqüentemente, garante-se que as características desejáveis para a metodologia proposta sejam satisfeitas, pois a mesma também possuirá forte base na abordagem Ramadge and Wonham [82].

Capítulo 4

Metodologia Proposta

A metodologia proposta neste trabalho utiliza alguns princípios de verificação formal para resolver o PCS de SEDs modelados por RdP. Com o objetivo de obtenção de uma abordagem genérica, são utilizados alguns dos resultados apresentados por Ghaffari et al. [35] e Ziller and Schneider [100] em conjunto com outros princípios. Da primeira abordagem, aproveita-se principalmente a utilização da Teoria das Regiões para a determinação de lugares de controle que serão adicionados à RdP da planta. Da segunda abordagem, extrai-se a utilização de equações em μ-calculus para a resolução do PCSG, ou seja, uma generalização da abordagem RW.

Este capítulo inicia-se com uma análise dos princípios gerais acerca desta metodologia. Na primeira seção, estes princípios são apresentados e justificados. A segunda seção apresenta as ferramentas utilizadas e descreve suas características principais. Por último, a terceira seção apresenta o algoritmo proposto, detalhando-o passo a passo. As ferramentas desenvolvidas também são explicitadas, de forma com que os resultados obtidos possam ser facilmente reproduzidos experimentalmente.

4.1 Princípios gerais

O objetivo principal deste trabalho é propor uma metodologia de síntese de supervisores, utilizando SEDs modelados por RdP e alguns princípios de verificação formal. Dentre os objetivos secundários, deseja-se também que esta metodologia apresente as seguintes características:

- Ser computacionalmente eficiente, ou pelo menos possuir a mesma ordem de complexidade da abordagem RW.
- Ser genérica, ou seja, poder ser aplicada a uma ampla classe de problemas.
- Ser flexível, ou seja, poder ser facilmente estendida as outras abordagens existentes na literatura, tais como Controle Modular [95], Controle Modular Local [80] e Controle Multitarefa [81].

- Ser baseada em trabalhos recentes, preferencialmente.

Após uma análise de diversas abordagens existentes na literatura envolvendo SEDs modelados por RdP e/ou processo de síntese utilizando princípios de verificação formal, optou-se por ter como base os trabalhos de Ghaffari et al. [35] e Ziller and Schneider [100]. Ambas as abordagens possuem os princípios fundamentais para o desenvolvimento de uma metodologia com os objetivos supra-citados, além de serem bastante contemporâneos a este trabalho.

A abordagem de Ghaffari et al. [35] utiliza SEDs modelados por RdP e o processo de síntese é dividido em duas etapas. A primeira etapa consiste em utilização direta da abordagem RW sobre o grafo de alcançabilidade da RdP. A outra etapa utiliza a Teoria das Regiões sobre o grafo de alcançabilidade resultante da etapa anterior e possui como objetivo a determinação dos lugares de controle que serão adicionados à RdP original, de forma a se obter o comportamento desejado em malha fechada. Ghaffari et al. [35] não entram em detalhes como os modelos em programação inteira obtidos são resolvidos na prática, ou seja, não citam em qual linguagem os modelos são gerados, nem quais otimizadores são utilizados. Com relação aos aspectos de modelagem, os SEDs são modelados da forma mais tradicional em RdP, ou seja, uma única RdP modela o comportamento global do sistema. Conseqüentemente, necessita-se usualmente de um considerável esforço intelectual, tanto na modelagem dos SEDs, quanto na análise dos modelos em RdP, pois nem sempre é trivial identificar os diversos sub-componentes da planta.

Por sua vez, Ziller and Schneider [100] utilizam SEDs modelados por autômatos (RdP não são mencionadas em nenhum momento deste trabalho). O processo de síntese é realizado através da utilização de equações em μ-calculus e consiste em uma generalização da abordagem RW. Os autores partem do princípio que o autômato utilizado consiste no produto síncrono dos autômatos da planta e das especificações de coordenação e não entram em detalhes sobre como os maus estados iniciais são determinados. A utilização de equações em μ-calculus se mostrou de grande valia, pois foi possível a determinação de um PCSG, ou seja, em uma generalização da abordagem RW. Além disso, provou-se que a ordem de complexidade resultante desta metodologia era similar à obtida pela metodologia anterior.

O algoritmo proposto a seguir procura extrair as principais características destes trabalhos e adiciona algumas características particulares ao processo de síntese. Primeiramente, optou-se por modelar cada sub-componente do SED independentemente, pois, além de ser algo comum na metodologia de síntese de supervisores, esta alternativa é mais intuitiva e simples do que modelar a planta como uma RdP única. Conseqüentemente, serão consideradas as linguagens geradas e marcadas por cada sub-componente e as sub-redes resultantes serão uma variação das RdP etiquetadas. As especificações de coordenação também serão modeladas como RdP, o que resulta em uma maior flexibilidade de modelagem. O comportamento desejado em malha fechada corresponderá, portanto, à fusão de transições destas RdP com as RdP dos diversos sub-componentes da planta. O maus estados iniciais serão determinados por meio de análise prévia das conseqüências imediatas desta fusão de transições. As demais características serão apresentadas com detalhes no decorrer deste capítulo.

4.2 Ferramentas utilizadas

Após um estudo de diversas ferramentas de modelagem e análise de propriedades comportamentais e estruturais de RdP, decidiu-se utilizar o ambiente de *software* Tina (*Time Petri Net Analyser*) [12]. Como ferramenta de verificação, optou-se por utilizar o *model checker* MEC [4]. Dentre os principais motivos de tais escolhas, podem ser citadas a possibilidade de interface entre os dois aplicativos e a facilidade de manipulação de RdP utilizando a primeira ferramenta. Além disso, o MEC foi a ferramenta de verificação formal escolhida, pois ele possibilita a utilização de equações em μ-calculus, através da definição de funções como sistemas de equações em pontos fixos.

4.2.1 Tina

Tina é um ambiente de software para edição e análise de RdP e RdPT [1]. Além das funcionalidades habituais de ferramentas do gênero (tais como a análise das propriedades de alcançabilidade, ausência de *deadlock* e vivacidade), o Tina apresenta várias técnicas para a construção de representações simbólicas para o comportamento da rede, cada uma preservando certas classes de propriedades dos espaços de estados [12]. Embora não seja um *model checker*, o Tina pode ser utilizado em conjunto com ferramentas desta natureza. O domínio de aplicação desta ferramenta é bastante amplo, o que vem sendo comprovado por sua utilização em diversos projetos industriais ou acadêmicos, tais como o Projeto Cotre [8]. Neste trabalho, utilizou-se a versão 2.7.4.

Visão geral

Além das funcionalidades padrão das ferramentas de análise de RdP e RdPT, o Tina oferece diversas construções abstratas que preservam classes de propriedades particulares dos espaços de estados destas redes. Estas classes de propriedades podem ser propriedades gerais (propriedades de alcançabilidade, ausência de *deadlock*, vivacidade), propriedades específicas que se baseiam na estrutura linear do espaço de estados (propriedades da lógica temporal LTL^{-X} [2] e testes de equivalência) ou propriedades específicas que se baseiam na estrutura em ramos (propriedades da lógica temporal CTL* e bissimulação fraca). Como pode-se observar, as propriedades específicas podem ser expressas utilizando tanto lógica temporal, quanto equivalência comportamental. Para as RdP, estas abstrações se baseiam em técnicas de ordem parcial e ajudam a prevenir explosão combinatória. Em SEDs altamente concorrentes, esta operação acarreta em redução drástica do espaço de estados. Para as RdPT, a construção destas representações é necessária, pois estas redes admitem tipicamente espaços de estados infinitos. As abstrações finitas para seus comportamentos são obtidas por diversas técnicas de classes de estados [11] [10] e seus desenvolvimentos recentes [9] [13].

Um primeiro grupo de ferramentas disponibilizadas pelo Tina oferecem as construções clássicas para RdP, tais como o grafo de alcançabilidade. Neste caso, é implementada uma técnica específica

[1] A descrição detalhada da RdPT e seu comportamento associado será vista no Capítulo 6. No entanto, a ausência de tais conceitos não interfere no entendimento desta seção

[2] A lógica temporal LTL^{-X} corresponde à lógica temporal LTL sem o operador X (*next*).

para a verificação eficiente da propriedade de limitabilidade de forma simultânea ao cálculo do grafo de marcações de uma RdP. A construção do grafo de alcançabilidade é interrompida quando é encontrado um lugar não-limitado ou quando uma quantidade limite de marcações (ou marcação de lugares individuais) é alcançada. Esta última característica pode ser utilizada para a verificação em tempo real de propriedades reduzíveis a uma propriedade de alcançabilidade [12].

Por sua vez, as técnicas de redução parcial constituem o framework das técnicas de redução propostas pelo Tina. Estas técnicas possuem o objetivo de prevenir a explosão combinatória e são baseadas na relação de independência entre eventos. Na prática, a relação exata não é disponível, pois seu cálculo requer a prévia construção do espaço de estados. Conseqüentemente, as implementações se baseiam tipicamente em aproximações da relação de independência. No Tina, a relação de independência é aproximada pela independência estrutural, definida como: t_1 e t_2 são independentes sse $Pre(t_1) \cap Pre(t_2) = \emptyset$, ou seja, as transições t_1 e t_2 são independentes caso não compartilhem lugares de entrada [12].

Uma técnica de redução é caracterizada tanto pelo fator de compressão que ela oferece, quanto pelo poder de análise que ela proporciona. Tina oferece três tipos de reduções, já mencionados anteriormente: um que permite a verificação de propriedades gerais e outros dois dedicados à verificação de propriedades específicas que se baseiam na estrutura linear ou em ramos dos espaços de estados. Em geral, os tempos de cálculo para a construção dos grafos relativos à preservação de propriedades da lógica temporal CTL* são muito superiores aos tempos referentes a construção dos grafos relativos a preservação de propriedades da lógica temporal LTL^{-X}. No entanto, o primeiro tipo de redução possui a vantagem de preservar mais propriedades, como, por exemplo, propriedades de vivacidade. Propriedades de segurança, por sua vez, podem ser verificadas em ambos os grafos.

É importante ressaltar que o Tina não é um *model checker*, pois não permite a verificação de propriedades, com a única exceção de propriedades de alcançabilidade. No entanto, pode ser utilizado como uma interface a um *model checker*, fornecendo espaços de estados reduzidos sobre os quais diversas propriedades podem ser verificadas de forma mais eficiente do que se fosse utilizado o espaço de estados original. Com este objetivo, o Tina apresenta seus resultados em uma variedade de formatos de entrada utilizados por alguns *model checkers* e verificadores de equivalência, tais como MEC, Aldébaran e BCG.

Interface com o usuário

O Tina é construído de forma modular. Seus módulos podem ser usados independentemente ou em conjunto, sendo constituídos por:

- um editor gráfico para RdP, RdPT e autômatos;
- uma ferramenta de análise estrutural e
- uma ferramenta de construção de abstrações de espaços de estados.

O editor gráfico gera arquivos que podem ser lidos posteriormente pelas ferramentas de análise estrutural e de construção de espaços de estados. Estas ferramentas podem ser chamadas de forma independente ou pelo próprio editor. No segundo caso, o editor permite a edição e/ou geração da representação gráfica de suas saídas.

Usadas independentemente, as ferramentas funcionam como filtros, o que facilita sua utilização em cadeias de desenvolvimento específicas ou existentes. Em outras palavras, estas ferramentas realizam determinadas operações sobre as estruturas fornecidas por um arquivo de entrada, apresentando os resultados em um arquivo de saída. Chamados em modo prompt, as linhas de comando permitem selecionar a abstração desejada.

Como entrada, são aceitas descrições em formato gráfico ou textual. O primeiro formato é gerado através das facilidades do editor gráfico, enquanto o segundo, embora gerado manualmente, é bem simples e intuitivo.

Os formatos de saída fazem com que o Tina seja utilizado como uma interface para algumas ferramentas de verificação. Os quatro formatos básicos são:

- *verbose*, mais adequado para finalidades pedagógicas;
- autômato;
- BCG, utilizado pelas ferramentas Aldébaran e BCG e
- MEC, utilizado pela ferramenta de mesmo nome [3].

Além disso, o Tina também pode naturalmente ser utilizado como interface para outras ferramentas, desde que haja um filtro que traduza algum dos formatos de saída disponíveis para aquele aceitado pela ferramenta utilizada.

Um exemplo de uma seção no Tina é ilustrada na figura 4.1, onde as três janelas foram organizadas de acordo com a conveniência do usuário. A janela superior à esquerda representa uma RdP sendo editada. A janela à direita indica o resultado textual (formato verbose) da construção grafo de alcançabilidade correspondente, enquanto a janela inferior à esquerda ilustra a representação gráfica deste comportamento.

[3] Tratando-se de RdP, a maior desvantagem ao exportar o comportamento de uma rede para o formato MEC é a perda das informações de peso nos arcos e quantidade de fichas nos lugares. Desta forma, estas propriedades são transmitidas de forma binária, ou seja, de acordo com sua presença ou ausência. Tratando-se de RdPT, as informações temporais também não são incluídas.

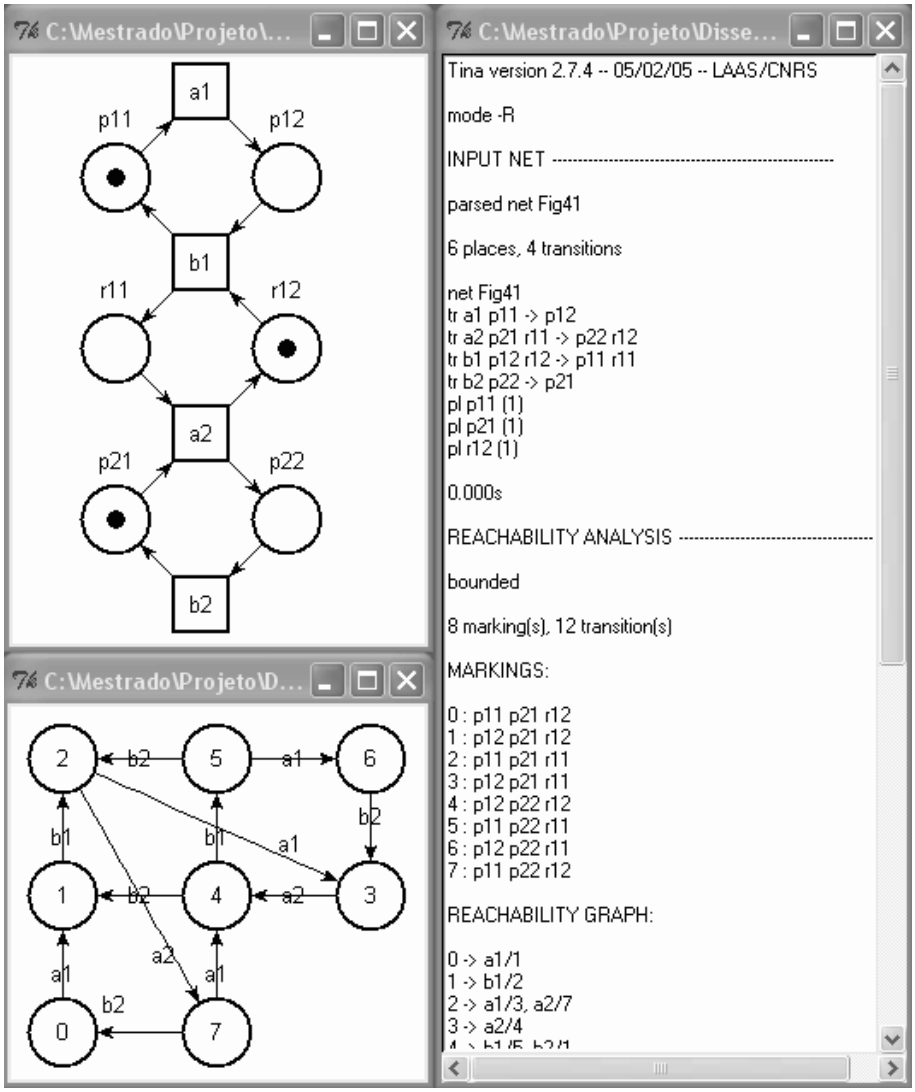

Figura 4.1: Exemplo de uma seção típica no Tina

4.2.2 MEC

MEC é um *model checker* que manipula sistemas de transições [4] e verifica propriedades através do cálculo de conjuntos de estados e conjuntos de transições [5]. Uma vez que o SED é modelado por um sistema de transições, o MEC possibilita a utilização de uma descrição formal para determinar quando este SED possui algumas propriedades. Neste trabalho, utilizou-se a versão 4.0.

[4] Um sistema de transições possui definição semelhante a um autômato de estados finitos. Ao invés de considerar apenas SEDs deterministas, o MEC assume comportamentos em que nunca haverá duas (ou mais) transições diferentes com a mesma etiqueta interligando os mesmos dois estados. Em outras palavras, não é possível distinguir duas transições possuindo a mesma causa e o mesmo efeito sobre o SED.

[5] O MEC também possui sistemas de sincronização como estruturas, mas estes não serão detalhados, pois não são relevantes no contexto deste trabalho.

Visão geral

MEC é uma ferramenta de verificação de propriedades de estados e de transições de sistemas de transições parametrizados. Por parametrização, entende-se que os estados e transições podem possuir informações adicionais, denominadas parâmetros. Tais parâmetros podem ser utilizados para expressar propriedades booleanas de estados e transições, tais como a definição de um estado inicial ou a atribuição de uma característica específica a uma transição (quando esta característica não está especificada pela etiqueta da transição).

As variáveis usadas em MEC possuem dois tipos, de acordo com o tipo de conjunto que pode ser atribuído às mesmas. Mais precisamente, o mecanismo básico implementado consiste na execução de atribuições $<$ *nome do conjunto* $>:=<$ *expressão* $>$, de forma semelhante a uma linguagem de programação. Em outras palavras, o MEC calcula ora conjuntos de estados, ora conjuntos de transições do último sistema de transições carregado em memória, denominado sistema de transições corrente. Para calcular as propriedades, o MEC utiliza algoritmos lineares com relação ao tamanho do grafo, que corresponde à soma das quantidades de estados e transições. Como todos os cálculos são realizados em memória, a única limitação é a quantidade de memória (física ou virtual) disponível no sistema operacional.

Um conjunto é referenciado por seu nome e é definido por uma expressão construída a partir de conjuntos vazios e totais, conjuntos já existentes, operadores booleanos e funções. As constantes vazia, denotada por $\{\}$, e total, denotada por $*$, podem se referir tanto a conjuntos de estados, quanto a conjuntos de transições. Normalmente, a expressão em si permite o MEC determinar seu tipo. Os operadores booleanos consistem nas operações de união, intersecção e diferença, denotados respectivamente por $\vee$, $\wedge$ e $-$. A propriedade associativa não é garantida, o que obriga utilização de parêntesis.

As funções consistem em funções pré-definidas pelo MEC ou funções definidas pelo usuário. As funções pré-definidas de maior interesse neste trabalho incluem a determinação de um conjunto de estados como fontes ou alvos de transições dadas, bem como suas funções reversas, ou seja, a determinação de um conjunto de transições que possuem suas fontes ou alvos em estados dados. Outras funções de grande interesse incluem a determinação dos estados acessíveis ou coacessíveis a partir de estados dados, restritos ou não a um conjunto de transições. Por sua vez, as funções definidas pelo usuário são obtidas através da definição de funções como sistemas de equações em pontos fixos, sendo recursivas por natureza. As funções definidas como sistemas de equações em pontos fixos mínimos possuem um poder de expressão tão eficiente quanto aquele obtido pelas equações em μ-calculus de profundidade unitária. Desta forma, as funções definidas pelo usuário também podem ser calculadas em tempo linear [4]. Dentro deste contexto, o MEC também permite definir funções como sistemas de equações em pontos fixos máximos.

É importante ressaltar que a declaração do tipo de uma variável é simultânea à sua definição inicial, sendo que esta operação possui aparência idêntica à atribuição de valores da linguagem de programação Pascal. Como uma variável é declarada simultâneamente à sua definição, o tipo da expressão que lhe é atribuída não deve ser ambíguo. Como exemplo, a expressão $<$ *nome do conjunto* $>:=$

$*$ não deve ser utilizada, pois o simples símbolo da constante total não proporciona nenhuma informação de tipo. Além disso, uma vez carregada em memória, uma variável não pode ser deletada, nem pode ter seu tipo redefinido. Por outro lado, seu conteúdo pode ser redefinido, desde que a expressão possua um contexto consistente com o tipo da variável. Uma expressão pode ser digitada a qualquer momento em que um conjunto é requerido.

De forma sintética, os objetos de interesse tratados pelo MEC são:

- sistemas de transição e
- funções de ponto-fixo definidas pelo usuário, ou seja, funções definidas como sistemas de equações em pontos fixos mínimos ou máximos,

enquanto as operações normalmente realizadas são:

- definição de objetos,
- cálculo de conjuntos de estados e de conjuntos de transições e
- comandos gerais, tais como carregar e salvar arquivos.

Todas as operações (incluindo as definições de objetos) são executadas pelo MEC quando os comandos textuais correspondentes são digitados em modo *prompt*, de forma iterativa com o usuário. Após analisados por um interpretador, estes comandos são executados em tempo real. Quando os erros são detectados e diagnosticados, nenhuma execução ocorre. Por outro lado, o MEC também é apto a carregar arquivos texto, sendo a forma usual de carregar definições de objetos e de funções. Os resultados da verificação, tais como sistemas de transições com conjuntos recentemente calculados, também podem ser salvos em arquivo texto, de acordo com o formato do respectivo comando de definição. Desta forma, este resultado pode ser reutilizado em uma sessão futura do MEC.

Interface com o usuário

A interface entre o MEC e o usuário é uma janela em modo *prompt* que mostra informações sobre as operações realizadas, tais como a lista dos objetos carregados e a lista das propriedades calculadas sobre o sistema de transições corrente. Basicamente, o menu é dividido em duas áreas horizontais: uma extensa área de informação e uma área de diálogo inferior. Um exemplo de uma seção no MEC é ilustrada na figura 4.2.

Figura 4.2: Exemplo de uma seção típica no MEC - FALTA FIGURA

A área de informação apresenta uma visão geral dos items carregados ou calculados. A primeira linha é denominada linha de status, pois apresenta informações gerais do sistema, tais como o número de versão do MEC e o consumo dos recursos computacionais. Abaixo da linha de status, a janela

longitudinal mais à esquerda mostra a lista dos sistemas de transições carregados, enquanto a janela à direita mostra os sistemas de sincronização carregados e as funções de ponto fixo definidas pelo usuário. O sistema de transições corrente é enfatizado na janela central, onde os nomes dos conjuntos de estados e de transições calculados são listados em conjunto com suas respectivas quantidades de elementos. As cardinalidades dos conjuntos de todos os estados e de todas as transições também são indicadas.

A área de diálogo é reservada para a entrada de comandos e para a visualização das mensagens de erro. Um comando em MEC consiste de uma ou mais linhas, cada uma terminada com o caractere "carriage return". A delimitação de um comando é feita através de "ponto e vírgula". Os comandos são validados linha a linha e a interpretação completa é realizada após a detecção de um "ponto e vírgula"seguido por "carriage return". Caso não ocorra nenhum erro, um comando é executado imediatamente após ser verificado.

As mensagens de erro são mostradas imediatamente após sua identificação, seja na validação de uma linha ou na interpretação de um comando. Em ambos os casos, o comando é descartado. Caso o erro ocorra durante o carregamento dos comandos presentes em um arquivo texto, a leitura do arquivo é interrompida e o número da linha onde o erro foi detectado é mostrada.

Comandos utilizados

Com a exceção dos comandos de definição, que utilizam uma sintaxe específica para definição dos objetos, os comandos em MEC possuem duas formas basicamente:

- atribuições: $<$ *nome do conjunto* $>:=<$ *expressão* $>$;
- procedimentos: $<$ *nome do procedimento* $>$ ($[<$ *lista de argumentos* $>]$) [6];

A seguir serão descritos apenas os comandos necessários ao entendimento deste documento, com a exceção das atribuições, que já foram descritas anteriormente. Uma descrição detalhada de todos os comandos é encontrado em [4].

Definição de sistemas de transições

A definição de um sistema de transições consiste em três partes:

- Um cabeçalho contendo a palavra reservada *transition_system* seguida pelo nome do sistema de transições e outros items opcionais.
- Uma lista de estados, cada um seguido por suas transições de saída.
- Uma lista de conjuntos de estados (opcional) [7].

[6] O símbolo [] indica que respectivo conteúdo pode não existir ou até ser optativo em alguns procedimentos

[7] Quando o Tina exporta um arquivo para o formato MEC, este campo não é preenchido.

Quando um sistema de transições é salvo em arquivo texto, esse é o comando gerado internamente pelo MEC.

Definição de funções

A definição de uma função em pontos fixos consistem em três partes:

- Um cabeçalho contendo a palavra reservada *function* seguida pelo nome da função, um conjunto de parâmetros e um valor de retorno. Os dois últimos componentes consistem no nome e no tipo da variável.
- Uma declaração das variáveis locais (opcional).
- Uma lista de equações que definem a solução da função. Cada equação possui a forma $<$ *nome do conjunto* $>:=<$ *expressão* $>$, onde $<$ *nome do conjunto* $>$ pode ser tanto o valor de retorno, quanto uma variável local e $<$ *expressão* $>$ é qualquer expressão construída utilizando operadores booleanos e/ou funções pré-definidas.

As variáveis são descritas como conjuntos de estados ou conjuntos de transições, representados, respectivamente, pelas palavras reservadas *state* e *trans*. Além disso, a utilização de variáveis sinalizadas permite a definição de funções como sistemas de equações em pontos fixos máximos, bem como pontos fixos mínimos, simplesmente alternando sinais. Isto se deve, pois o menor elemento para inclusão corresponde ao maior elemento para restrição e vice-versa [4]. As palavras reservadas correspondentes são formadas pela inclusão do caractere *underscore*, ou seja, *_state* e *_trans*.

Funções pré-definidas

As funções pré-definidas proporcionam o cálculo de conjuntos de estados ou de transições, bem como a definição de novas funções. Algumas destas funções podem ser restritas, o que significa que elas são aplicadas a sub-sistemas de transições cujos estados e transições são definidos por subconjuntos associados. Quando aplicável, as transições são restritas por um subconjunto de transições cujas origens estão entre os estados restritos.

Sejam σ o nome de um conjunto de estados e τ o nome de um conjunto de transições, as principais funções pré-definidas dentro do contexto deste trabalho são:

- $src(\tau)$: fornece o conjunto dos estados fonte de τ.
- $tgt(\tau)$: fornece o conjunto dos estados alvo de τ.
- $rsrc(\sigma)$: fornece o conjunto das transições cujas fontes são definidas como σ.
- $rtgt(\sigma)$: fornece o conjunto das transições cujos alvos são definidos como σ.
- $reach(\sigma,\tau)$: fornece o conjunto dos estados alcançáveis a partir de σ, podendo ser restrito a τ.

- *coreach*(σ,τ): fornece o conjunto dos estados a partir dos quais σ é alcançável, podendo ser restrito a τ.

Todas estas funções são avaliadas em tempo linear com relação ao tamanho do sistema de transições [4].

Comando *load*

O comando *load* realiza a leitura de um arquivo texto, fornecendo seus dados ao interpretador de comandos. Sua sintaxe é *load(<nome do arquivo>)*, onde *<nome do arquivo>* corresponde a um arquivo texto qualquer. Este arquivo contém normalmente definições de funções e de sistemas de transições, mas também pode incluir demais comandos.

Comando *dts*

O comando *dts* substitui o sistema de transições corrente. Sua sintaxe é *dts(<nome do sistema de transições>)*, onde *<nome do sistema de transições>* corresponde a um sistema de transições previamente carregado.

Comando *ats*

O comando *ats* possui a sintaxe *ats(<nome do arquivo>, <nome do conjunto de estados>, <nome do conjunto de transições> [, <nome do sistema de transições>])*. Funcionalmente, é utilizado para salvar no arquivo texto indicado pelo primeiro argumento a definição, de acordo com o formato MEC, do sistema de transições cujo nome corresponde ao quarto argumento, restrito aos conjuntos de estados e de transições definidos pelo segundo e terceiro argumentos, respectivamente. Estes argumentos são expressões de conjuntos, e a restrição de transição se aplica às transições de saída dos estados restritos. Quando o quarto argumento é omitido, esta função se refere ao sistema de transições corrente. Adicionalmente, os conjuntos totais permitem uma visão completa da definição de um sistema de transições. Conseqüentemente, *ats(<nome do arquivo>, *, * [, <nome do sistema de transições>])* representa a sintaxe do comando para mostrar o sistema de transições corrente em sua totalidade.

Esta é uma forma bastante poderosa para visualização dos sistemas (ou sub-sistemas) de transições através de um editor de texto qualquer. O comando *wts* é comando análogo para mostrar um sistema de transições na tela do MEC.

4.3 Algoritmo proposto

Como já mencionado, a metodologia proposta se baseia fortemente nos trabalhos apresentados por Ghaffari et al. [35] e Ziller and Schneider [100]. O algoritmo correspondente consiste nos seguintes passos:

1. Modele cada componente do SED por uma RdP. O conjunto destas redes é denominado RdP da Planta (RdP-P), pois constitui o comportamento da planta em malha aberta.

2. Modele cada especificação de coordenação por uma RdP. A composição do conjunto de todas as RdP de especificação de coordenação (RdP-EC) com a RdP-P constitui o comportamento desejado em malha fechada, denominada RdP da especificação (RdP-E).

3. Gere o grafo de alcançabilidade G_A^E do comportamento desejado em malha fechada.

4. Utilize a abordagem Ziller and Schneider [100] para resolver o PCS sobre o grafo G_A^E. Encontre o grafo de alcançabilidade G_R^E e o conjunto de transições Ω que levam a marcações fora de G_R^E.

5. Gere um modelo em programação linear inteira para cada transição $t \in \Omega$, baseando-se na segunda parte da abordagem Ghaffari et al. [35].

6. Resolva cada modelo em programação linear inteira.

7. Caso todos os modelos possuam solução, o supervisor minimamente restritivo pode ser obtido. Passe para o passo 8. Caso contrário, não existe solução ótima para este PCS.

8. Elimine os lugares de controle redundantes e adicione os demais lugares de controle à RdP-E, constituindo a RdP supervisionada (RdP-S).

Os dois primeiros passos deste algoritmo, que correspondem à modelagem do PCS, utilizam a ferramenta Tina. O passo 1 consiste na modelagem dos componentes da planta, sendo mais intuitiva do que a modelagem da planta por uma única RdP. Além disso, os modelos são baseados nas linguagens geradas e marcadas por cada componente, seguindo os mesmos conceitos da modelagem por autômatos da abordagem RW. Conseqüentemente, são utilizadas como estruturas as RdP etiquetadas [8], definidas em 3.2. Também é importante ressaltar que esta modelagem facilita a utilização de extensões futuras ao processo de síntese do supervisor, tais como Controle Modular [95], Controle Modular Local [80] e Controle Multitarefa [81].

Durante a modelagem dos componentes da planta, algumas inconveniências poderão surgir. Considera-se um componente genérico, cujo comportamento pode ser modelado pela figura 4.3. O respectivo modelo em RdP ilustrado pela figura 4.4 é errôneo, pois duas transições distintas não podem possuir a mesma etiqueta.

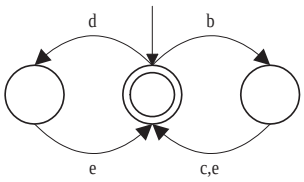

Figura 4.3: Autômato para um componente genérico

Duas soluções foram propostas para a resolução deste problema. A primeira solução consiste em associar diversas transições à ocorrência de um mesmo evento no modelo por autômatos. Desta forma, no exemplo em questão, as etiquetas das transições que se referem ao evento *e* do autômato

[8] No decorrer da apresentação da metodologia proposta e dos respectivos resultados, as RdP etiquetadas serão denominadas apenas de RdP.

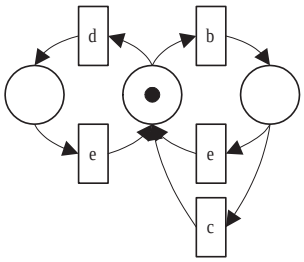

Figura 4.4: RdP errônea para um componente genérico

ilustrado pela figura 4.3 poderão possuir nomes distintos. Um possível modelo em RdP para este comportamento é ilustrado pela figura 4.5, onde as transições etiquetadas por e_1 e e_2 estão associadas à ocorrência do evento e. A maior inconveniência desta solução é que durante a composição entre RdP, uma das atividades presentes no próximo passo, cada arco de entrada ou de saída a uma transição associada ao evento e deverá ser conectado simultâneamente às transições e_1 e e_2. Esta solução, portanto, poderá acarretar em explosão de arcos, caso a ocorrência de diversos eventos esteja associada a diversas transições na RdP.

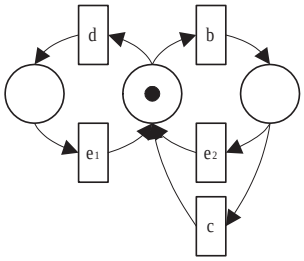

Figura 4.5: Possível RdP para um componente genérico

A segunda solução faz uso direto da Teoria das Regiões, exatamente como foi proposta inicialmente por Badouel et al. [6]. Com esta alternativa, às vezes é possível gerar RdP sem a necessidade de associar mais de transição à ocorrência de um mesmo evento no modelo por autômatos. Visando comparar as duas soluções propostas, o comportamento do autômato ilustrado pela figura 4.6 foi modelado pelas RdP ilustradas nas figuras 4.7 e 4.8. Embora pareça uma solução mais razoável (e automática), o uso da Teoria das Regiões pode se defrontar com problemas infactíveis por natureza. Em outras palavras, não há garantias que um resultado válido seja sempre obtido, pois a Teoria das Regiões nem sempre encontra uma RdP pura com base em um grafo de alcançabilidade.

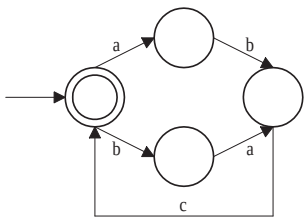

Figura 4.6: Autômato para outro componente genérico

O passo 2 consiste na modelagem das especificações de coordenação e possui os mesmos princípios do passo anterior. A única diferença é a inclusão da determinação do comportamento desejado

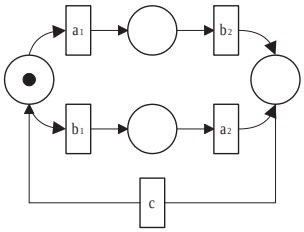

Figura 4.7: Possível RdP para outro componente genérico - 1^a solução

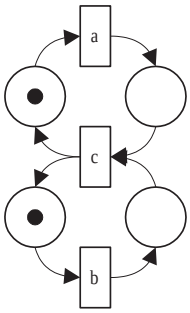

Figura 4.8: Possível RdP para outro componente genérico - 2^a solução

em malha fechada RdP-E, através da fusão de transições entre o comportamento da planta em malha aberta RdP-P e o conjunto das especificações de coordenação RdP-EC. Como já mencionado anteriormente, a existência de diversas transições associadas à ocorrência de um mesmo evento do SED é indesejada, pois pode acarretar em explosão de arcos durante a fusão de transições.

A geração do grafo de alcançabilidade G_A^E do comportamento desejado em malha fechada, atividade única do passo 3, também é realizada utilizando-se a ferramenta Tina. O grafo de alcançabilidade G_A^E é salvo em um arquivo de saída no formato MEC, sendo exportado como um sistema de transições. Um exemplo de um arquivo gerado automaticamente é ilustrado na figura 4.9. As propriedades dos estados representam a presença de fichas nos lugares da RdP-E, enquanto as propriedades das transições representam tão somente as etiquetas das mesmas.

O passo 4 corresponde à etapa mais complexa do algoritmo, pois diversas adaptações foram necessárias para que a abordagem Ziller and Schneider [100] fosse utilizada. A razão comum destas adaptações foi o fato do sistema de transições exportado pelo Tina não ser completamente semelhante à estrutura de Kripke gerada pela abordagem anterior. Primeiramente, os estados presentes no sistema de transição gerado não possuem propriedades atômicas além da indicação da presença de fichas em determinados lugares da RdP-E. Em segundo lugar, não existe uma sub-estrutura possuindo somente transições não-controláveis. Ao invés de desenvolver um programa para gerar tal estrutura de Kripke, optou-se por definir funções em sistemas de equações em μ-calculus no *model checker* MEC que utilizassem somente os sistemas de transições exportados pelo Tina e que fossem equivalentes aos sistemas de equações propostos pela abordagem Ziller and Schneider [100].

Primeiramente, analisou-se qual solução seria tomada como base para a realização destas adaptações: a proposta em [100] ou a proposta em [99]. A primeira solução possui a vantagem de possuir uma extensão que permite uma sutil redução na ordem de complexidade do algoritmo. A segunda

```
transition_system espec2 < width = 0 >;
0 < property = (p11,p21,r12) >  |-
  0 -> 1 < property = a1 > ;
1 < property = (p12,p21,r12) >  |-
  1 -> 2 < property = b1 > ;
2 < property = (p11,p21,r11) >  |-
  2 -> 3 < property = a1 > ,
  3 -> 7 < property = a2 > ;
3 < property = (p12,p21,r11) >  |-
  4 -> 4 < property = a2 > ;
4 < property = (p12,p22,r12) >  |-
  5 -> 5 < property = b1 > ,
  6 -> 1 < property = b2 > ;
5 < property = (p11,p22,r11) >  |-
  7 -> 6 < property = a1 > ,
  8 -> 2 < property = b2 > ;
6 < property = (p12,p22,r11) >  |-
  9 -> 3 < property = b2 > ;
7 < property = (p11,p22,r12) >  |-
  10 -> 4 < property = a1 > ,
  11 -> 0 < property = b2 > ;
< initial = { 0 } >.
```

Figura 4.9: Arquivo em formato MEC para o grafo de alcançabilidade G_A^E

solução, por sua vez, pode ser generalizada, de forma com que alguns problemas não cobertos pela abordagem RW possam ser tratados. Conseqüentemente, o PCSG pode, por exemplo, possuir aplicações em comum com aquelas tratadas pelo Controle Multitarefa [81]. Além da vantagem da generalização, outro aspecto que levou à escolha da solução proposta em [99] foi uma importante limitação no *model checker* MEC: não é possível a definição de uma função que possua dois valores de retorno, o que seria necessário caso fosse utilizado o sistema de equações desenvolvido em [100]. Portanto, optou-se por utilizar a solução proposta em [99], descrita novamente a seguir.

$$\begin{cases} x_C \overset{\mu}{=} \kappa_\pi(x_G) \wedge (EX x_C \vee x_m) & (4.1) \\ x_G \overset{\nu}{=} x_u \wedge AX(x_G \wedge \kappa_\pi(x_C)) \wedge \neg x_b & (4.2) \end{cases}$$

$$x_{A_c} \overset{\mu}{=} x_C \wedge (EX^{-1} x_{A_c} \vee x_{q_0}) \wedge \neg x_u, \quad (4.3)$$

Após analisar as características dos sistemas de transições exportados pelo Tina e as equações 4.1, 4.2 e 4.3, concluiu-se que as adaptações necessárias se resumem a três classes:

- Implementação das funções existenciais EX e EX^{-1} e da função universal AX.
- Definição das condições iniciais x_m e x_b.
- Adaptação das equações, de forma a não utilizar uma sub-estrutura que considera somente transições não-controláveis.

As funções existenciais e universais foram implementadas no MEC utilizando-se as funções pré-definidas *src*, *tgt*, *rsrc* e *rtgt* [9]. As funções EX, EX^{-1} e AX definidas no MEC são ilustradas na figura 4.10, nesta ordem. As funções existências são auto-explicativas, enquanto a função universal AX é um pouco mais complexa. A expressão $rtgt(*-X)$ fornece o conjunto das transições cujos estados alvo não possuem a propriedade booleana X. A expressão completa, por sua vez, fornece o conjunto dos estados que não são fonte das transições mencionadas anteriormente. Esta seqüência de operações equivale à expressão em lógica temporal $AX(X)$.

```
function EX(X: state) return Y:state;
begin
  Y = src(rtgt(X))
end.

function EY(X: state) return Y:state;
begin
  Y = tgt(rsrc(X))
end.

function AX(X: state) return Y:state;
begin
  Y = * - src(rtgt(* - X))
end.
```

Figura 4.10: Definição das funções existenciais e universais no MEC

As definições das condições iniciais x_m e x_b são mais dependentes da modelagem do problema do que de funções definidas no MEC. Os estados marcados da RdP-E são determinados diretamente pela marcação dos seus respectivos lugares. Mais precisamente, um estado é marcado caso todos seus lugares que contenham fichas sejam marcados [10]. Por sua vez, um estado é considerado mau estado inicial quando a presença de um lugar em RdP-EC faz com que o disparo de uma transição não-controlável, antes permitido pelos lugares RdP-P, não seja mais possível. Desta forma, a determinação dos maus estados iniciais da RdP-EC é feita a partir da análise da atuação da RdP-EC sobre a RdP-P. Esta análise, portanto, necessita de forte presença do projetista. As funções responsáveis pelas condições iniciais são ilustradas na figura 4.11. O primeiro parâmetro da função *BAD_STATE* corresponde aos lugares em RdP-P que, através de suas marcações, habilitam uma transição não controlável, dada pelo segundo parâmetro. A função *MARKED_STATE*, por sua vez, nem possui a necessidade de existir.

A classe mais complexa das adaptações (e que, por sinal, engloba as duas adaptações anteriores), consiste na definição de equações no MEC para resolver o PCS sem utilizar uma sub-estrutura que considera somente transições não-controláveis, ou seja, sem a necessidade de gerar a estrutura de Kripke proposta pela abordagem Ziller and Schneider [99]. O sistema de equações constituído pelas equações 4.1 e 4.2 deu origem à definição da função ilustrada na figura 4.12. A interpretação

[9] Em alguns casos, as funções pré-definidas *reach* e *coreach* também poderiam ser utilizadas.

[10] Nesta definição, é importante não confundir a marcação de uma rede (indicação da quantidade de fichas nos lugares) com a marcação dos lugares ou estados da rede (indicação das tarefas completas). Por este motivo, os termos marcação e estado serão utilizados preferencialmente no primeiro e no segundo contexto, respectivamente. Espera-se que as poucas exceções não interfiram na compreensão deste documento

```
function BAD_STATE(X: state; T: trans) return Y:state;
begin
  Y = X - src(T)
end.

function MARKED_STATE(X: state) return Y:state;
begin
   Y = X
end.
```

Figura 4.11: Definição das condições iniciais no MEC

da primeira equação presente nesta função é trivial, pois a variável x_m corresponde ao o valor de saída da função *MARKED_STATE*, enquanto o termo $src(tgt(x_c))$ corresponde à implementação da função $EX(x_c)$ [11]. Por último, a regra $\kappa_\pi(x_g)$ presente na equação 4.1 é mapeada diretamente na variável x_g, pois a sub-estrutura de Krikpe responsável pela determinação dos estados co-acessíveis na metodologia Ziller and Schneider [99] equivale ao sistema de transições exportado pelo Tina, pois ambos consideram todas as transições do SED.

```
function PCS(Xb: state; Xm: state; U: trans) return Xc:_state;
var Xg:_state
begin
  Xc = Xg /\ (src(rtgt(Xc)) \/ Xm);
  Xg = (* - src(rtgt(* - (Xg /\ Xc)) /\ U)) /\ (* - Xb)
end.
```

Figura 4.12: Definição de função no MEC para a solução do PCS

A interpretação da segunda equação é mais complexa, pois seus termos não são mapeados de forma disjunta nos termos da equação 4.2, com a exceção dos maus estados iniciais. Desconsiderando a alternância entre sub-estruturas imposta pela regra $\kappa_\pi(x_c)$, o termo $AX(x_G \wedge \kappa_\pi(x_C))$ é mapeado no termo $(* - src(rtgt(* - (x_g \wedge x_c))))$. Para abstrair a utilização de uma sub-estrutura que considera apenas transições não-controláveis, o que é imposto na equação 4.2 pela regra $\kappa_\pi(x_c)$ e pelo termo x_u, foi necessário redefinir parte da função AX. Desta forma, a variável U presente na expressão $(src(rtgt(* - (x_g \wedge x_c)) \wedge U))$ permite que a equação resultante considere somente as transições não-controláveis. Por último, a negação dos maus estados iniciais é mapeada no termo $(* - x_b)$, onde x_b corresponde ao valor de saída da função *BAD_STATE*.

Para determinar o componente acessível dos estados calculados anteriormente, a equação 4.3 deu origem à função ilustrada na figura 4.13. Em sua única equação constituinte, o termo $(tgt(rsrc(X_{A_c})))$ corresponde à implementação da função $EX^-(x_{A_c})$. O termo x_c é dado pelo valor de saída da função *PCS*, enquanto o termo $\neg x_u$ não precisa ser considerado, pois o sistema de transições manipulado pelo MEC equivale em sua totalidade à sub estrutura de Kripke responsável pela determinação dos estados acessíveis na abordagem Ziller and Schneider [99].

A parte conceitual do passo 4 se encerra com a definição de uma equação para a determinação do conjunto de transições Ω que levam a marcações fora de G_R^E, denominadas instâncias de separação de

[11] Como tudo indica, o MEC não permite que uma função chame outra função definida pelo usuário

```
function AC_PCS(Xc: state; Xqo: state) return Xac:state;
begin
  Xac = Xc /\ ((tgt(rsrc(Xac))) \/ Xqo)
end.
```

Figura 4.13: Definição de função no MEC para a determinação dos estados acessíveis da solução do PCS

eventos. A equação ilustrada na figura 4.14 consiste na conjunção de três termos: $rsrc(R_c)$ calcula as transições de saída dos estados de G_R^E, $rtgt(*-R_c)$ calcula as transições de chegada dos estados fora de G_R^E e *cont* limita o conjunto das transições ao conjunto das transições controláveis.

```
t_separacao := (rsrc(Rc) /\ rtgt(* - Rc)) /\ cont;
```

Figura 4.14: Equação no MEC para a determinação do conjunto de transições Ω que levam a marcações fora de G_R^E

A parte prática do passo 4 consiste resolver o PCS sobre o grafo G_A^E, utilizando as equações e funções definidas anteriormente. Um exemplo de saída de uma sessão do MEC, gerado através da função *ats*, é ilustrado na figura 4.15. No sistema de transições exportado, os estados não possuem propriedades booleanas, enquanto as transições continuam possuindo suas respectivas etiquetas. Além disso, as transições controláveis e não-controláveis são indicadas, respectivamente, pelas propriedades *cont* e *ncont*, enquanto as instâncias de separação de eventos também possuem a propriedade *t_separacao*. Esta última informação é de fundamental importância para a identificação destas transições no próximo passo.

O passo 5 é baseado na segunda parte da abordagem Ghaffari et al. [35] e consiste na geração de diversos modelos em programação linear inteira. Para a automação desta atividade, desenvolveu-se um programa na linguagem de programação C++. Em linhas gerais, este programa é um filtro cuja entrada consiste no arquivo exportado pelo MEC (o grafo G_R^E) através do comando *ats* e cuja saída é um modelo para cada instância de separação de eventos. Estes modelos são gerados na linguagem AMPL (*Modeling Language for Mathematical Programming*) [1], por ser uma linguagem de otimização amplamente utilizada por diversas ferramentas do gênero.

Em cada modelo, o programa desenvolvido gera expressões em AMPL relativas às equações 3.4, 3.5 e 3.6 sobre o grafo G_R^E. As expressões das condições de alcançabilidade são determinadas de forma trivial, por meio de uma busca dos estados da estrutura interna criada pelo algoritmo. A determinação das condições de separação de eventos é realizada com base nas transições que possuem a propriedade *t_separacao*, informação obtida a partir do arquivo exportado pelo MEC e também armazenada na estrutura interna. Por último, a determinação das equações de ciclo poderia ser computacionalmente complexa, caso não fosse suficiente encontrar somente os ciclos básicos. Alguns algoritmos desenvolvidos para a determinação dos ciclos básicos de um grafo foram apresentados por Welch [94], Gotlieb and Corneil [37] e Paton [73]. O primeiro é mais eficiente em termos de velocidade, enquanto o segundo é mais eficiente em termos de memória. O algoritmo de Paton [73], por sua vez, possui as melhores características dos algoritmos anteriores e, portanto, foi utilizado como base.

```
transition_system espec2;
0 |-
 0 -> 1 < property = ( a1, cont  ) >;

1 |-
 1 -> 2 < property = ( b1, ncont  ) >;

2 |-
 2 -> 3 < property = ( a1, cont, t_separacao  ) >,
 3 -> 7 < property = ( a2, cont  ) >;

7 |-
 10 -> 4 < property = ( a1, cont  ) >,
 11 -> 0 < property = ( b2, ncont  ) >;

4 |-
 5 -> 5 < property = ( b1, ncont  ) >,
 6 -> 1 < property = ( b2, ncont  ) >;

5 |-
 7 -> 6 < property = ( a1, cont, t_separacao  ) >,
 8 -> 2 < property = ( b2, ncont  ) >;
<
p11 = { 0, 2, 7, 5 };
p21 = { 0, 1, 2 };
r12 = { 0, 1, 7, 4 };
p12 = { 1, 4 };
r11 = { 2, 5 };
p22 = { 7, 4, 5 };
initial = { 0 };
Xm = { 0 };
Xb = {  };
Xc = { 0, 1, 2, 7, 4, 5 };
Rc = { 0, 1, 2, 7, 4, 5 };
e_separacao = { 2, 5 }
>.
```

Figura 4.15: Exemplo de saída de execução do MEC

Embora a geração automática dos modelos em programação matemática foi completamente implementada, o mesmo não pode ser dito com relação à resolução destes modelos, atividade do passo 6. Duas alternativas possíveis para a resolução automática destes modelos seriam:

- Incorporação de rotinas de baixo nível de algum otimizador no código fonte do mesmo programa que gera os modelos.
- Desenvolvimento de um script que realiza chamadas de algum otimizador para cada modelo gerado.

A alternativa que utiliza rotinas de baixo nível de algum otimizador possui a vantagem de ser mais automatizada e eficiente. Por outro lado, o conhecimento aprofundado de tais rotinas torna-se necessário. A utilização de um script é menos eficiente computacionalmente, embora o mesmo seja bem menos

custoso de ser desenvolvido. Devido à limitações temporais [12], decidiu-se pela segunda alternativa. Para realizar uma simples integração com o script implementado no próximo passo, o programa desenvolvido em C++ gera diversos arquivos texto, divididos em três classes:

- Um arquivo em AMPL contendo as equações de ciclo e as condições de alcançabilidade (equações 3.4 e 3.5 da abordagem Ghaffari et al. [35]) de G_R^E. Este arquivo é denominado modelo base.
- Um arquivo em AMPL para cada $t \in \Omega$ em G_R^E, contendo a respectiva equação da condição de separação de evento (equação 3.6 da abordagem Ghaffari et al. [35]).
- Um arquivo de comandos em AMPL para cada $t \in \Omega$.

O passo 6 consiste na resolução de cada modelo em programação linear inteira. De forma a proporcionar uma resolução automatizada, implementou-se um script em linux (modo *bash*) que realiza diversas instâncias do otimizador CPLEX [2] instalado na estação de trabalho Dantzig presente no Laboratório de Controle e Informática do Departamento de Automação e Sistemas [13]. Em linhas gerais, este script gera cada modelo ao concatenar ao final do modelo base o conteúdo do arquivo responsável por cada condição de separação de evento de G_R^E. Os arquivos recém criados são posteriormente resolvidos, através da chamada do arquivo de comandos em AMPL correspondente. O script em linux é ilustrado na figura 4.16. Mesmo sem entrar em detalhes sobre a sintaxe das funções utilizadas, observa-se claramente o princípio 'monta e resolve'. A solução de cada modelo consiste na descrição um lugar de controle ou na informação de que o problema é infactível. A descrição de um lugar de controle é feita por sua marcação inicial e pelos valores presentes na coluna correspondente da matriz de incidência de RdP-E.

```
cont=1
for file in Temp*
do
  echo "Resolvendo arquivo" $file
  cat Base Temp$cont > Modelo$cont.mod
  ampl R$cont.amp > Resultado$cont.txt
  let cont=cont+1
done

rm T*
rm Base
```

Figura 4.16: Script em linux para realização de diversas chamadas ao CPLEX

O passo 7 consiste na validação do passo anterior. Basta um problema em programação linear inteira ser infactível para o PCS não apresentar solução ótima através da utilização da Teoria das

[12] Cogitou-se estudar rotinas de baixo nível do otimizador CPLEX presente na estação de trabalho *Dantzig* do Laboratório de Controle e Microinformática do Departamento de Automação e Sistemas. No entanto, o tempo previsto para a compreensão destas rotinas era muito superior ao disponível para tal atividade.

[13] De forma não automatizada, estes modelos poderiam ser resolvidos individualmente utilizando os otimizadores remotos disponíveis no site *Neos Solvers*

Regiões. Caso contrário, existem lugares de controle que, após adicionados à RdP-E, fazem com que o comportamento em malha fechada seja equivalente ao grafo G_R^E, a solução minimamente restritiva.

O passo 8 possui o objetivo de reduzir o supervisor, através da eliminação dos lugares de controle redundantes. Foram visualizadas duas heurísticas para a verificação desta redundância. A primeira alternativa consiste em eliminar previamente todos os lugares de controle cujas restrições dos modelos correspondentes em programação linear inteira possam ser satisfeitas por outras soluções. A segunda alternativa consiste em adicionar todos os lugares de controle à RdP-E e eliminar aqueles cujas ausências não alteram o comportamento representado por G_R^E. Embora as duas opções sejam válidas, nenhuma foi implementada de forma automatizada. Ao final deste passo, os lugares de controle resultantes (aqueles que representam o supervisor reduzido) podem ser adicionados à RdP-E, constituindo, portanto, a RdP-S reduzida.

4.4 Discussão

Este capítulo apresentou a principal contribuição deste trabalho. A metodologia proposta acarretou em maior aproximação entre uma técnica específica de verificação formal (utilização de equações em μ-calculus) com a síntese de supervisores proposta pela abordagem RW. Além disso, os SEDs são modelados por RdP, o que contribui para as abordagens alternativas ao controle supervisório que não utilizam estruturas em autômatos.

Diversas otimizações são passíveis serem implementadas, tais como a consideração de pesos não unitários e a presença de mais de uma ficha nos lugares da RdP-E. Estes dois aspectos não foram considerados, mais devido a limitações das ferramentas utilizadas do que de restrições prévias da metodologia proposta. Outra melhoria considerável consiste na automatização da redução do supervisor, seja por qualquer uma das heurísticas apresentadas. Visando atender a generalização deste problema e/ou a simplificação da complexidade computacional, extensões da abordagem RW também poderiam ser implementadas. Esta área, portanto, revela um vasto campo de estudo.

Finalizando, o Capítulo 5 mostra os resultados de alguns exemplos práticos desta metodologia, enquanto o Capítulo 6 apresenta perspectivas para utilização de modelos cujos aspectos temporais são considerados.

Capítulo 5

Resultados

A metodologia proposta neste trabalho foi aplicada a diversos problemas. Inicialmente, foram utilizados alguns exemplos bastante comuns na literatura do PCS. O principal objetivo foi facilitar a comparação dos resultados obtidos com aqueles provenientes da aplicação direta da abordagem Ramadge e Wonham sobre SEDs modelados por autômatos, já que a complexidade computacional foi provada possuir a mesma ordem de grandeza. Os três exemplos apresentados são: linha de transferência simples, linha de transferência com retrabalho de peças rejeitadas e sistema AGV. Em todos estes casos, foi visto que o grafo de alcançabilidade da RdP supervisionada resultante é isomórfico ao autômato do supervisor obtido pela abordagem clássica.

Ao final deste capítulo, é resolvido o problema da célula de fabricação apresentada originalmente por [35]. Aqui, o objetivo foi ressaltar as principais diferenças entre as duas abordagens, que consistem basicamente na modelagem alternativa do SED e na utilização de equações em μ-calculus durante o processo de síntese.

5.1 Linha de transferência simples

Considera-se a linha de transferência ilustrada na figura 2.7. Este sistema é composto por três máquinas (M_1, M_2 e M_3) e dois *buffers* intermediários de capacidade unitária (B_1 e B_2). Para $i = 1,2,3$, consideram-se α_i como o início de operação da máquina M_i com a retirada de uma peça de B_{i-1} (quando existir), e β_i como o fim de operação da máquina M_i com o depósito de uma peça em B_i (quando existir). Deseja-se que não ocorra nem *overflow*, nem *underflow* nos *buffers* intermediários. Para isto, sabe-se que $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3\}$ e $\Sigma_u = \{\beta_1, \beta_2, \beta_3\}$.

5.1.1 Modelagem

A RdP-E deste SED é ilustrada na figura 5.1. A RdP-P é formada pela sub-rede constituída pelos lugares P, enquanto a RdP-EC é formada pela sub-rede constituída pelos lugares R. Os nomes

dos lugares de cada componente da planta (especificação de coordenação) possuem dois índices, um para representar a qual máquina (especificação de coordenação) se refere e outro para diferenciar os lugares de um mesmo componente. Cada máquina M_i é representada por dois lugares, um indicando o estado de repouso (P_{i1}) e outro indicando o processamento de uma peça (P_{i2}). Cada especificação de coordenação i é modelada pelos estados R_{i1} e R_{i2}, cujos arcos de entrada e saída indicam a alternância entre as transições β_i e α_{i+1}, necessariamente nesta ordem. Por último, considera-se que os lugares marcados são P_{11}, P_{21}, P_{31}, R_{12} e R_{22}.

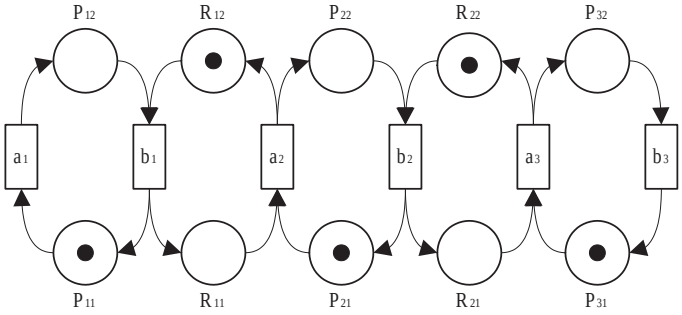

Figura 5.1: RdP-E para a linha de transferência simples

5.1.2 Síntese

O grafo de alcançabilidade G_A^E possui 32 estados, ou seja, a RdP-E possui 32 marcações diferentes. Como os lugares R_{12} e R_{22} da RdP-EC possuem arcos a transições não-controláveis, é necessário determinar os maus estados iniciais induzidos pelo comportamento desejado em malha fechada. Pela análise da RdP-P, β_1 está habilitada em toda marcação em que o lugar P_{12} possui ficha. Desta forma, qualquer marcação em RdP-E que viola esta condição é considerada mau estado. Análise análoga pode ser feita considerando o lugar P_{22} e a transição β_2.

Após a aplicação da abordagem Ziller and Schneider [100], obtém-se o grafo G_R^E. Este grafo possui 18 marcações, 6 ciclos básicos e 8 instâncias de separação de evento, sendo também isomórfico ao autômato do supervisor obtido pela abordagem Ramadge e Wonham. As seis equações de ciclo básico são idênticas e, portanto, redundantes. Os lugares de controle a serem adicionados a RdP-E são obtidos após a resolução de 8 modelos de programação linear inteira, um para cada instância de separação de evento.

Uma possível solução para a RdP-S é ilustrada na figura 5.2, onde os três lugares de controle adicionados a RdP-E correspondem à sub-rede constituída pelos lugares C. Prova-se que o lugar C_3 é redundante, podendo ser eliminado para a obtenção da RdP-S reduzida.

A tabela 5.1 ilustra algumas características de linhas de transferência simples com diferentes quantidades de máquinas. Para cada instância, são ilustradas a quantidade de marcações em G_A^E, a quantidade de marcações em G_R^E, a quantidade de ciclos básicos presentes em G_R^E e a quantidade de instâncias de separação de evento distintas que devem ser resolvidas pela abordagem Ghaffari et al. [35]. Para esta mesma classe de problemas, a tabela 5.2 apresenta a quantidade de lugares de controle

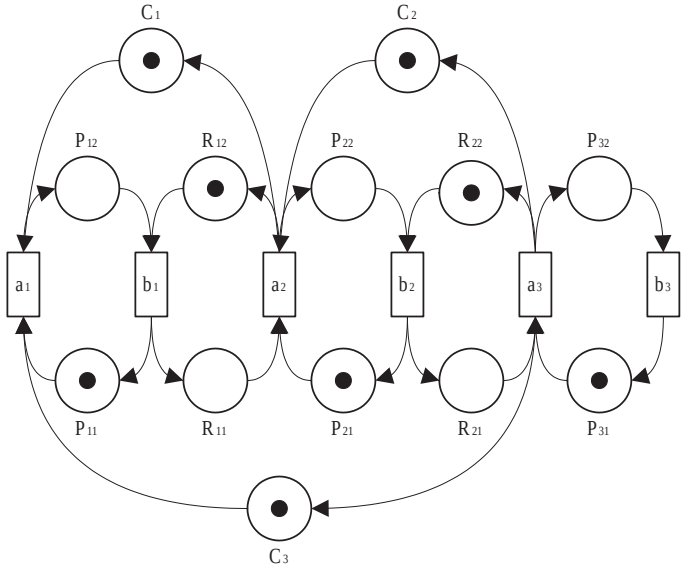

Figura 5.2: RdP-S para a linha de transferência simples

obtidos por uma possível solução e o tempo médio de resolução do conjunto de todos os modelos em programação linear inteira. Esta tabela também apresenta a quantidade de lugares de controle presentes na RdP-S reduzida. Em ambas as soluções, observa-se que a quantidade de lugares de controle a serem adicionados é muito inferior à quantidade de instâncias de separação de evento a serem resolvidas. Estas, por sua vez, também são muito menores em número do que a quantidade total de instâncias de separação de evento realmente existentes no grafo G_R^E.

Problema	G_A^E	G_R^E		
Quantidade de máquinas	**Marcações**	**Marcações**	**Ciclos básicos - Equações distintas**	**Instâncias de separação de evento** [1]
2	8	6	3-1	2
3	32	18	6-1	8
4	128	54	6-3	30
5	512	162	6-2	108
6	2048	486	6-2	378

Tabela 5.1: Características de diversas instâncias para a linha de transferência simples

Problema	**RdP-S possível**		**RdP-S reduzida**
Quantidade de máquinas	**Lugares de controle**	**Tempo de execução**	**Lugares de controle**
2	1	0.025s	1
3	3	0.078s	2
4	7	3.110s	3
5	9	12.670	4
6	22	77.150s	5

Tabela 5.2: Resultados envolvendo diversas instâncias para a linha de transferência simples

5.2 Linha de transferência com retrabalho de peças

Considera-se a linha de transferência com retrabalho de peças rejeitadas ilustrada na figura 2.2. Este sistema é composto por uma máquina M_1, um *buffer* unitário B_1, uma máquina M_2, um *buffer* unitário B_2 e uma máquina de medição M_3 que entrega as peças boas na saída da linha e retorna as peças rejeitadas no buffer B_1 para serem retrabalhadas. Para $i = 1,2,3$, consideram-se α_i como o início de operação da máquina M_i com a retirada de uma peça de B_{i-1} (quando existir), e β_i como o fim de operação da máquina M_i com o depósito de uma peça em B_i (quando existir). Já o evento β_4 representa o retorno de uma peça para ser retrabalhada. Deseja-se que não ocorra nem *overflow*, nem *underflow* nos buffers intermediários. Para isto, sabe-se que $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3\}$ e $\Sigma_u = \{\beta_1, \beta_2, \beta_3, \beta_4\}$.

5.2.1 Modelagem

A RdP-E deste SED é ilustrada na figura 5.3, onde a RdP-P é formada pela sub-rede constituída pelos lugares P, enquanto a RdP-EC é formada pela sub-rede constituída pelos lugares R. Para $i = 1,2$, a máquina M_i é modelada por dois lugares, um representando o estado de repouso (P_{i1}) e outro indicando o processamento de uma peça (P_{i2}). A modelagem de M_3 é um pouco mais complexa, pois também considera sinais internos do sistema que indicam se a peça foi fabricada com sucesso (β_{3S}) ou se a mesma está defeituosa e deve ser trabalhada (β_{4S}). Cada especificação de coordenação i é modelada pelos estados R_{i1} e R_{i2}, cujos arcos de entrada e saída indicam a alternância entre as transições que representam o depósito e a retida de peças do buffer B_i, necessariamente nesta ordem. Considera-se que os lugares marcados são P_{11}, P_{21}, P_{31}, R_{12} e R_{22}.

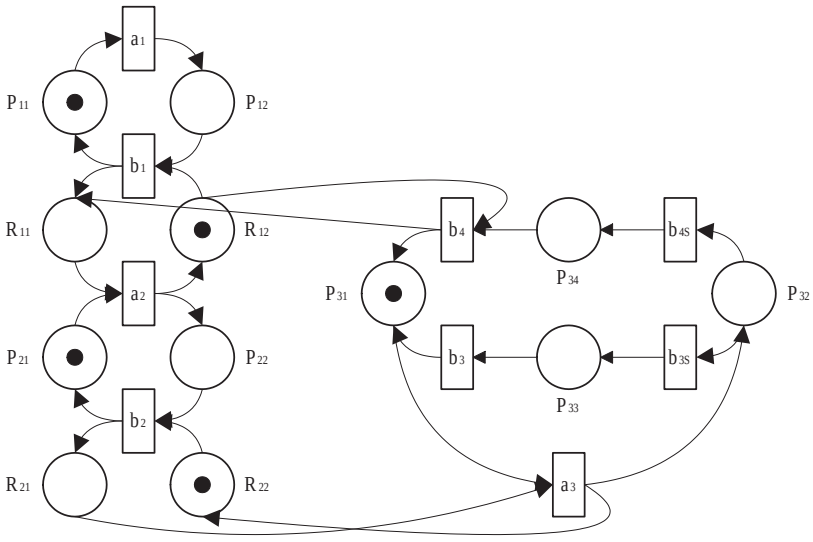

Figura 5.3: RdP-E para a linha de transferência com retrabalho de peças

5.2.2 Síntese

O grafo de alcançabilidade G_A^E possui 64 estados. Observando a RdP-EC, o lugar R_{12} possui arcos às transições não controláveis β_1 e β_4, enquanto o lugar R_{22} possui arco à transição não controlável

β_2. Desta forma, três situações podem induzir a maus estados iniciais em RdP-E:

- A marcação possui ficha em P_{12}, mas β_1 está impossibilitada de disparar.
- A marcação possui ficha em P_{22}, mas β_2 está impossibilitada de disparar.
- A marcação possui ficha em P_{34}, mas β_4 está impossibilitada de disparar.

O grafo G_R^E obtido após aplicação da abordagem Ziller and Schneider [100] possui 12 marcações, 4 ciclos básicos (envolvendo 2 equações distintas) e 8 instâncias de separação de evento, sendo também isomórfico ao autômato do supervisor obtido pela abordagem clássica. Dando continuidade ao processo de síntese, 8 modelos de programação linear inteira devem ser formulados e resolvidos para a obtenção dos lugares de controle.

Uma possível solução para a RdP-S corresponde a adição de 4 lugares de controle à RdP-E, o que é ilustrado na figura 5.4. Para facilitar a visualização, os lugares da RdP-E e seus respectivos arcos de entrada e saída não foram representados.

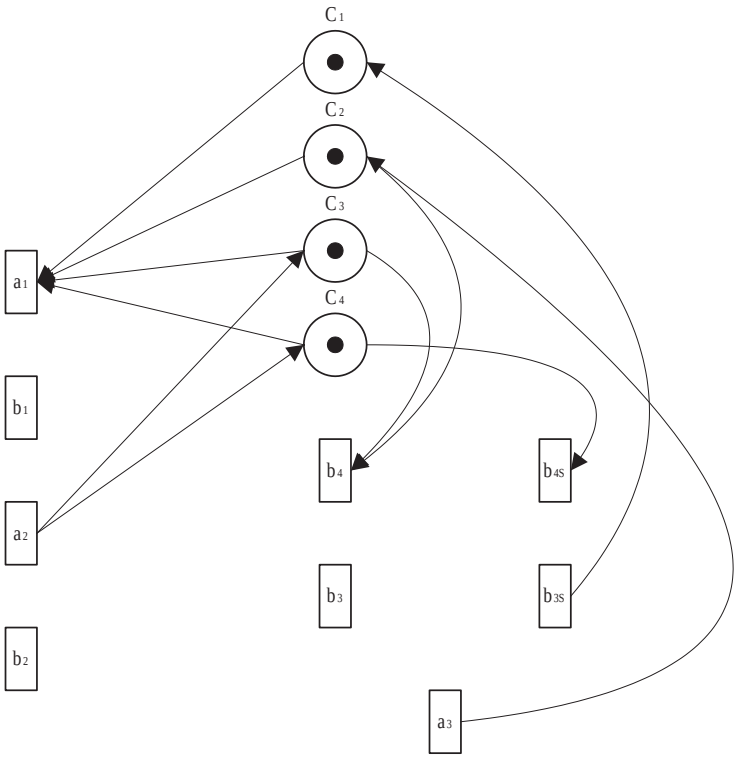

Figura 5.4: Lugares de controle presentes na RdP-S para a linha de transferência com retrabalho de peças

Em uma análise superficial, parece que o supervisor proposto tenta inibir a transição β_4 através dos lugares C_2 e C_3 e a transição β_{4S} através do lugar C_4. Uma análise cuidadosa levando em consideração os invariantes de lugar da RdP-S, enunciados na tabela 5.3, prova que o supervisor não desabilita nenhuma transição não-controlável. Suponha que β_4 esteja habilitada em RdP-E, mas esteja inibida

pelo lugar de controle C_2, ou seja, $M(P_{34}) \geq 1$ e $M(C_2) = 0$. Pela equação do sétimo invariante de lugar, tem-se

$$M(P_{12}) + M(P_{22}) + M(R_{11}) + M(R_{21}) = 1$$

Substituindo este resultado na equação do sexto invariante de lugar, chega-se à expressão

$$M(C_1) + M(P_{32}) + M(R_{34}) = 1,$$

que nitidamente contradiz a suposição inicial. Desta forma, embora possua um arco de saída à transição β_4, o lugar de controle C_2 não desabilita transições não controláveis. Análise semelhante pode ser feita com relação aos lugares de controle C_3 e C_4. Por último, observa-se que os lugares C_2, C_3 e C_4 são redundantes, podendo ser eliminados para a obtenção de um supervisor reduzido. A RdP-S reduzida é ilustrada na figura 5.5

	Invariantes de lugar	**Interpretação**
1)	$M(P_{11}) + M(P_{12}) = 1$	A máquina M_1 está em repouso ou processando uma peça.
2)	$M(P_{21}) + M(P_{22}) = 1$	A máquina M_2 está em repouso ou processando uma peça.
3)	$M(P_{31}) + M(P_{32}) + M(P_{33}) + M(P_{34}) = 1$	M_3 está em repouso, processando uma peça ou se preparando para depositá-la no devido lugar.
4)	$M(R_{11}) + M(R_{12}) = 1$	O *buffer* B_1 possui dimensão unitária, nunca ocorrendo *underflow*, nem *overflow*.
5)	$M(R_{21}) + M(R_{22}) = 1$	O *buffer* B_2 possui dimensão unitária, nunca ocorrendo *underflow*, nem *overflow*.
6)	$M(C_1) + M(P_{12}) + M(P_{22}) + M(P_{32}) + M(P_{34}) + M(R_{11}) + M(R_{21}) = 1$	Se uma das máquinas M_1 ou M_2 está processando uma peça ou se a máquina M_3 está verificando a qualidade de uma peça ou está na eminência em devolver uma peça para ser retrabalhada, então oss *buffer* B_1 e B_2 estão vazios.
7)	$M(C_2) + M(P_{12}) + M(P_{22}) + M(R_{11}) + M(R_{21}) = 1$	Se uma das máquinas M_1 ou M_2 está processando uma peça, então os *buffers* B_1 e B_2 estão vazios.
8)	$M(C_3) + M(P_{12}) + M(R_{11}) = 1$	Se a máquina M_1 está processando uma peça, então o *buffer* B_1 está vazio.
9)	$M(C_4) + M(P_{12}) + M(P_{34}) + M(R_{11}) = 1$	Se a máquina M_1 está processando uma peça ou a máquina M_3 está na eminência em devolver uma peça para ser retrabalhada, então o *buffer* B_1 está vazio.

Tabela 5.3: Invariantes de lugar da RdP-S da linha de transferência com retrabalho de peças

5.3 Sistema AGV

Considera-se o sistema AGV ilustrado na figura 5.6. Este sistema é composto por uma máquina M_1, uma máquina M_2, um veículo autônomo *AGV* (Auto Guided Vehicle) e uma esteira automática. Para $i = 1, 2$, consideram-se α_i e β_i como o início e o fim de operação da máquina M_i, respectivamente. O veículo autônomo *AGV* só é capaz de transportar peças da esquerda para a direita, movimentando sem carga no sentido inverso. Em maiores detalhes, o *AGV* carrega uma peça produzida por M_1 através de β_1 e a transporta até M_2 através de α_2 ou até a esteira através de γ. Além disso, carrega uma peça produzida por M_2 através de β_2 e a transporta até a esteira através de γ. Como especificação

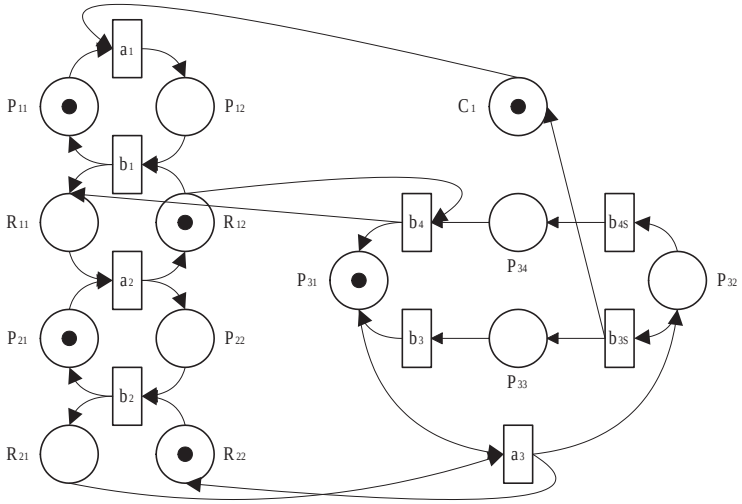

Figura 5.5: RdP-S reduzida para a linha de transferência com retrabalho de peças

de coordenação, deseja-se que toda peça produzida seja processada pelas máquinas M_1 e M_2, nesta ordem. Do conhecimento do SED, sabe-se que $\Sigma_c = \{\alpha_1, \alpha_2, \gamma\}$ e $\Sigma_u = \{\beta_1, \beta_2\}$.

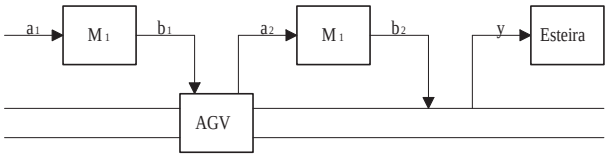

Figura 5.6: Sistema AGV

5.3.1 Modelagem

A RdP-E deste SED é ilustrada na figura 5.7, onde é mantido o mesmo padrão para a nomenclatura dos estados de RdP-P e de RdP-EC. Para $i = 1,2$, a máquina M_i é modelada pelos lugares P_{i1} e P_{i2}, que representam os estados de repouso e processamento de peças, respectivamente. O veículo autônomo *AGV* é modelado por três lugares: P_{31} indica o estado de repouso, enquanto P_{32} (P_{33}) indica o carregamento de uma peça proveniente de M_1 (M_2). As transições γ_1 e γ_2 correspondem ao evento γ, ou seja, ao transporte de uma peça até a esteira. Para manter a modelagem coerente, o lugar R_{11} de RdP-EC possui a mesma estrutura de arcos com relação a estas duas transições. A especificação de coordenação é modelada através da alternância das transições β_1 e α_2, nesta ordem, e da proibição da ocorrência imediata das transições γ_1 e γ_2 após a ocorrência de β_1. Considera-se que os lugares marcados são P_{11}, P_{21}, P_{31}, R_{11} e R_{12}. Devido à existência do invariante de lugar $M(R_{11}) + M(R_{12}) = 1$, pode-se abstrair a marcação dos lugares R_{11} e R_{12} durante o processo de síntese.

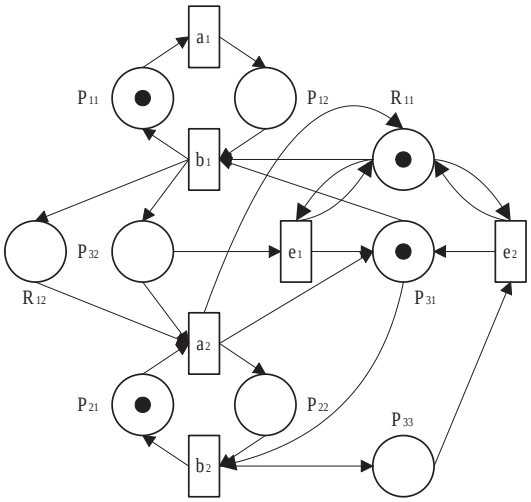

Figura 5.7: RdP-E para o sistema AGV

5.3.2 Síntese

O grafo de alcançabilidade G_A^E possui 9 estados. A única transição não-controlável cujo disparo pode ser inibido por um lugar de RdP-EC é β_1. Desta forma, toda marcação em RdP-E que possui ficha em P_{12} e não habilita β_1 é considerada mau estado inicial.

O grafo G_R^E obtido após a aplicação da abordagem Ziller and Schneider [100] possui apenas 6 marcações, 2 ciclos básicos (embora todos possuindo a mesma equação) e 2 instâncias de separação de evento. Esta grafo é isomórfico ao autômato do supervisor obtido pela abordagem clássica e está ilustrado na figura 5.8. A resolução deste problema é extremamente simples e a RdP-S reduzida é ilustrada na figura 5.9. Outra opção para um supervisor consiste na adição mais um lugar de controle com marcação unitária e conectado à transição α_1 por um arco de entrada e à transição β_2 por um arco de saída. Embora não seja mínima no número de lugares, esta solução é tão permissiva quanto a solução reduzida.

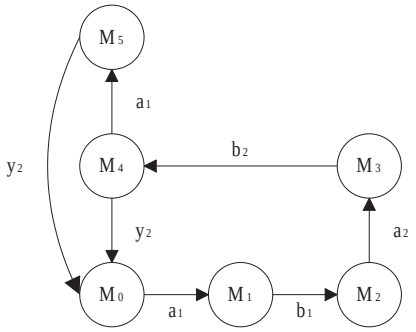

Figura 5.8: G_R^E para o sistema AGV

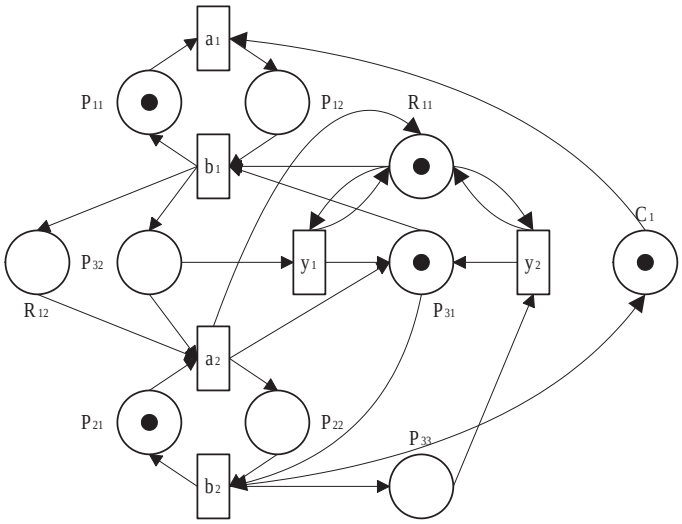

Figura 5.9: RdP-S para o sistema AGV

5.4 Célula de fabricação

Considera-se a célula de fabricação ilustrada na figura 3.1 [35]. Este sistema é composto por uma máquina M, um *buffer* intermediário de capacidade unitária B e dois veículos autônomos AGV_1 e AGV_2. A máquina M é considerada pouco confiável e pode apresentar defeitos durante a fabricação de alguma peça. Independentemente da peça produzida ser classificada como boa ou defeituosa, ela é depositada no buffer B. O veículo autônomo AGV_1 é responsável por descarregar uma peça classificada como boa, enquanto o AGV_2 descarrega peças defeituosas. Como especificação de coordenação, deseja-se que o acesso ao buffer B não seja feito simultâneamente pelos dois veículos. A explicação de todas as transições envolvidas é dada na tabela 3.1. Sabe-se também que as únicas transições não-controláveis são t_2 e t_6.

5.4.1 Modelagem

Neste problema, a modelagem do SED é completamente diferente daquela apresentada por Ghaffari et al. [35]. Ao invés de modelar o SED por uma única RdP, cada componente assíncrono é modelado por uma RdP individual e o conjunto destas redes constitui a RdP-P. Para efeitos de comparação, todos as etiquetas das transições possuirão o mesmo significado físico do exemplo tratado por Ghaffari et al. [35].

A RdP da máquina M é ilustrada na figura 5.10. Os lugares P_{11}, P_{12} e P_{13} possuem, respectivamente, a mesma interpretação dos lugares p_{13}, p_1 e p_7 apresentados na tabela 3.1, ou seja, M em repouso, M ocupada e M com defeito.

As RdP dos dois veículos autônomos possuem exatamente a mesma estrutura cíclica. Para $i = 2,3$, P_{i1} indica o estado de repouso, P_{i2} indica o acesso ao *buffer* e P_{i3} indica o carregamento de uma peça. Estas RdP são ilustradas na figura 5.11.

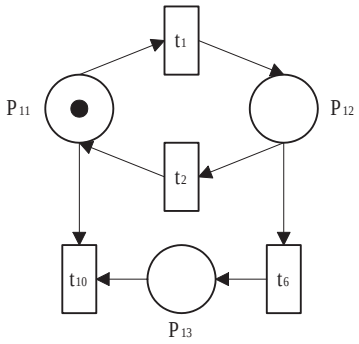

Figura 5.10: RdP para a máquina *M* da célula de fabricação

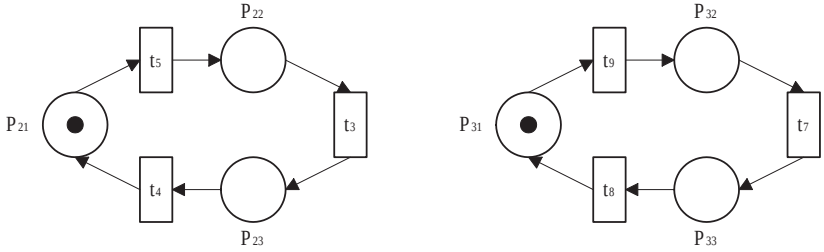

Figura 5.11: RdP para os veículos autônomos AGV_1 e AGV_2 da célula de fabricação

A RdP-EC está ilustrada na figura 5.12. Basicamente, ela restringe o acesso simultâneo dos veículos autônomos ao *buffer*, ao mesmo tempo que associa corretamente cada veículo à qualidade da última peça produzida. Respeitando as características do SED apresentado na figura 3.1, o AGV_1 é responsável por transportar as peças classificadas como boas, enquanto o AGV_2 é responsável por transportar as peças defeituosas. A composição da RdP-EC com a RdP-P constitui a RdP-E.

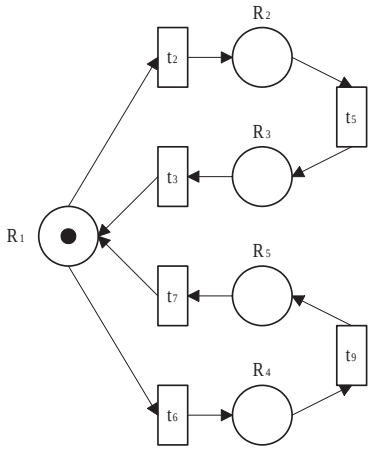

Figura 5.12: RdP-EC para a célula de fabricação

5.4.2 Síntese

O grafo de alcançabilidade G_A^E possui 41 estados, 7 a menos do que o grafo de alcançabilidade obtido por Ghaffari et al. [35]. Dois estados nitidamente excluídos nesta abordagem são as duas

marcações de bloqueio presentes na modelagem anterior. Analisando a RdP-EC, observa-se que as duas transições não-controláveis possuem arcos de entrada provenientes do lugar R_1. Assim, toda marcação em RdP-E que possui ficha no lugar P_{12} e não habilita as transições t_2 e t_6 devem ser considerados maus estados iniciais.

O grafo G_R^E obtido após a aplicação da abordagem Ziller and Schneider [100] é isomórfico ao grafo sob supervisão obtido por Ghaffari et al. [35]. Ambos possuem 30 marcações, 12 ciclos básicos (envolvendo 6 equações distintas) e 12 instâncias de separação de evento. No entanto, não foi possível a determinação de um supervisor por meio da adição de lugares de controle, pois a resolução dos 12 modelos de programação linear inteira mostrou se tratar de problemas infactíveis.

5.5 Análise da aplicabilidade da metodologia proposta

Buscou-se entender o motivo pelo qual a metodologia proposta neste documento não se aplica em sua integridade ao problema da célula de fabricação. Como a Teoria das Regiões foi originalmente apresentada em [6] sob um ponto de vista teórica da ciência da computação, procurou-se alguma explicação empírica com base nos problemas que foram efetivamente solucionados com a metodologia proposta. No entanto, não foi encontrado nenhum aspecto relevante que levasse a alguma dedução significativa. Todas as análises foram feitas considerando a estrutura geral da RdP-E, pois a aplicação da Teoria das Regiões considera somente marcações, caminhos e ciclos de um grafo de alcançabilidade, não se importando como tal grafo foi gerado. Em outras palavras, nenhuma diferenciação é feita entre a RdP-P e a RdP-EC.

Pensando em termos de grafos de estados e grafos de eventos, o problema da linha de transferência simples foi o único que apresentou uma dessas características peculiares. Sua RdP-E era um grafo de eventos, pois cada lugar possuía exatamente um arco de entrada e um arco de saída. Todas as demais RdP-E possuíam quantidades distintas de arcos ligando lugares e transições. Também não foi encontrado nenhuma correspondência entre a aplicabilidade desta metodologia com aspectos quantitativos das RdP-EC, tais como proporções entre transições controláveis e não controláveis, transições e lugares, instâncias de separação de evento e marcações, etc. As tabelas 5.4 e 5.5 são complementares e apresentam algumas destas características para alguns problemas tratados. Os dois problemas não apresentados neste capítulo são descritos em [80].

Problema	**Característica peculiar**	**Lugares**	**Transições** (T_c-T_u)	**Arcos**
Linha de transferência simples (2 máq.)	Grafo de eventos	6	2-2	12
Linha de transferência com retrabalho de peças	Não	12	3-6	28
Sistema AGV	Não	9	4-2	26
Recursos compartilhados por dois usuários	Não	12	4-2	32
Célula de fabricação	Não	14	8-2	32
Linha de transferência industrial	Não	20	6-6	42

Tabela 5.4: Características de alguns problemas tratados (I)

Por último, foi descoberta uma a solução interessante para o problema da célula de fabricação. Embora sua RdP-S não possa ser obtida pela adição de lugares de controle, a adição de dois arcos

Problema	Marcações	Instâncias de separação de evento	Resolvido
Linha de transferência simples (2 máq.)	6	2	Sim
Linha de transferência com retrabalho de peças	12	8	Sim
Sistema AGV	6	2	Sim
Recursos compartilhados por dois usuários	7	4	Sim
Célula de fabricação	30	12	Não
Linha de transferência industrial	212	234	Não

Tabela 5.5: Características de alguns problemas tratados (II)

[2] à RdP-E é suficiente para que o comportamento resultante seja equivalente ao grafo G_R^E obtido no decorrer da metodologia. Estes arcos foram obtidos empiricamente e representam um *self-loop* entre o lugar R_1 e a transição t_1. A RdP-S resultante é ilustrada na figura 5.13. Além de fazer uso de uma modelagem mais intuitiva, esta solução é um pouco mais compacta do que a obtida em Ghaffari et al. [35], pois possui um lugar e um arco a menos em sua estrutura.

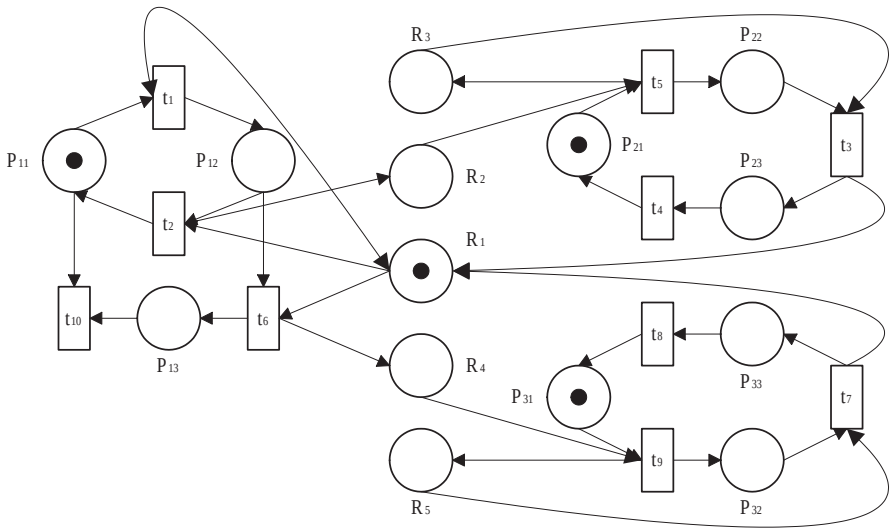

Figura 5.13: RdP-S para a célula de fabricação

5.6 Discussão

A metodologia proposta apresentou resultados satisfatórios na maioria dos exemplos tratados. Quando a metodologia falhou, é facilmente observado que tal falha ocorreu na etapa da aplicação direta da Teoria das Regiões. Com isso, prova-se que aspectos da verificação formal podem certamente serem incorporados ao processo de síntese de supervisores, o que integra as duas classes de técnicas envolvendo SEDs.

Um estudo minucioso da Teoria das Regiões, da forma como foi proposta por [6], poderia trazer justificativas para sua ausência de total aplicabilidade nos problemas em geral. Mesmo assim, em

[2] Para se adequar a terminologia adotada, estes arcos poderiam ser chamados de arcos de controle.

algumas situações pode-se determinar uma RdP supervisionada de forma empírica, desde que se tenha o grafo G_R^E. Por último, a ausência de exemplos onde as RdP não são binárias é mais devido à deficiência de ferramentas do que a limitações na metodologia proposta. No entanto, ainda não se pode garantir que não existam restrições para o envolvimento de tais redes no processo de síntese.

Capítulo 6

Perspectivas para a utilização de Redes de Petri Temporais

A especificação e análise de sistemas concorrentes cujos comportamentos são dependentes de valores explícitos de tempo levou à definição de diversas extensões às RdP, de forma a considerar explicitamente aspectos temporais. Dentro deste contexto, duas importantes extensões são as redes de Petri temporizadas [84] e as redes de petri temporais [65]. No primeiro modelo, é associada uma duração de disparo às transições [1], enquanto no último é associado uma duração de sensibilização às transições. Através da representação do mecanismo de *watchdog*, prova-se que o modelo da rede de Petri temporal é mais geral que o da rede de Petri temporizada [19]. Por este motivo, somente a segunda alternativa será considerada neste trabalho.

Após uma breve apresentação dos aspectos mais importantes da rede de Petri temporal e seu grafo de classes de estados associado, o restante deste capítulo discorre sobre perspectivas de utilização deste modelo no PCS, de forma a enriquecer o processo de síntese com informações temporais. Em abordagem análoga, Brandim and Wonham [17] generalizam a abordagem Ramadge and Wonham [82], ao permitir a utilização de SEDs modelados por autômatos temporizados [3]. Sava and Alla [85] resolvem o PCS em SEDs modelados por redes de Petri temporais, mas realizam o processo de síntese sobre autômatos temporizados que possuem o mesmo comportamento destas redes.

6.1 Redes de Petri Temporais

Uma rede de Petri Temporal é uma extensão de uma rede de Petri, associando a cada transição uma duração de sensibilização. O disparo de uma transição é considerado instantâneo, mas a mesma deve estar sensibilizada durante o intervalo de tempo dado. Ao associar um intervalo (θ_{MIN}, θ_{MAX}) a cada transição, considera-se que a duração de sensibilização deve ser maior que θ_{MIN} e menor

[1] Também existem extensões com lugares temporizados e com arcos temporizados. A equivalência entre as extensões com transições temporizadas e com lugares temporizados é facilmente comprovada [19]

que θ_{MAX}. Isto significa que, após estar sensibilizada, tal transição deve disparar após um instante de tempo superior a θ_{MIN}, mas inferior a θ_{MAX}. Diversas abordagens foram propostas para uma representação compacta dos estados de uma rede de Petri temporal. Dentre elas tem-se o grafo linear de classes de estados, proposto por Berthomieu and Menasche [11].

6.1.1 Definições

Uma rede de Petri temporal marcada é uma dupla

$$N_T =< N, IED >, \tag{6.1}$$

onde N é uma RdP e $IED : T \rightarrow (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty))$ é uma função que associa um intervalo estático de disparo a cada transição. A cada transição t_i é associado um par de valores $(\theta^E_{min}(i), \theta^E_{max}(i))$, onde $\theta^E_{min}(i)$ e $\theta^E_{max}(i)$ são valores racionais positivos tais que $\theta^E_{min}(i) \leq \theta^E_{max}(i)$, $0 \leq \theta^E_{min}(i) < \infty$ e $0 \leq \theta^E_{max}(i) \leq \infty$. A partir de agora, rede de Petri temporal marcada será denotada por RdPT.

O estado de uma RdPT é representado como um par $S = (M, I)$, onde M é uma marcação e I é um vetor de todos possíveis intervalos de disparo. Cada entrada do vetor I possui um par de valores de tempo para cada transição sensibilizada. Em geral, os intervalos dinâmicos de disparo $(\theta_{min}(i), \theta_{max}(i))$ são diferentes dos intervalos estáticos $(\theta^E_{min}(i), \theta^E_{max}(i))$. Por questões de comodidade, o termo dinâmico será omitido nas notações adiante.

O disparo de uma transição t_i no tempo θ, a partir de um estado $S = (M, I)$, é possível sse:

- A transição t_i está sensibilizada em M e
- o tempo θ está compreendido entre $\theta_{min}(i)$ e o menor valor $\theta_{max}(k)$ entre todas as transições t_k sensibilizadas.

A primeira condição é herdada das definições de uma RdP, enquanto a segunda resulta da necessidade de disparo das transições dentro dos intervalos de disparo. Este disparo leva a um estado $S' = (M', I')$, cujos elementos são calculados independentemente:

1. Calcula-se a nova marcação M' pela equação fundamental da RdP, ou seja, $M' = M - Pre(., t_i) + Post(., t_i)$.
2. Calcula-se o vetor dos novos intervalos de disparo I' por meio de três regras:
   - Todas transições não sensibilizadas pela marcação M' recebem intervalos de disparo vazio.
   - Para todas transições t_k sensibilizadas pela marcação M e não conflitantes com t_i, o intervalo de disparo é dado por $(max(0, \theta_{min}(k) - \theta), \theta_{max}(k) - \theta)$.

- Todas as outras transições recebem intervalos idênticos aos respectivos intervalos estáticos de disparo.

Da mesma forma como é definida para RdP, uma seqüência σ corresponde a uma seqüência de transições disparadas sucessivamente. Em RdPT, no entanto, uma informação temporal também deve ser considerada, o que faz com que uma lista de disparos consista em uma seqüência de pares (*transição*, *instante de tempo*). A regra de disparo define uma relação de alcançabilidade entre os estados de uma RdPT e o comportamento da rede é caracterizado pelo conjunto de estados alcançáveis a partir do estado inicial S_0. Devido à relação entre os intervalos de disparo das diversas transições, uma RdPT normalmente possui um número irrestrito de estados, não sendo possível a representação de seu comportamento através de um grafo de alcançabilidade $G_A(N_T, S_0)$. Desta forma, se faz necessária uma representação compacta dos estados de uma RdPT.

6.1.2 Classes de estados

Visando uma representação compacta dos estados de uma RdPT, Berthomieu and Menasche [11] propuseram o grafo linear de classes de estados. Como características, este grafo preserva propriedades de alcançabilidade e propriedades temporais de tempo linear. Ao invés de considerar um estado alcançável a partir do estado inicial por meio de uma seqüência de valores temporais relacionados a uma seqüencia de disparo σ, uma classe de estados considera o conjunto de todos os estados alcançáveis a partir do estado inicial por meio do disparo de todos os valores possíveis correspondendo à mesma seqüencia σ. Em suma, este conjunto de estados é considerado um pseudo-estado agregado, sendo denominado classe de estados associado à seqüencia de disparo σ. Este conceito pode ser ilustrado pela figura 6.1 [10].

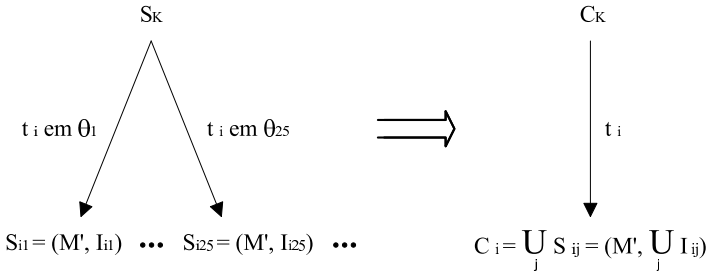

Figura 6.1: Representação do conceito de classe de estado

Dado um estado S_k, o disparo de uma transição t_i em diferentes instantes de tempo θ_j leva a diferentes estados S_{ij}, todos com a mesma marcação M'. O objetivo da classe de estado C_i é agrupar estes estados resultantes. A aplicação em cadeia deste conceito faz com que o comportamento de uma RdPT seja representado por um grafo finito.

Uma classe de estado de uma RdPT é representada como um par $C = (M, D)$, onde M é uma marcação alcançável a partir da marcação inicial pelo disparo de uma seqüência σ e D é a união de todos os domínios de disparo dos estados alcançáveis a partir do estado inicial, pelo disparo das listas com a seqüência σ. Domínios em classes de estados expressam um conjunto de intervalos para cada

transição habilitada e, conseqüentemente, relações entre os tempos de disparo destas transições. Um domínio pode ser expresso como o conjunto solução do sistema de inequações lineares $At \leq b$, onde A é uma matriz de constantes, b é um vetor de constantes e t é um vetor de variáveis correspondendo às transições sensibilizadas pela marcação M [11].

O disparo de uma transição t_i, a partir de uma classe [2] $C = (M, D)$, é possível sse [11]:

- A transição t_i está sensibilizada em M e
- existe no domínio D um vetor cuja componente correspondendo à transição t_i não é superior a qualquer outro componente [3].

Novamente, a primeira condição é herdada das definições de uma RdP. Já a segunda condição expressa que a transição t_i é disparada dentro do seu intervalo permitido, além de ser disparada antes de todas as outras transições t_j habilitadas. Este disparo leva a uma classe $C' = (M', D')$, cujos elementos são calculados independentemente [11]:

1. Calcula-se a nova marcação M' pela equação fundamental da RdP, ou seja, $M' = M - Pre(.,t_i) + Post(.,t_i)$.
2. Calcula-se o novo domínio D' por meio de quatro passos:
   - Acrescente as condições de disparo para a transição t_i ao sistema $At \leq b$.
   - Elimine do sistema as variáveis associadas às transições conflitantes com t_i.
   - Expresse cada variável permanecente t_j, com $j \neq i$, como a soma da variável t_i e a nova variável t'_j, e elimine do sistema todas antigas variáveis, incluindo t_i.
   - Adicione no sistema resultante uma variável para cada transição recém sensibilizada, restrinjindo-a ao intervalo estático de disparo associado a tal transição.

A relação de alcançabilidade definida pela regra de disparo permite a construção de uma árvore de classes de estados na qual a raiz corresponde à classe inicial e existe um arco rotulado com a transição t_i, indo da classe C à classe C', sse a transição t_i é disparável em C e seu disparo leva à classe C'. O grafo de classes de estados $G_{CE}(N_T, C_0)$ é obtido a partir desta árvore, pela fusão de classes equivalentes. Duas classes são equivalentes sse suas marcações são idênticas, bem como seus domínios de disparo. A comparação entre domínios de disparo pode ser feita em tempo polinomial [5].

Um grafo de classes de estados pode ser facilmente mapeado em um autômato temporizado. Sem entrar em maiores detalhes, este mapeamento é realizado através das seguintes regras:

[2] Para simplificar a notação, uma classe de estado será denominada apenas classe a partir deste instante.

[3] Seja D o conjunto solução de algum sistema $At \leq b$ e seja t_i a i-ésima transição sensibilizada, então a segunda condição é verdadeira sse o sistema de inequações é consistente, ou seja, $At \leq b$ e $t_i \leq t_j, \forall t_j, j \neq i$ [11].

- Desconsiderando aspectos temporais, o autômato temporizado será isomórfico ao grafo de classes de estados da RdPT.
- Existirá somente um relógio global x no autômato. A cada transição do autômato, o valor deste relógio é reinicializado.
- É associada uma guarda $x \geq \theta_{MIN}$ a cada transição do autômato, onde θ_{MIN} corresponde ao limite mínimo do intervalo de disparo da respectiva transição no grafo.
- É associada um invariante $x \leq \theta_{MAX}$ a cada lugar do autômato, onde θ_{MAX} corresponde ao limite máximo comum de todas as transições de saída da respectiva classe no grafo.

Alguns autores realizam um mapeamento diferente entre RdPT e autômatos temporizados. O autômato temporizado resultante do mapeamento proposto por Sava and Alla [85] possui duas grandes diferenças. Primeiramente, cada lugar do autômato está associado a uma marcação da RdPT (e não a uma classe do grafo de classes de estados associado). Conseqüentemente, a estrutura do autômato temporizado não é isomórfica à estrutura do grafo de classes de estados da RdPT. Em segundo lugar, o autômato possui um relógio global distinto para cada transição presente na RdPT. Um relógio é reinicializado em uma transição do autômato caso a transição associada na RdPT seja recém sensibilizada. Em outro trabalho, Cassez and Roux [21] definem um autômato A_i para cada transição t_i presente na rdPT. Os estados destes autômatos representam a situação das respectivas transições: habilitada, desabilitada ou sendo disparada. Cada autômato A_i possui um relógio global e seu estado inicial depende da marcação inicial da RdPT. A cada transição de um autômato A_i, um vetor que indica a quantidade de fichas nos lugares da RdPT é atualizado. Também é definido um autômato auxiliar [4], cuja importânica se dá durante o produto síncrono dos autômatos A_i, operação que preserva a bissimilaridade temporal.

6.1.3 Exemplo

Considera-se a RdPT hipotética ilustrada na figura 6.2 [86]. Os intervalos estáticos de disparo são todos fechados e estão representados próximos às suas respectivas transições. O grafo de classes de estados correspondente está ilustrado na figura 6.3. Com relação aos aspectos temporais, estão representados somente os intervalos em que cada transição pode efetivamente ser disparada, ou seja, os domínios de cada classe não são mostrados em sua integridade. A tabela 6.1, por sua vez, contem uma descrição detalhada de cada classe.

A classe inicial C_0 compreende uma marcação com apenas uma ficha no lugar P_1 e um domínio no qual somente a transição t_1 está habilitada. Esta transição pode ser disparada imediatamente ou no máximo em um intervalo de 10 unidades de tempo. Pela análise das classes C_1 e C_2, observa-se que os intervalos de disparo presentes nos domínios são mais abrangentes do que os intervalos mostrados no grafo.

[4] No documento original foi utilizado o termo supervisor. Neste documento, optou-se por utilizar o termo auxiliar, para evitar confusões com a terminologia adotada na abordagem Ramadge and Wonham [82].

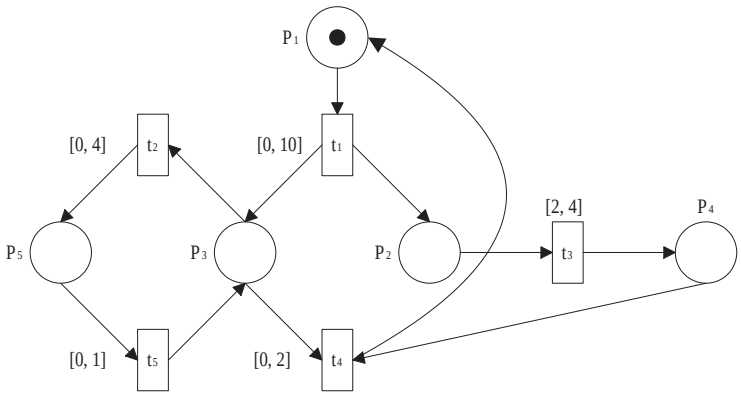

Figura 6.2: Rede de Petri Temporal hipotética

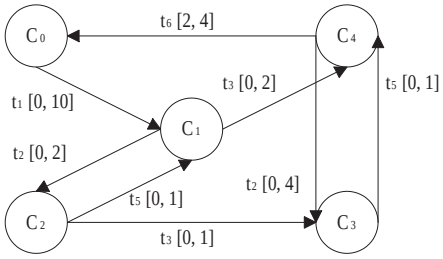

Figura 6.3: Grafo de classes de estados para a RdPT hipotética

Schastai et al. [86] fizeram uma importante observação com respeito à análise dos aspectos temporais em uma RdPT. Dados uma classe C e uma seqüência σ, o cálculo do maior intervalo de tempo em que todas as transições presentes em σ são disparadas seqüencialmente não pode ser realizado baseando-se simplesmente no grafo de classes de estados associados a esta rede. Como exemplo, considera-se a seqüencia $\sigma = t_1t_2t_3t_5t_6$. Pela análise do grafo de classes de estados da figura 6.3, a soma dos limites superiores θ_{max} associados a cada transição resulta no valor 18. Já pela análise da RdPT da figura 6.2, verifica-se que o maior intervalo de tempo é na realidade 17. Schastai et al. [86] propõe então uma metodologia que permite a determinação dos tempos válidos para todos os eventos contidos em uma seqüencia σ em um grafo. Sem esta metodologia, as análises temporais em grafos de classes de estados não podem envolver uma seqüencia que contenha mais de uma transição.

6.2 Perspectivas

Na tentativa de resolver o PCS para SEDs modelados por RdPT, há grande interesse na utilização do grafo de classes de estados proposto por Berthomieu and Menasche [11], pois o mesmo preserva propriedades de alcançabilidade e propriedades temporais de tempo linear, além de possuir um número finito de classes. Inicialmente, pensou-se na utilização de uma metodologia baseada naquela proposta para SEDs modelados por RdP, ou seja, para sistemas em que aspectos temporais não são tão relevantes. Uma das principais vantagens daquela metodologia era a não necessidade de utilização do grafo de alcançabilidade da planta durante o processo de síntese do supervisor. A condição necessária e suficiente para a determinação dos maus estados iniciais era a identificação

Classe	Marcação	Domínio
C_0	P_1	$0 \leq t_1 \leq 10$
C_1	P_2 P_3	$0 \leq t_2 \leq 4$
		$0 \leq t_3 \leq 2$
C_2	P_2 P_5	$0 \leq t_3 \leq 2$
		$0 \leq t_5 \leq 1$
C_3	P_4 P_5	$0 \leq t_5 \leq 1$
C_4	P_3 P_3	$0 \leq t_2 \leq 4$
		$2 \leq t_4 \leq 4$

Tabela 6.1: Descrição das classes da RdPT hipotética

de quais situações uma dada transição não-controlável ficava impossibilitada de disparar devido ao acréscimo de um lugar proveniente da RdP-EC. Em outras palavras, um estado seria classificado como mau estado inicial caso alguma transição não-controlável ficasse impossibilitada de disparar devido à inexistência (ou insuficiência) de fichas em pelo menos um lugar da RdP-EC que fosse entrada desta transição.

No entanto, ao analisar SEDs modelados por RdPT, esta condição deixa de ser suficiente, pois não se sabe se uma transição ficou impossibilitada de disparar devido à marcação da RdPT-EC ou se já estava impossibilitada devido à dependência mútua dos intervalos de disparo da RdP-P. Além disso, para definir uma má classe [5] no grafo de classes de estados da especificação, não basta identificar unicamente a inibição de transições não-controláveis que eram possíveis de ocorrer no sistema em malha aberta. Classes que restringem o intervalo de disparo de alguma destas transições também devem ser consideradas más classes, pois, usando a terminologia anterior, estas classes inibem o disparo de uma transição não-controlável durante um certo intervalo de tempo.

Outro fator a levar em consideração é que o mapeamento entre as classes dos grafos de classes de estados da planta e da especificação não pode ser realizado de forma implícita, assim como era realizado na metodologia anterior. Nesta, um estado do grafo de alcançabilidade da especificação é identificado somente por sua marcação e o mapeamento é trivial, pois a marcação do grafo de alcançabilidade da especificação contém a marcação do grafo da planta. Em outras palavras, a única diferença na marcação do grafo de alcançabilidade da especificação é devido à presença de fichas em lugares da RdP-EC. Em grafos de classes de estados, uma classe é identificada por uma marcação e um domínio. Como a determinação de um domínio normalmente não pode ser realizada através de uma análise superficial dos intervalos estáticos de disparo das transições de uma RdPT, o mapeamento entre estados da planta e da especificação depende explicitamente do grafo de classes de estados da planta, o que não era necessário para SEDS modelados por RdP. Devido aos fatores mencionados, chega-se à conclusão que o grafo de classes de estados da planta é necessário para a realização do mapeamento entre classes e, conseqüentemente, para a identificação das más classes.

Inicialmente, pensou-se na utilização de sistemas modelados por RdPT que não possuíssem nenhuma restrição nos intervalos estáticos de disparo das transições. Assim, os limites mínimos e máximos de disparo poderiam possuir quaisquer valores, desde que respeitando a definição de uma RdPT. O mapeamento entre os grafos de classes de estados seria então realizado da forma tradicional

[5] A definição de uma má classe em RdPT é uma extensão da definição de mau estado em RdP.

utilizando-se a teoria das linguagens e autômatos, pois é facilmente provado que um grafo de classes de estados é sempre determinístico, ou seja, dado uma certa classe, o disparo de uma transição levará sempre à mesma classe. Desta forma, a correspondência entre as classes iniciais seria imediata, enquanto as demais classes seriam mapeadas por meio das seqüências de disparo a partir das classes iniciais. Dadas duas classes já mapeadas, C_i^P e C_i^E, presentes respectivamente nos grafos de classes de estados da planta e da especificação, o mapeamento entre as classes C_j^P e C_j^E seria dado por meio da regra $C_i^E[t_i \rightarrow C_j^E \Rightarrow C_i^P[t_i \rightarrow C_j^P$. Este algoritmo varreria o grafo de classes de estados da especificação e seria finalizado quando todas as classes estivessem mapeadas.

Após testes envolvendo diversas instâncias de problemas, verificou-se que este mapeamento nem sempre era possível, pois, devido à dependência mútua entre os intervalos de disparo das transições, a regra $C_i^E[t_i \rightarrow C_j^E \not\Rightarrow C_i^P[t_i \rightarrow C_j^P$ poderia se manifestar caso fossem utilizadas RdPT sem nenhuma restrição às temporizações. A RdPT ilustrada na figura 6.4 exemplifica esta situação, onde a RdPT-P é representada pelos lugares P, enquanto a RdPT-EC é representada pelos lugares R. Os intervalos estáticos de disparo estão representados próximos as suas respectivas transições, com exceção dos intervalos $[0,\infty[$, que foram omitidos na ilustração. Considera-se $T_C = \{\alpha_1, \alpha_2, \alpha_3\}$ e $T_U = \{\beta_1, \beta_2, \beta_3\}$. Os intervalos estáticos de disparo das transições não-controláveis possuem valores bem definidos, o que representa fisicamente uma duração mínima e uma duração máxima de execução de uma atividade. Os intervalos estáticos de disparo das transições controláveis possuem o valor mais amplo possível, indicando que não há restrições temporais quanto às atuações na planta. Embora representado por uma RdPT com apenas 10 lugares, os grafos de classes de estados da especificação e da planta possuem, respectivamente, 1085 e 31040 classes [6]. Uma situação na qual o mapeamento falha ocorre após o disparo da seqüência $\sigma = \alpha_1\beta_1\alpha_1\alpha_2\beta_1\alpha_1\beta_2\alpha_2\alpha_3\beta_1\alpha_1\beta_2\alpha_2$ a partir da classe inicial. Na classe resultante do grafo de classes de estados da especificação, as transições β_1 e β_3 estão habilitadas, enquanto na respectiva classe do grafo da planta, as transições habilitadas são α_1, β_1 e β_2. Assim sendo, a classe de entrada da transição β_3 não pode ser mapeada em nenhuma classe da planta.

A utilização de RdPT sem nenhuma restrição aos intervalos estáticos de disparo das transições também possui outra incoveniência durante o processo de síntese de supervisores. Como os intervalos de disparo presentes nos domínios das classes são mais abrangentes do que os intervalos mostrados no grafo de classes de estados, a eliminação de uma transição de saída de uma classe pode acarretar em uma profunda transformação no grafo, pois os limites superiores das outras transições habilitadas na mesma classe podem ser ampliados e/ou outras transições que anteriormente era inibidas temporalmente podem ficar habilitadas. Além disso, estas transformações seriam naturalmente propagadas para as demais classes, o que resultaria em um grafo de classes de estados totalmente diferente do original. Tomando-se como exemplo o grafo de classes de estados da figura 6.3, a eliminação da transição t_5 na classe C_2 resultaria na ampliação do limite superior do intervalo de disparo da transição t_2 para o valor 2. Mantendo-se a linha de raciocínio utilizada até o presente momento, estas características observadas impossibilitam a utilização de quaisquer valores para os intervalos estáticos de disparo das transições.

[6] Uma RdP isomórfica a esta RdPT possui grafos de alcançabilidade com respectivamente 8 e 32 estados. Esta comparação ilustra como a adição de aspectos temporais acarreta em explosão de estados.

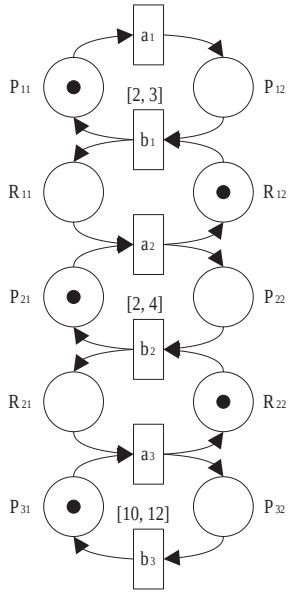

Figura 6.4: Exemplo de RdPT na qual o mapeamento falha

No entanto, caso o limite superior de todas os intervalos estáticos de disparo fosse infinito, as características recém mencionadas não ocorrem, pois, de acordo com os quatro passos da determinação do domínio de uma classe, todos os intervalos de disparo das transições presentes nos grafos de classes de estados da planta também possuirão limites superiores infinitos. Desta forma, a regra $C_i^E[t_i \rightarrow C_j^E \Rightarrow C_i^P[t_i \rightarrow C_j^P$ será sempre verdadeira, pois o modelo da planta assumirá um comportamento mais permissivo dentro das restrições impostas pelos aspectos temporais. Além disso, a eliminação de uma transição de saída de uma classe não causará transformações nas outras transições do grafo de classes de estados, justamente pelo fato dos limites superiores dos intervalos de disparo das transições possuírem o valor ilimitado. Embora tal restrição diminua a classe de problemas tratados, ela permitiria a utilização de uma metodologia bem próxima àquela utilizada para sistemas modelados por RdP, caso não fosse observado outro importante obstáculo envolvendo a modelagem de SEDs por RdPT. A segunda etapa de um possível algoritmo para a síntese de supervisores consiste na modelagem de cada especificação de coordenação por uma RdPT, apenas considerando aspectos lógicos, ou seja, sem alterar os intervalos estáticos de disparo presentes na RdPT-P. No entanto, é facilmente verificado que esta metodologia acarreta desnecessariamente em um comportamento menos permissivo, o que pode ser comprovado pelo simples exemplo ilustrado pela figura 6.5. A RdPT-P é representada pelos lugares P e corresponde a três processos não cíclicos que possuem tempos de inicialização distintos. A RdPT-EC é representada pelos lugares E e corresponde à requisição de que as transições sejam obrigatoriamente disparadas na seqüencia $\sigma = \alpha_1\beta_1\alpha_2$. Assume-se também que $T_c = \{\alpha_1, \alpha_2\}$ e $T_u = \{\beta_1\}$.

A modelagem presente na figura 6.5 é errônea, pois restringe desnecessariamente o intervalo de disparo da transição não-controlável β_1 após o disparo da transição α_1 no comportamento desejado em malha fechada. Embora os intervalos estáticos das transições presentes na RdPT-P não foram alterados pela RdPT-EC, a presença do lugar $E2$ proporciona esta restrição temporal desnecessária, justamente por recém habilitar a transição β_1 após o disparo da transição α_1.

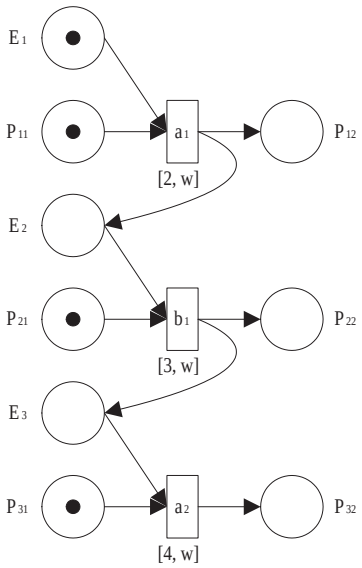

Figura 6.5: Exemplo de RdPT na qual o comportamento é mais restritivo do que o necessário

6.2.1 Possível algoritmo

Embora a veracidade da metodologia proposta para a síntese de supervisores para SEDs modelados por RdPT não foi completamente comprovada, existe boas perspectivas acerca do algoritmo proposto a seguir. Buscou-se uma abordagem próxima àquela utilizada para SEDs modelados por RdP, mas que ao mesmo tempo não apresentasse as características indesejadas mencionadas anteriormente. A metodologia se baseia nos seguintes passos:

1. Modele cada componente do SED por uma RdPT. O conjunto destas redes é denominado RdPT da planta (RdPT-P), pois constitui o comportamento da planta em malha aberta. Todas as transições t_i devem possuir intervalos estáticos de disparo $[\theta^E_{min}(i), \infty]$, onde $\theta^E_{min}(i) \geq 0$.

2. Modele cada especificação de coordenação por uma RdPT, apenas considerando aspectos lógicos, ou seja, como todos os intervalos estáticos de disparo $[0, \infty]$. A composição do conjunto de todas as RdPT de especificação de coordenação (RdPT-EC) com a estrutura da RdPT-P desconsiderando os intervalos estáticos de disparo das transições constitui o comportamento lógico desejado em malha fechada.

3. Gere o grafo de classe de estados do comportamento da planta em malha aberta e o grafo de alcançabilidade do comportamento lógico desejado em malha fechada, respectivamente G^P_{CE} e G^{EC}_A. Gere o grafo de classes de estados da especificação G^E_{CE}, através do produto síncrono de G^P_{CE} e G^{EC}_A, mantendo os valores dos intervalos dinâmicos de disparo de G^P_{CE}.

4. Obtenha a componente *trim* de G^E_{CE} e mapeie suas classes em classes de G^P_{CE}.

5. Pela abordagem Ramadge and Wonham [82], identifique as más classes do grafo resultante da especificação. Elimine aquelas que não possuem transições controláveis de saída e carimbe [7] as

[7] Foi utilizada a terminologia carimbar, para não se confundir com a marcação de estados, ou seja, a representação de

demais. Verifique a possibilidade de restringir o limite superior de uma transição controlável de saída de todas as más classes carimbadas, de forma a impossibilitar temporalmente e localmente nas classes equivalentes em G^P_{CE} o disparo das transições não-controláveis que fizeram com que tais classes fossem classificadas como más classes. Caso esta ação não seja possível, elimine as respectivas classes carimbadas. Por último, obedeça a uma das regras abaixo:

(a) Caso nenhuma classe tenha sido eliminada nesta etapa e não haja más classes carimbadas, o algoritmo pode ser encerrado.

(b) Caso nenhuma classe tenha sido eliminada nesta etapa e o passo anterior tenha sido o passo 6, vá para o passo 7.

(c) Caso contrário, vá para o passo 6.

6. Obtenha a componente *trim* do grafo de classes de estados resultante, sem alterar o carimbo das más classes remanescentes. Volte para o passo 5.

7. Restrinja o limite superior de uma transição controlável de saída de todas as más classes carimbadas, de forma a impossibilitar temporalmente e localmente em G^P_{CE} o disparo das transições não-controláveis que fizeram com que tais classes fossem classificadas como más classes. Ajuste o grafo da especificação resultante de acordo com as regras abaixo:

   Sejam $[\theta_{min}(u),\infty]$, $[\theta_{min}(c),\infty]$, $[\theta_{min}(h_i),\infty]$ e $[\theta_{min}(p_j),\infty]$ os intervalos de disparo da transição não-controlável cujo disparo se queira inibir [8], da transição controlável a ser manipulada, das transições habilitadas na classe envolvida e das transições que permanecem habilitadas após o disparo da transição controlável, respectivamente.

   (a) Ajuste o intervalo de disparo da transição controlável a ser manipulada para $[\theta_{min}(c),\theta_{min}(u))$.

   (b) Ajuste o intervalo de disparo de cada transição habilitada na classe envolvida para $[\theta_{min}(h_i),\theta_{min}(u))$. Caso $\theta_{min}(u) \leq \theta_{min}(h_i)$, elimine a respectiva transição do grafo de classes de estados resultante.

   (c) Ajuste o intervalo de disparo de cada transição que permaneceu habilitada após o disparo da transição controlável para $[max(0,\theta_{min}(h_i)-\theta_{min}(u)),\theta_{min}(u)-\theta_{min}(c))$. Caso $max(0,\theta_{min}(h_i)-\theta_{min}(u)) \leq (\theta_{min}(u)-\theta_{min}(c))$, elimine a respectiva transição do grafo de classes de estados resultante.

   (d) Repita iterativamente os sub-passos (a), (b) e (c) para todos os estados alvo das transições que possuíram seus intervalos de disparo alterados. Este algoritmo tende a convergir.

   (e) Volte para o passo 5.

Os dois primeiros passos deste algoritmo possuem os mesmos princípios da metodologia proposta para a resolução do PCS para SEDs modelados por RdP. A única diferença é que aspectos temporais são levados em consideração durante a modelagem do comportamento da planta em malha aberta,

tarefas completas.

[8] Ao invés de considerar todas as transições não controláveis cujo disparo se queira inibir, basta levar em consideração aquela que possui o intervalo de disparo com menor limite inferior

pois os limites inferiores dos intervalos estáticos de disparo das transições podem assumir valores não nulos. Além disso, a operação de desconsiderar os intervalos estáticos de disparo das transições durante o passo 2 equivale a considerá-los como se fossem $[0,\infty]$ [9]. Assim como na metodologia anterior, a modelagem por componentes favorece extensões futuras, tais como Controle Modular [95] e Controle Modular Local [80]. A modelagem da cada especificação de coordenação, por sua vez, é feita sobre a estrutura de RdTP-P sem considerar os intervalos estáticos de disparo das transições, justamente para não restringir desnecessariamente o comportamento desejado em malha fechada. Ambos os passos são realizados com auxílio da ferramenta Tina.

O passo 3 é conseqüência da necessidade da comparação entre os grafos de classes de estados da planta e da especificação, respectivamente G^P_{CE} e G^E_{CE}. O grafo de classes de estados do comportamento da planta em malha aberta é gerado diretamente a partir da RdPT-P, enquanto o grafo de classes de estados do comportamento lógico desejado em malha fechada é calculado em duas etapas. Tanto os grafos de alcançabilidade, quanto os grafos de classes de estados são gerados pela ferramenta Tina. Ao contrário da metodologia proposta a síntese de supervisores para SEDs modelados por RdP, a ferramenta MEC não pode ser utilizada em etapas posteriores, pois não considera aspectos temporais. Optou-se, portanto, por exportar os grafos utilizando-se os formatos *verbose* e autômato do Tina.

O passo 4 é dividido em duas etapas distintas. A primeira parte consiste na determinação da componente simultâneamente acessível e co-acessível do grafo de classes de estado da especificação. Devido às características da RdPT que lhe deu origem, a eliminação das transições não acarreta em alterações na estrutura remanescente do grafo. Esta etapa pode ser realizada por uma ferramenta de manipulação de autômatos [10], pois não depende de aspectos temporais. A segunda parte consiste no mapeamento entre as classes de G^E_{CE} e G^P_{CE} que obedecem a regra $C^E_i[t_i \rightarrow C^E_j \Rightarrow C^P_i[t_i \rightarrow C^P_j$. Prova-se que esta regra é sempre válida, pois o grafo de classes de estados da especificação é gerado a partir do produto síncrono envolvendo o grafo de classes de estados da planta. Esta etapa pode ser facilmente implementada por um algoritmo desenvolvido em qualquer linguagem de programação. É desejável que a componente trim de G^E_{CE} seja calculada antes da realização do mapeamento entre as classes, pois esta ordem reduz a complexidade do algoritmo, mesmo que em pequenas proporções.

Por fim, os passos 5, 6 e 7 representam uma tentativa de determinação da máxima linguagem controlável para o PCS onde os SEDs são modelados por RdTP. Embora sua veracidade ainda não foi completamente comprovada, estas etapas correspondem à melhor visão do que venha a ser uma solução deste problema até o momento presente. Basicamente, é utilizada uma generalização da abordagem Ramadge and Wonham [82], ao incluir dois novos aspectos na metodologia. O primeiro corresponde à eliminação das más classes. Uma classe é definitivamente considerada má classe caso não possua nenhuma transição controlável de saída que possa proibir, através da restrição de seu intervalo de disparo, o disparo na classe equivalente em G^P_{CE} das transições não-controláveis que a tornaram má classe. Estas classes podem ser eliminadas sem a preocupação de propagação de efeitos no comportamento do grafo de classes de estados. Isto se justifica pelo fato das transições de entrada

[9] Observe que uma RdPT $N_T =< N, IED >$ onde todos os intervalos estáticos de disparo são $[0,\infty]$ é equivalente a RdP N

[10] Neste trabalho utilizaram-se alguns filtros do Grail

e de saída destas classes possuírem o limite superior de seus intervalos de disparo infinito. Como estes intervalos de disparo não restringem temporalmente o disparo de nenhuma outra transição presente no domínio das más classes, não há alterações nos intervalos de disparo das transições remanescentes, nem surgimento de novas transições no grafo de classes de estados. Como o grafo de classes de estados é por construção mais complexo do que um grafo de alcançabilidade, tenta-se postergar ao máximo a eliminação das más classes, visando a obtenção de um supervisor minimamente restritivo. A restrição de um intervalo de disparo de uma transição controlável deve ser local, ou seja, os intervalos de disparo desta transição devem permanecer inalterados nas outras regiões do grafo de classes de estados. Esta restrição consiste em tornar o limite superior do intervalo de disparo de uma transição controlável infinitesimamente [11] inferior aos limites inferiores de todos os intervalos de disparo das transições não-controláveis que se queira proibir o disparo. Quando esta anulação não é possível, a classe envolvida deverá ser eliminada do grafo. É importante ressaltar que a implementação do supervisor deve ser capaz de distinguir em qual momento os intervalos de disparo das transições controláveis devem ser momentaneamente alterados.

O segundo aspecto é que a identificação das más classes não considera somente a inibição de transições não-controláveis que normalmente ocorreriam no sistema em malha aberta. Classes que restringem o intervalo de disparo de uma transição não-controlável também devem ser consideradas más classes, pois estas classes inibem tais transições durante uma faixa finita do intervalo de disparo. Este tipo de má classe só ocorre nas iterações nas quais o passo 7 tenha acontecido pelo menos uma vez. Isto se justifica, pois quando o limite superior do intervalo de disparo de uma transição controlável é propositalmente diminuído, esta alteração deve ser propagada para todas as transições de saída da respectiva classe. Além disso, esta restrição também pode acarretar em aumentos nos limites inferiores dos intervalos de disparo das transições que permanecem habilitadas nas classes imediatamente sucessoras. Há indícios que este algoritmo convirja para a solução ótima, mas ainda não foram encontrados exemplos de dimensões reduzidas e que apresentassem todas as peculiaridades envolvidas neste procedimento.

Por último, apresenta-se uma versão alternativa para os passos 2 e 3 do algoritmo anterior. Os passos expostos a seguir partem do princípio que as restrições de coordenação são modeladas diretamente por autômatos e possuem a vantagem de serem mais diretos. Em contrapartida, se distanciam mais ainda da metodologia proposta para o caso não-temporal:

- Modele cada especificação de coordenação por um autômato, apenas considerando aspectos lógicos.
- Gere o grafo de classes de estados do comportamento da planta em malha aberta e o produto síncrono dos autômatos das especificações de coordenação, respectivamente G^P_{CE} e A_{EC}. Gere o grafo de classes de estados da especificação G^E_{CE}, através do produto síncrono de G^P_{CE} e A_{EC}, mantendo os valores dos intervalos dinâmicos de disparo de G^P_{CE}.

[11] A implementação de valores infinitesimamente inferiores (ou superiores) pode ser obtida através da combinação de intervalos abertos e fechados para a representação dos intervalos de disparo das transições

6.2.2 Exemplo

Considera-se a linha de transferência ilustrada na figura 6.6. Este sistema é composto por duas máquinas (M_1 e M_2) e um *buffer* intermediário de capacidade unitária (B_1). Para i = 1,2, consideram-se α_i como o início de operação da máquina M_i com a retirada de uma peça de B_{i-1} (quando existir), e β_i como o fim de operação da máquina M_i com o depósito de uma peça em B_i (quando existir). Sabe-se que $\Sigma_c = \{\alpha_1, \alpha_2\}$ e $\Sigma_u = \{\beta_1, \beta_2\}$. Do conhecimento prévio da planta, sabe-se também que o processamento de uma peça pela máquina M_1 gasta pelo menos 4 unidades de tempo, enquanto o processamento pela máquina M_2 gasta pelo menos 2 unidades de tempo.

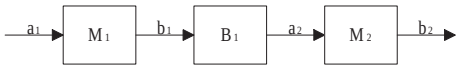

Figura 6.6: Linha de transferência com duas máquinas

Os comportamentos das máquinas M_1 e M_2 podem ser modelados pela RdPT-P apresentada na figura 6.7, que representa o comportamento da planta em malha aberta. A transição β_1 possui um intervalo estático de disparo igual a $[2, \infty]$, o que representa o tempo de processamento de uma peça pela máquina M_1. Observação análoga pode ser feita com a transição β_2. As transições controláveis, por sua vez, possuem intervalos $[0, \infty]$, pois podem ser acionadas a qualquer momento. Os lugares marcados são P_{11} e P_{21}, enquanto as demais características da rede seguem a metodologia proposta para RdP.

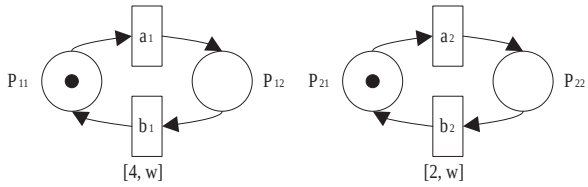

Figura 6.7: RdPT-P da linha de transferência com duas máquinas

Como restrição de controle, deseja-se que não ocorra nem *underflow*, nem *overflow* no *buffer*, o que é satisfeito alternando-se os eventos β_1 e α_2 (sem alterar os intervalos estáticos de disparo destas transições na RdPT-P). Esta restrição pode ser modelada pela RdPT-EC apresentada na figura 6.8. Os dois lugares desta RdPT são considerados marcados.

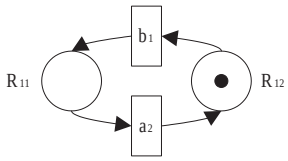

Figura 6.8: RdPT-EC da linha de transferência com duas máquinas

O grafo de classes de estados do comportamento da planta em malha aberta G^P_{CE} e o grafo de alcançabilidade do comportamento lógico desejado em malha fechada G^{EC}_A são ilustrados respectivamente pelas figuras 6.9 e 6.10. No primeiro, somente a classe C_0 é marcada, enquanto no último, os estados marcados são $\overline{E}_0$ e $\overline{E}_2$. Com relação aos intervalos temporais, optou-se por não representar

os domínios das classes em G^P_{CE} e, portanto, somente estão representados os intervalos de disparo das transições. Observa-se também que não há limites superiores para os disparos das transições no grafo da figura 6.9, o que é conseqüência dos valores assumidos para os limites superiores dos intervalos estáticos de disparo das transições da RdPT-P.

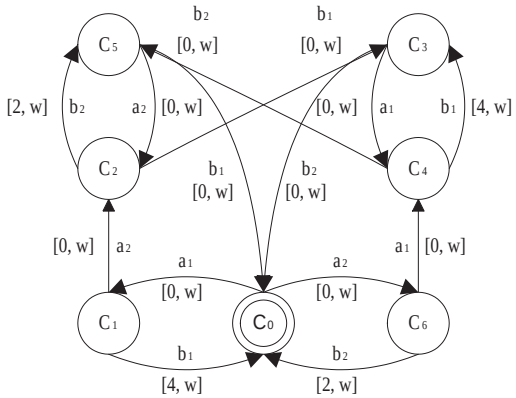

Figura 6.9: Grafo de classes de estados da RdPT-P da linha de transferência com duas máquinas

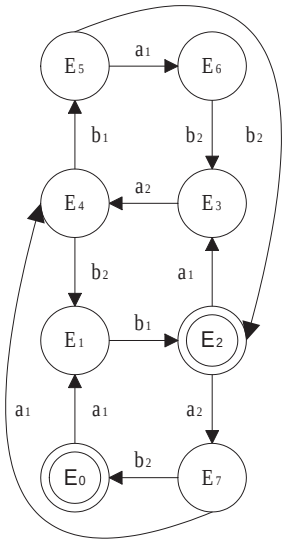

Figura 6.10: Grafo de alcançabilidade da RdPT-EC da linha de transferência com duas máquinas

O grafo de classes de estados da especificação G^E_{CE} está ilustrado na figura 6.11. Este grafo é *trim* e suas classes marcadas são $\overline{C_0}$ e $\overline{C_2}$. O mapeamento entre as classes de G^E_{CE} e as classes de G^P_{CE} é indicado pelos nomes presentes na metade inferior de cada classe em G^E_{CE}.

Varrendo simultaneamente os grafos G^E_{CE} e G^P_{CE}, observa-se na primeira iteração que as classes $\overline{C}_3$, $\overline{C}_9$ e $\overline{C}_{10}$ inibem a transição não controlável β_1 e, portanto, são consideradas más classes. Como a classe $\overline{C}_9$ não possui transições controláveis de saída, esta classe deve ser eliminada. A classe $\overline{C}_{10}$ também deve ser eliminada, pois não é possível reduzir o intervalo dinâmico de α_2, sua única transição controlável de saída, para evitar a ocorrência de β_1 na classe equivalente em G^P_{CE}. Assim sendo, somente a classe $\overline{C}_3$ é carimbada. Na próxima iteração, o grafo resultante permanece *trim*, ao mesmo tempo que não há mais necessidade de eliminar nenhuma classe em G^E_{CE}. O intervalo de

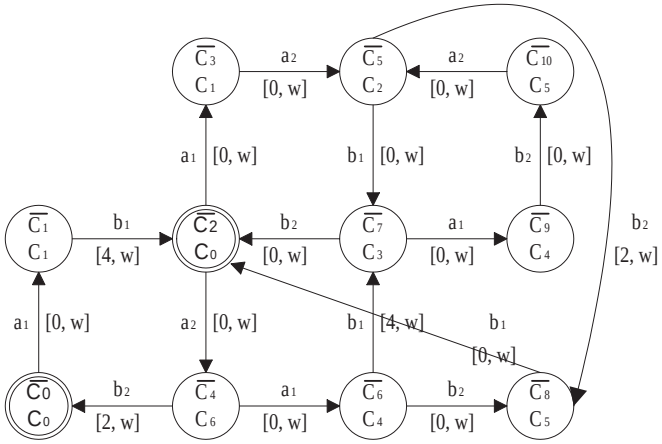

Figura 6.11: Grafo de classes de estado da RdPT-E da linha de transferência com duas máquinas

disparo de transição de saída α_2 da classe $\overline{C}_3$ deve ser reduzido à faixa de valores $[0,4)$, de forma com que o disparo de β_1 na classe equivalente em G^P_{CE} seja inibido temporalmente e localmente. O grafo de classes de estados do supervisor obtido é ilustrado na figura 6.12

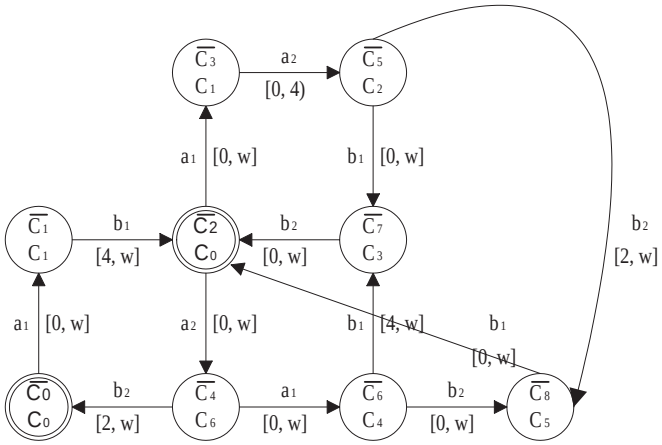

Figura 6.12: Grafo de classes de estado do supervisor da linha de transferência com duas máquinas

6.3 Discussão

A resolução do PCS envolvendo RdPT (ou qualquer outra extensão temporal para as RdP) é, certamente, um vasto campo de pesquisa. Neste capítulo foram dados algumas contribuições iniciais para uma metodologia semelhante àquela proposta no Capítulo 4. No entanto, ainda resta muito a ser feito para que se chegue a um resultado tão expressivo quanto o anterior.

A leitura aprofunda de alguns trabalhos poderão auxiliar na tarefa de prosseguimento desta linha de raciocínio. Por um lado, pode-se buscar compreender outros grafos que representem o comportamento dinâmico de uma RdPT [9] [13]. Por outro lado, pode-se buscar equivalências e analogias entre as perspectivas discutidas até o momento com os resultados apresentados por Sava and Alla

[85], Cassez and Roux [21] ou até mesmo Brandim and Wonham [17]. Ademais, pode-se utilizar outras estruturas para modelar os SEDs, tais como a rede de Petri temporizada [84].

Capítulo 7

Conclusões

Esta dissertação contribui para a teoria de controle de sistemas a eventos discretos com a proposta de uma abordagem que considera aspectos de verificação formal e sistemas modelados por redes de Petri etiquetadas. Por ser baseada na abordagem Ramadge e Wonham, o presente trabalho também herda as principais características da síntese de supervisores desenvolvida por estes autores.

Inicialmente, descreve-se todos os formalismos necessários ao entendimento deste trabalho. Em seguida, são enunciados diversos resultados acerca da síntese de supervisores utilizando sistemas modelados por redes de Petri. Duas recentes abordagens são descritas com mais detalhes, pois constituem os pilares deste trabalho. Por um lado, a abordagem de Ghaffari et al. utiliza a Teoria das Regiões para a determinação de lugares de controle que podem ser adicionados a um modelo em rede de Petri, de forma a satisfazer de forma minimamente restritiva um comportamento desejado. Por outro lado, a abordagem de Ziller e Schneider utiliza equações em μ-calculus para realizar a síntese de supervisores menos restritivos sobre sistemas modelados por autômatos. Como ponto em comum, as duas metodologias se baseiam, em maior ou menor grau, na abordagem de Ramadge e Wonham.

A abordagem de Ramadge e Wonham é baseada em linguagens controláveis e permite a síntese automática de supervisores. O procedimento de síntese é baseado no modelo da dinâmica do sistema em malha aberta e na especificação do comportamento desejado. Quando o sistema é alterado ou os objetivos de controle são modificados, novos controladores podem ser facilmente e automaticamente projetados. Além de ser mais conveniente do que os procedimentos manuais, esta teoria introduz o conceito de supervisor minimamente restritivo, ou seja, aquele que atribui maior grau de liberdade ao sistema controlado. Na abordagem proposta, os principais conceitos das abordagens de Ghaffari et al. e de Ziller e Schneider são mesclados, sem perda dos princípios da abordagem de Ramadge e Wonham.

Durante a modelagem, tanto o sistema quanto as especificações de controle são modelados por redes de Petri. Ao contrário da abordagem de Ghaffari et al, os diversos componentes do sistema são modelados individualmente. Embora parte do poder de expressão em redes de Petri é perdido, ganha-se em aspectos como flexibilidade e generalidade. A rede de Petri equivalente ao comportamento desejado em malha fechada é então obtida pela fusão de transições entre as redes de petri dos diversos

componentes da planta com redes das especificações de coordenação. Posteriormente, a determinação de um comportamento minimamente restritivo é realizada utilizando-se uma adaptação das equações em μ-calculus propostas por Ziller e Schneider sobre o grafo de alcançabilidade da rede resultante. Por último, a Teoria das Regiões é utilizada da mesma forma como foi proposta por Ghaffari el al. Prova-se facilmente que a ordem de complexidade da metodologia proposta é no máximo semelhante à obtida por Ramadge e Wonham, o que a torna tão viável quanto a abordagem clássica.

A incorporação de sistemas modelados como redes de Petri temporais também é discutida, sem, no entanto, chegar a um resultado tão expressivo quanto o caso não-temporal. Contudo, a análise realizada indica algumas inconveniências ao utilizar uma metodologia mais próxima possível à anterior. Como sugestões para trabalhos futuros, esta seria a área com perspectivas mais ricas.

Outro interessante campo de pesquisa seria a incorporação de algumas extensões à metodologia proposta. Algumas sugestões iniciais seriam o Controle Modular, o Controle Modular Local e o Controle Multitarefa. As duas primeiras extensões possuiriam poderiam contribuir para o aumento da flexibilidade e da eficiência computacional, enquanto a última poderia resultar em uma maior generalização dos problemas tratados. No último caso, o Problema de Controle Supervisório Generalizado definido por Ziller e Schneider representa um fundamental ponto de partida.

Outras indicações para trabalhos futuros constituem em melhoria de aspectos pontuais da metodologia proposta. Uma sugestão seria a utilização de sistemas modelados por redes de petri não-binárias e com pesos nos arcos. A priori, imagina-se que os maiores esforços seriam no sentido de criar filtros de manipulação dos grafos de alcançabilidade destas redes. Outro campo de pesquisa seria a implementação de um programa otimizado para resolução dos diversos modelos em programação linear inteira. Como já discutido, a melhor solução neste sentido seria a chamada de rotinas de algum otimizador diretamente a partir do programa que gera os modelos. Uma última sugestão seria a implementação da redução automatizada dos lugares de controle a serem adicionados à rede de Petri, de forma com que o efeito desta supervisão atenda o comportamento minimamente restritivo.

Referências Bibliográficas

[1] Ampl - modeling language for mathematical programming. http://www.ampl.com/, 2006.

[2] Ilog cplex. http://www.ilog.com/products/cplex/, 2006.

[3] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126: 183–235, 1994.

[4] A. Arnold, D. Bégay, and P. Crubillé. *Contruction and Analysis of Transition Systems with MEC*. World Scientific, 1994.

[5] B. Aspvall and Y. Shiloach. A polynomial time algorithm for solving systems os linear inequalities with two variables per inequality. *20th Annual Symp. on Foudations of Computer Sciences*, pages 205–217, 1979.

[6] E. Badouel, L. Bernardinello, and P. Darondeau. Polynomial algorithms for the synthesis of bounded nets. *Proc. CAAP 95*, 915:364–378, 1995.

[7] C. Berge. *Graphs and Hypergraphs*. Elsevier, 1976.

[8] J.L. Bernartt, B. Berthomieu, J.P. Bodeveix, S. Devulder, J.M. Farines, M. Falali, P. Gaufillet, J.L. Lambert, P. Michel, A. Naspolini, G. Padiou, P.O. Ribet, and F. Vernadat. *Cotre: Choix des techniques et outils de Vérification*. Airbus-France, TNI-Valiosys, ENST. Bretagne, IRIT, LAAS et ONERA-DTIM, 2004.

[9] B. Berthomieu. La méthode des classes d'états pour l'analyse des réseaux temporels - mise en oeuvre, extension à la multi-sensibilisation. *Modélisation des Systèmes Réactifs*, 2001.

[10] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, 17(3), 1991.

[11] B. Berthomieu and M. Menasche. An enumerative approach for analyzing time petri nets. *Proc. IFIP Cong. 1983*, 1983.

[12] B. Berthomieu, P.O. Ribet, and F. Vernadat. *L'outil TINA: Construction d'espaces d'états abrastraits pour les réseaux de Petri et réseaux temporels*. LAAS-CNRS, 2003.

[13] B. Berthomieu and F. Vernadat. State class constructions for branching analysis of time petri nets. *TACAS 2003*, 2003.

[14] G. Bhat and R. Cleaveland. *Efficient Model Checking via the Equational μ-calculus*, 1995.

[15] O.R. Boissel. Optimal feedback control for discrete-event process systems using simulated annealing. Master's thesis, University of Notre Dame, 1993.

[16] O.R. Boissel and J.C. Kantor. Optimal feedback control design for discrete-event systems using simulated annealing. *Computers and Chemical Engineering*, 19:253–266, 1995.

[17] B.A. Brandim and W.M. Wonham. *Supervisory Control of Timed Discrete-Event Systems*, 1992.

[18] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. McKenzie. *Systems and Software Verification: Model-Checking Techiniques and Tools*. Spinger, 2001.

[19] J. Cardoso and R. Valette. *Redes de Petri*. Editora da UFSC, 1997.

[20] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[21] F. Cassez and O.H. Roux. *From Time Petri Nets to Timed Automata*. IRCCyN/CNRS, 2003.

[22] E.M. Clarke and E.A. Emerson. Syhnthesis of synchronisation skeletons for braching time logic. *Logics of Programs*, LNCS 131:52–71, 1981.

[23] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[24] E.M. Clarke, O. Grumberg, H. Hirashi, S. Jha, D.E. Long, K.L. McMillan, and L.A. Ness. Verification of the futurebus+ cache coherence protocol. *Proceedings 11th Int. Symp. on Computer Hardware Description Languages and Their Apllications*, 1993.

[25] R. Cleaveland, M. Klein, and B. Steffen. Faster model checking for the modal μ-calculus. *Computer Aided Verification (CAV 92)*, 663:410–422, 1992.

[26] E. Costa and A. Lima. Utilizando redes de petri multiplexadas na síntese de supervisores de sistemas a eventos discretos. *Anais do IV Simpósio Brasileiro de Automação Inteligente - SBAI*, pages 533–538, 1999.

[27] E.M.M. Costa and A.M.N. Lima. Utilizando lógica temporal e rede de petri na síntese de supervisores de sistemas a eventos discretos. *Não sei*, 2001.

[28] J.E.R. Cury. *Teoria de Controle Supervisório de Sistemas a Eventos Discretos*. Universidade Federal de Santa Catarina, 2001.

[29] R. De Nicola and F. Vaandrager. Action versus state based logics for transition systems. *Lecture Notes in Computer Science*, (469):407–419, 1990.

[30] R.C. Dorf and R.H. Bishop. *Modern Control Systems*. Addison Wesley, 1998.

[31] E. Emerson and C.L. Lei. Efficient model checking in fragments of the proposisitional μ-calculus. *IEEE Symposium on Logic in Computer Science (LICS)*, pages 267–278, 1986.

[32] E.A. Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *14th ACM Symp. Theory of Computing (STOC 82)*, pages 169–180, 1982.

[33] J. Ezpeleta, J.M. Colom, and J. Martinez. A pn based deadlock prevention policy for flexible manufacturing systems. *IEEE Trans. Robot. Automat*, 11(2):173–184, 1995.

[34] G. Frey. Automatic implementation of petri net based control algorithms on plc. *American Control Conf.*, pages 2819–2823, 2000.

[35] A. Ghaffari, N. Rezg, and X. Xie. Conception du superviseur optimal vivant à l'aide de la théorie des régions. *MRS 2001*, 2001.

[36] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive petri net controller theory of regions. *IEEE Transactions on Robotics and Automation*, 19(1), 2003.

[37] C.C. Gotlieb and D.G. Corneil. Algorithnms for finding a fundamental set of clycles for an undirected linear graph. *Communications of the ACM*, 10(12):780–783, 1967.

[38] T. Gu, J. Gao, and C. Zhou. Petri-net-based coordinate control of discrete events in batch processes. *3rd APIS/IMAC*, pages 135–137, 1996.

[39] A. Guia. Petri net techniques for supervisory control of discrete event systems. *1th Int. Work. on Manufacturing and Petri Nets*, pages 1–30, 1996.

[40] A. Guia and F. DiCesare. Blocking and controllability of petri nets in supervisory control. *IEEE Trans. Autom. Control*, 39:818–823, 1994.

[41] A. Guia, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. *IEEE International Conference on Systems, Man, and Cybernetics*, pages 974–979, 1992.

[42] A. Guia, F. DiCesare, and M. Silva. Petri net supervisors for generalized mutual exclusion constraints. *12th IFAC World Congress*, pages 267–270, 1993.

[43] L. E. Holloway, L.E. Guan, and L. Zhang. A generalization of state avoidance politicies for controlled petri nets. *IEEE Trans.on Automatic Control*, 41(6):804–816, 1996.

[44] L.E. Holloway and F. Hossain. Feedback control for sequencing specifications in controlled petri nets. *3rd Int. Conf. on Computer Integrated Manufacturing*, pages 242–250, 1992.

[45] L.E. Holloway and B.H. Krogh. Synthesis of feedback logic for a class of controlled petri nets. *IEEE Trans. on Automatic Control*, 35(5):514–523, 1990.

[46] L.E. Holloway and B.H. Krogh. Synthesis of feedback control logic for discrete manufacturing systems. *Automatica*, 27(4):641–651, 1991.

[47] L.E. Holloway and B.H. Krogh. On closed-loop liveness of discrete event systems under maximally permissive control. *IEEE Trans. on Automatic Control*, 37(5):692–697, 1992.

[48] L.E. Holloway, B.H. Krogh, and A. Guia. A survey pf petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 7:151–190, 1997.

[49] G.J. Holzmann. The theory and practice of a formal method: Newcore. *Proceedings 13th IFIP World Congress*, pages 35–44, 1994.

[50] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, USA, 1979.

[51] A. Ichikawa and K. Hiraishi. Analysis and control of discrete event systems represented by petri nets. *Discrete Event Systems: models and applications*, 103:115–134, 1988.

[52] M.V. Iordache and P.J. Antsaklis. *Synthesis of Supervisors Enforcing General Linear Constraints in Petri Nets*, 2002.

[53] M.V. Iordache, J.O. Moody, and P.J. Antsaklis. Synthesis of deadlock prevention supervisors using petri nets. *IEEE Transactions on Roboticas and Automation*, 18(1):59–68, 2002.

[54] M. Jantzen. Language theory of petri nets. *Lectures Notes in Computer Sciences*, 254(1): 397–412, 1987.

[55] J. P. Katoen. *Concepts, Algorithms, and Tool for Model Checking*, 1999. Lecture Notes of The Course.

[56] B.H. Krough. Conttroled petri nets and maximally permissive feedback logics. *25th Annual Allerton Conference*, pages 317–326, 1987.

[57] R. Kumar and V. Garg. *Modeling and Control of Logical Discrete Systems*. Kluvwer Academic Publishers, 1995.

[58] R. Kumar and L.E. Holloway. Supervisory control of deterministic petri net languages with regular specifications languages. *IEEE Trans. on Automatic Control*, 41(2):245–249, 1996.

[59] J.L. Lassez, V.L. Nguyen, and E.A. Sonenberg. Fixed point theorems ans semantics: a folk tale. *Information Processing Letters*, 14(3):112–116, 1982.

[60] R.W. Lewis. *Programming Industrial Control Systems Using IEC 1131-3*. IEE Press, revised edition edition, 1998.

[61] Y. Li and W. M. Wonham. Control of vector discrete-event systems. i- the base model. *IEEE Trans. on Automatic Control.*, 38:1214–1227, 1993.

[62] Y. Li and W. M. Wonham. Control of vector discrete-event systems. ii- controller synthesis. *IEEE Trans. on Automatic Control.*, 39:512–530, 1994.

[63] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. *12th ACM Symp. Principles of Programming Languages (POPL 85)*, pages 97–107, 1985.

[64] E.J. Lima and C.E.T. Dórea. Synthesis and plc implementation of supervisory control via place invariants for a manufacturing cell. *IFAC 2004*, 2004.

[65] P. Merlin. *A study of the recoverability of computer systems*. PhD thesis, University of California, 1974.

[66] J.O. Moody and P.J. Antsaklis. Deadlock avoidance in petri nets with uncontrollable transitions. *American Control Conference*, pages 24–26, 1998.

[67] J.O. Moody and P.J. Antsaklis. *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers, 1998.

[68] J.O. Moody, P.J. Antsaklis, and M.D. Lemmon. Automated design of a petri net feedback controller for a robotic assembly cell. *INRIA/IEEE Sym. on Emerging TEchnologies and Factory Automation*, 2:117–128, 1995.

[69] J.O. Moody and P.J. Antsanklis. Petri net supervisors for des in the presence of uncontrolable and unobservable transitions. *34rd Annual Allerton Conference*, 1995.

[70] J.O. Moody, M. Lemmon, and P.J. Antsaklis. Supervisory control of petri nets with uncontrollable/unobservable transitions. *35th IEEE CDC*, 1996.

[71] A. Morgenstern and K. Schneider. *A Unified Model Checking Framework for the Supervisor Synthesis Problem*, 2005.

[72] D. Niwinski. On fixed point clones. *Internationl Colloquium on Automata, Languages and Programming (ICALP)*, 226:464–473, 1986.

[73] K. Paton. An algorithm for finding a fundamental set of cycles of a graph. *Communications of the ACM*, 12(9), 1969.

[74] S. Peng and M. Zhou. Conversion between ladder diagrams and pns in discrete-event control design - a survey. *IEEE Int. Conf. System, Man and Cybernetics*, 4:2682–2687, 2001.

[75] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[76] C. A. Petri. *Communication with automata*. PhD thesis, Darmstadt University, 1962.

[77] A. Pnueli. The temporal logic of programs. *Proceedings 18th IEEE Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[78] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13 (1):45–60, 1981.

[79] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. *Lecture Notes in Computer Science*, 137:337–351, 1982.

[80] M.H. Queiroz. Controle supervisório modular de sistemas de grande porte. Master's thesis, Universidade Federal de Santa Catarina, 2000.

[81] M.H. Queiroz. *Controle Supervisório Modular e Multitarefa de Sistemas Compostos*. PhD thesis, Universidade Federal de Santa Catarina, 2004.

[82] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event process. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[83] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[84] C. Ranchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*, 1974.

[85] A.T. Sava and H. Alla. Commande par supervision des systèmes à événements discrets temporisés. *MRS 2001*, 2001.

[86] V. Schastai, E. A. Lima, and L. A. Künzle. Sequence analysis for time petri nets. *WOODES 2004*, 2004.

[87] K. Schneider. *Verification of Reactive Systems: Formal Methgods and Algorithms*, 2003.

[88] R.S. Sreenivas. Enforcing liveness via supervisory control in discrete event dynamic systems modeled by completely controlled petri nets. *IEEE Transactions on Automatic Control*, submetido, 1996.

[89] R.S. Sreenivas. On commoner's liveness theorem and supervisotry policies that enforce liveness in free-choice petri nets. *IEEE Transactions on Automatic Control*, submetido, 1996.

[90] R.S. Sreenivas. On supervisory policies that enforce global fairness and bounded fairness in partially controlled petri nets. *DEDS*, submetido, 1996.

[91] A. Tarski. A lattice-theoretical fixpoint theorem and its aplications. *Pacific J. Math*, 5:285–309, 1955.

[92] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *1st IEEE Symp. Logic in Computer Science (LICS 96)*, pages 332–344, 1986.

[93] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. *Lectures Notes in Computer Science*, 1043:238–266, 1996.

[94] J.T. Welch. A mechanical analysis of the cyclic structure of undirected linear graphs. *ACM*, 13 (2):205–210, 1966.

[95] W.M. Wonham. *Supervisory Control of Discrete-Event Systems*. University of Toronto, 2004.

[96] E.C. Yamalidou, E.D. Adamides, and D. Bovin. Optimal failure recovery in batch processing using petri net models. *American Control Conference 1992*, 3:1906–1910, 1992.

[97] E.C. Yamalidou and J.C. Kantor. Modeling and optimal control of discrete-event chemical processes using petri nets. *Computers and Chemical Engineering*, 15:503–519, 1991.

[98] E.C. Yamalidou, J.O. Moody, M.D. Lemmon, and P.J. Antsaklis. Feedback control of petri nets based on place invariants. *Automatica*, 32(1):15–28, 1996.

[99] R. Ziller and K. Schneider. *A Generalized Approach to Supervisor Synthesis*, 2003.

[100] R. Ziller and K. Schneider. *A μ-Calulus Approach to Supervisor Synthesis*, 2003.

[101] R. Ziller and K. Schneider. *Reducing Complexity of Supervisor Synthesis*, 2003.

[102] R. Ziller and K. Schneider. *Combining Supervisor Synthesis and Model Checking*, 2005.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo