UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Marcelo Teixeira

EXPLORANDO O USO DE DISTINGUIDORES E DE AUTÔMATOS FINITOS ESTENDIDOS NA TEORIA DO CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

Florianópolis

2013

Marcelo Teixeira

EXPLORANDO O USO DE DISTINGUIDORES E DE AUTÔMATOS FINITOS ESTENDIDOS NA TEORIA DO CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. José Eduardo Ribeiro Cury
Coorientador: Prof. Dr. Max Hering de Queiroz

Florianópolis

2013

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Marcelo Teixeira

EXPLORANDO O USO DE DISTINGUIDORES E DE AUTÔMATOS FINITOS ESTENDIDOS NA TEORIA DO CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

Esta Tese foi julgada aprovada para a obtenção do Título de "Doutor em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 23 de setembro 2013.

Prof. Chefe, Dr. Jomi Fred Hubner
Coordenador do Curso

Banca Examinadora:

Prof. Dr. José Eduardo Ribeiro Cury
Presidente

Prof. Dr. André Bittencourt Leal
UDESC

Prof. Dr. Antonio Eduardo Carrilho da Cunha
IME

Prof. Dr. Jean-Marie Farines
UFSC

Prof. Dr. Leandro Buss Becker
UFSC

Prof. Dra. Patrícia Nascimento Pena
UFMG

À minha esposa, Lovania, pela sintonia perfeita...

AGRADECIMENTOS

Agradeço a Deus, porque dele recebi a vida e, assim, posso lutar pelos meus objetivos.

Agradeço a todos os meus colegas e amigos pelo companheirismo, em tantos lugares e em tantas situações... são pessoas que eu fui encontrando, que me cativaram, mas que inevitavelmente se distanciaram, pois assim é a vida!

Sou grato ao CNPq, pelas bolsas de doutorado e de doutorado-sanduíche e pela taxa de bancada.

Sou, também, profundamente grato ao Professor Dr. Alexandre Zanatta, meu orientador de Graduação, por ter feito mais do que me orientar em um TCC. Foi a partir dos conselhos dele que passei a crer que as barreiras são intransponíveis somente se, assim, eu quiser.

Meus sinceros agradecimentos ao meu orientador de Mestrado, Professor Dr. Ricardo Massa, por ter coragem o suficiente para acreditar que "frutos poderiam ser colhidos" do meu trabalho. E desculpe-me por não ter ficado para o doutorado em Pernambuco.

Agradeço também ao professor Robi Malik, da Universidade de Waikato, na Nova Zelândia, por ter me aceito para os 9 meses de estágio-sanduíche e por ter se engajado fortemente na nossa ideia. Aliás, agradeço à Nova Zelândia inteira, por ser tão bela, tão simples, tão hospitaleira, tão *amazing*. Foi um período mágico que deixa muitas saudades.

Obrigado à minha banca de defesa de qualificação e de tese. Professores André Leal, Antonio Carrilho e Patrícia Pena e, também, ao meus mais que professores, meus amigos do DAS, Jean-Marie e Leandro.

Obrigado ao Max, meu coorientador, pelas palavras sempre bem postas e raciocínios profundos, inteligentes e tecnicamente decisivos.

Completando a lista de "agradecimentos acadêmicos" com chave de ouro, presto continência ao meu orientador de Doutorado, Professor Dr. José Cury. Que grata surpresa me reservou o destino ao me direcionar para um trabalho com esse profissional tão espetacular, com essa pessoa tão diferenciada... surpreendi-me, inúmeras vezes, com a forma pura e precisa com que o Cury conduz as coisas e com a sua agilidade

de pensamento. Meu ídolo!

Agradeço à minha mãe, Lourdes, pela bondade - o princípio de tudo. Também, sou-lhe grato por me ensinar que as melhores e maiores lições de nossas vidas provêm da simplicidade das coisas. É justo também agradecer ao meu pai, Darci (*in memorian*), meu modelo de comportamento. Eu sei que sempre estás comigo e que sempre estarás ao meu lado, me iluminando o tempo todo ... que assim seja, para sempre.

Por fim, divido este doutorado (doaria se pudesse) à Lovania: minha esposa, amiga, cúmplice e conselheira, dona da única opinião que realmente importa. Deu cada passo comigo, abdicou da própria carreira em função da minha, foi "carregada" para Recife, Florianópolis, Nova Zelândia e sabe-se lá para onde mais será levada... talvez para Pato Branco, para junto da nossa casa, do nosso filho, do nosso cachorro, todos tão planejados.

"Se enxerguei mais longe, foi porque me apoiei nos ombros de gigantes"

Isaac Newton

RESUMO

Esta tese de doutorado investiga dois aspectos relevantes na *Teoria do Controle Supervisório* (TCS) de *Sistemas a Eventos Discretos* (SEDs): (i) o grau de dificuldade enfrentado ao modelar especificações para serem cumpridas pelo sistema sob controle; (ii) a complexidade computacional do procedimento de síntese de uma solução para o Problema do Controle Supervisório (PCS). Para tratar desses aspectos, são propostas duas diferentes abordagens: o uso de *Distinguidores* e o uso de *Autômatos Finitos Estendidos* (AFEs).

A abordagem com distinguidores consiste em refinar o conjunto original de eventos do modelo de um SED, em um novo conjunto. Cada refinamento é apropriadamente escolhido para identificar uma instância particular em que o evento original ocorre no sistema. Então, um mapa chamado distinguidor é proposto para estabelecer a relação entre as cadeias dos alfabetos original e refinado. Fazendo-se uso de eventos refinados, pode-se simplificar a modelagem de especificações de controle cuja representação seria bastante complexa no alfabeto original. Além disso, é mostrado que resolver o PCS usando distinguidores leva diretamente à solução ótima de controle, porém, sem vantagens computacionais na síntese em relação ao método não-refinado. Nesse sentido, é também mostrado como construir *aproximações* para o modelo refinado de um SED. Ao serem usadas na síntese de supervisores, as aproximações permitem reduzir o custo computacional do procedimento, ao mesmo tempo em que permitem preservar a controlabilidade, a máxima permissividade e o não-bloqueio da solução de controle.

A segunda proposta a ser apresentada consiste em modelar um SED através de AFEs. Os AFEs são estruturas de estados cujas transições são estendidas com *fórmulas* que atualizam variáveis de tal modo que seus valores passam a fazer parte dos estados. Assim, fazendo-se uso de valores de variáveis, pode-se facilmente expressar a semântica de uma especificação através de condições lógicas, implementadas sobre o modelo de um SED. Mostra-se que o uso de AFEs leva diretamente à solução ótima para o PCS a qual, no entanto, é obtida sem vantagens computacionais, em relação ao método convencional. Isso ocorre, porque os valores das variáveis, ainda que implícitos no modelo do sistema, precisam ser considerados na síntese, o que elimina possíveis ganhos trazidos pela simplificação da modelagem. Nesse sentido, propõe-se um método para *abstrair* certas variáveis no AFE que modela um SED. Ao

serem usadas na síntese, tais abstrações reduzem o custo computacional do procedimento, ao mesmo tempo em que permitem preservar a controlabilidade, a máxima permissividade e o não-bloqueio da solução de controle. Um algoritmo para obter supervisores, a partir de AFEs, também é apresentado e ilustrado por meio de um exemplo.
Sempre que possível, as duas abordagens propostas são comparadas e ilustradas por exemplos. Em particular, o exemplo de um sistema de manufatura é adotado ao longo da tese para permitir a análise dos diferentes métodos de síntese.
Como contribuição final, propõe-se associar o uso de distinguidores a um método descentralizado de síntese, em particular ao *Controle Modular Local* (CML). Inicialmente, mostra-se que o uso direto de distinguidores, em geral, complexifica a resolução do CML. Essa inconveniência é mitigada pela síntese combinada, um método por meio do qual supervisores locais são obtidos usando distinguidores apenas quando apropriado. Esse método leva a um comportamento global controlado que é equivalente ao CML original, porém, que também é processado com equivalente custo computacional. Nesse sentido, mostra-se ainda como combinar vantagens do CML, de distinguidores e de aproximações. O mesmo exemplo do sistema de manufatura ilustra essa contribuição final.
Palavras-chave: Sistemas a Eventos Discretos. Teoria do Controle Supervisório. Distinguidores. Aproximações. Autômatos Finitos Estendidos. Abstrações. Controle Modular Local.

ABSTRACT

This doctoral dissertation deals with two relevant aspects of the Supervisory Control Theory (SCT) of Discrete Event Systems (DES): (i) the degree of difficulty faced when modelling specifications to be fulfilled by the system under control; (ii) the computational complexity of the procedure to synthesize a control solution to the Supervisory Control Problem (SCP). In order to address these aspects, one proposes two different approaches: the use of distinguishers and the use of Extended Finite-state Automata (EFA).

The approach with distinguishers consists in refining the original set of events of a DES model into a new set. Each refinement is properly chosen to identify a particular instance in which the original event occurs in the system. A map named *Distinguisher* is then proposed to establish the relationship among strings of the original and refined alphabets. By using refined events, one can simplify the modelling of control specifications whose design would be rather complex in the original alphabet. Besides this, it is shown that solving the SCP using distinguishers directly leads to the optimal control solution, yet without providing computational advantages in synthesis with respect to the nonrefined method. In this sense, one shows how to construct *approximations* for the refined DES model. Approximations can be used in synthesis, reducing its computational cost, while preserving controllability, least restrictiveness and nonblocking of the control solution.

The second approach to be proposed consists in modelling a DES using EFA. EFA are structures of states whose transitions are augmented with *formulas* that update variables in such a way that their values turn to be part of the reached states. Thus, by taking advantages of variable values, one can easily model formulas to restrict a DES model. One shows that EFA can be used in synthesis and this directly leads to the optimal solution to the SCP, yet without providing computational advantages. This happens because variable values have to be unfolded in synthesis, eliminating possible gains brought by simplifying the modeling. In this sense, one shows how to *abstract* some variables in the EFA modeling a DES. Abstractions can be used in synthesis, reducing its computational cost, while preserving controllability, least restrictiveness and nonblocking of the control solution. An algorithm to compute supervisors from EFA is also proposed and illustrated by an example.

In the dissertation, whenever possible, the two proposed approaches are compared and illustrated by examples. In particular, an example of a manufacturing system is adopted along the document to allow to compare the different synthesis methods.
As a final contribution, one proposes to associate distinguishers to a decentralised method of synthesis, in particular to the *Local Modular Control* (LMC). It is initially shown that the direct use of a distinguisher, in general, complexifies the resolution of the LMC. This inconvenience is mitigated by the combined synthesis, a method through which local supervisors are obtained with distinguishers only when appropriate. The combined method leads to a global controlled behaviour that is equivalent to the original LMC, however, it is processed also with equivalent computational cost. In this sense, it is further shown how to combine advantages from LMC, distinguishers and approximations. The same example of the manufacturing system illustrates this final contribution.
Keywords: Discrete Event Systems. Supervisory Control Theory. Distinguishers. Approximations. Extended Finite-state Automata. Abstractions. Local Modular Control.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Os sistemas computacionais são um suporte fundamental para o crescimento da cadeia produtiva industrial, influenciando diretamente na economia do mundo globalizado. Assim sendo, é natural que esses sistemas evoluam muito rapidamente e também se tornem cada vez mais abrangentes e complexos. Fato tal decorre da grande demanda por tecnologia e da aparentemente inesgotável disponibilidade de recursos computacionais que, quando aliados à criatividade dos projetistas de *software*, dão origem a sofisticados sistemas automatizados.

O que se observa atualmente, no entanto, é que o aporte tecnológico tem tomado dimensões tais que a capacidade de raciocínio humano, acerca de uma possível solução técnica, pode ser um gargalo no desenvolvimento de sistemas. Além do mais, a exigência por flexibilidade nos processos industriais torna inviável depender apenas da inspiração do engenheiro para o desenvolvimento de soluções computacionais. Rapidamente essa prática vem sendo aliada a operações automáticas executadas sobre modelos de sistemas, representados com alto nível de abstração e, assim, em melhor sintonia com a percepção humana.

A modelagem de um sistema se justifica por inúmeras razões. Dentre elas, o fato de que nem sempre é possível, ou mesmo seguro, atuar experimentalmente sobre sua estrutura real, tendo em vista os efeitos práticos desse tipo de interferência. Além disso, um modelo permite abstrair detalhes do sistema modelado que podem ser irrelevantes em determinado contexto, facilitando, assim, compreender o sistema, manipulá-lo a partir de diferentes perspectivas, entre outras coisas. Técnicas avançadas de Engenharia de *Software* integram modelos para diversos fins, incluindo a síntese automática de sistemas, a verificação formal, a geração de código, etc.

O conteúdo desta tese também se desenvolve sob esse prisma, ou seja, sob a modelagem de sistemas para a síntese automática de *software*. A particularidade é que a proposta é centrada na síntese de controladores para uma família específica de sistemas, os chamados *Sistemas a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2008). Os SEDs têm como característica o fato de serem guiados pela ocorrência assíncrona de eventos, em oposição aos sistemas dirigidos pelo tempo, e abrangem importantes categorias de sistemas, como os de manufatura, de telecomunicações, de transações eletrônicas, de jogos e entretenimento, etc.

Em geral, os subsistemas que compõem um SED apresentam a

tendência de operar em paralelo, concorrendo por recursos do sistema e guiados pela ocorrência assíncrona de eventos. Essa característica altera o estado global do sistema nas mais variadas e imprevisíveis maneiras, limitando bastante a compreensão, e consequentemente, a ação humana sobre o seu comportamento. Esse cenário se torna ainda mais complexo quando se trata de um SED de grande porte, ou quando o SED é modificado para envolver novos componentes, ou mesmo para alterações nos seus requisitos funcionais, o que é bastante comum na prática (QUEIROZ; CURY, 2000b) (MALIK; FABIAN; ÅKESSON, 2011).

Se por um lado é um desafio controlar o comportamento de um SED, por outro lado, essa é uma necessidade. Além do mais, por uma questão de sobrevivência de mercado, o desempenho e a qualidade da cadeia produtiva imperam no meio industrial, e normalmente são alcançados através do controle ótimo dos inúmeros recursos integrados. Controlar adequadamente o comportamento de um SED e, ao mesmo tempo, garantir o aproveitamento máximo de seus recursos, é complexo, em geral.

Um método que permite intervir, precisamente, no comportamento de um SED, é dado pela *Teoria do Controle Supervisório* (TCS) (RAMADGE; WONHAM, 1987b, 1989). A TCS baseia-se na *Teoria dos Autômatos Finitos (AFs) e Linguagens* para descrever formalmente a síntese de supervisores (ou controladores) ótimos para SEDs. Supervisores ótimos são estruturas que atuam sobre o modelo de um SED de maneira minimamente restritiva e não-bloqueante, desabilitando eventos conforme especificações de controle.

Embora seja amplamente explorada e constantemente estendida e melhorada, a TCS apresenta dois aspectos que limitam a prática do controle supervisório, a saber:

- o esforço computacional necessário para sintetizar supervisores ótimos; e
- o esforço despendido pelo engenheiro para expressar certas especificações que devem ser cumpridas sob controle.

Em termos gerais, esses dois aspectos direcionam o conteúdo desta tese e estão estreitamente relacionados ao conceito de *informação*. Informação é um aspecto chave para a resolução de um *Problema de Controle Supervisório* (PCS) de SEDs. A forma como as informações a respeito do sistema são interpretadas, disponibilizadas e manipuladas, influencia nas condições para a resolução do problema. Algumas dessas condições se relacionam à existência ou não de uma solução, ao custo computacional envolvido para computar essa solução, à dificuldade para

expressar as especificações que, ao serem cumpridas sob controle, levam à solução, etc.

Ao controlar um SED, pode ocorrer que as restrições impostas pelos canais de comunicação, que conectam o sistema aos controladores, induzam à observabilidade parcial dos eventos do sistema. Isso pode se dar devido à impossibilidade de observar certos eventos, tais como os sinais de sensores (LIN; WONHAM, 1988; CIESLAK et al., 1988; LIN; WONHAM, 1990; HUANG; RUDIE; LIN, 2008), ou devido ao uso de arquiteturas descentralizadas de controle, caso em que os controladores locais possuem acesso a apenas uma parte dos eventos gerados pelo sistema (RUDIE; WONHAM, 1992; LIN; WONHAM, 1990; YOO; LAFORTUNE, 2002a).

A observação parcial de eventos altera as condições de existência de uma solução para o PCS, em relação à observabilidade total. Mesmo que uma solução exista, a forma como ela é calculada é particular, assim como o é a complexidade do procedimento de síntese. Alguns trabalhos têm investigado formas de obter um conjunto de eventos que, quando observáveis, garantem a existência de uma solução para o PCS (YOO; LAFORTUNE, 2002b; HAJI-VALIZADEH; LOPARO, 1996; KHULLER; KORTSARZ; ROHLOFF, 2004).

A informação também pode ser examinada de outra perspectiva: a do nível de abstração com que o sistema é modelado. Esse aspecto influencia diretamente na forma como as especificações de controle podem ser expressas. Por exemplo, considere a ocorrência de um certo evento observável ao longo da evolução do sistema. Dependendo do nível de abstração com que o sistema é visto (modelado), o engenheiro de controle pode ter de rastrear (memorizar) outras combinações de eventos que levam a uma ocorrência específica, para que, só então, consiga interpretar o significado correto daquela ocorrência. É o caso, por exemplo, da modelagem de uma especificação que previne o *underflow* em um *buffer* qualquer. Modelar essa especificação pode depender do rastreamento de todos os eventos que inserem e retiram elementos do *buffer*, para que, só então, se possa identificar o evento que o torna vazio. Por outro lado, se o nível de abstração adotado for tal que explicita, dentre todos os eventos possíveis, aquele evento que de fato torna o *buffer* vazio, então expressar essa especificação tende a ser mais simples.

Algumas extensões do *controle hierárquico* focam na abstração do modelo original do sistema em um modelo de mais alto nível, a partir do qual algumas especificações, originalmente complexas, passam a ser facilmente modeladas através do uso de símbolos de alto nível. Duas

dessas extensões podem ser identificadas na literatura: a primeira constrói o modelo abstrato apagando eventos do modelo original através de *projeções naturais* (CUNHA; CURY, 2007; SCHMIDT; MOOR; PERK, 2008; FENG; WONHAM, 2008); a segunda, atribui ao modelo abstrato seu próprio conjunto de eventos e constrói o modelo em alto nível com base em um *mapa-repórter*, que noticia quando o modelo original alcança certos estados (ZHONG; WONHAM, 1990; WONG; WONHAM, 1996). Em ambos os casos, cada ocorrência de evento no modelo abstrato pode representar a ocorrência de uma cadeia de eventos no modelo original.

Em oposição à abordagem *bottom-up*, sobre a qual o controle hierárquico é fundamentado, pode acontecer que o nível de abstração com o qual um SED é modelado seja, originalmente, alto. Nesses casos, modelar especificações de controle também pode ser complexo. Enquanto a habilidade humana para estruturar uma modelo limita-se a algumas dezenas de estados, o modelo para uma única especificação pode envolver milhares de estados (CURY et al., 2012; TEIXEIRA; CURY; QUEIROZ, 2011, 2013). Dessa forma, explorar uma abordagem *top-down* para a modelagem de SEDs pode ser de grande valia. Nessa abordagem, ao invés de ser abstraído, o modelo original do sistema é enriquecido (refinado) com informações acerca da ocorrência de certos eventos (BOUZON; QUEIROZ; CURY, 2008, 2009), ou sobre o *status* de certos estados (SKÖLDSTAM; ÅKESSON; FABIAN, 2007; OUEDRAOGO et al., 2011), o que pode viabilizar a modelagem de especificações complexas. Nesses casos, cada evento do modelo original representa diferentes circunstâncias no universo refinado, explicitando a oposição à abordagem *bottom-up*.

Esta tese é estruturada sobre a abordagem *top-down*. Dentre seus objetivos gerais está a estruturação de dois métodos para modificar o modelo de um SED, refinando-o conforme a abordagem *top-down*, mencionada anteriormente. Após serem individualmente apresentados, discutidos e exemplificados, esses dois métodos são ainda comparados, para que se oriente o engenheiro na aplicabilidade de cada um deles. Finalmente, um dos métodos é estendido a uma abordagem de síntese descentralizada. Esses objetivos gerais são especificados a seguir.

O primeiro método a ser apresentado baseia-se no refinamento do conjunto original de eventos do AF que modela um SED. O modelo refinado do sistema, além de incorporar um novo conjunto de eventos, é também associado a um *Distinguidor*, um mecanismo que mapeia cada cadeia de eventos originais em cadeias compostas por eventos refinados. Cada cadeia refinada representa ocorrências dos eventos originais com uma semântica em particular. A ideia de um distinguidor foi originalmente proposta em Bouzon, Queiroz e Cury (2008, 2009). Esta tese dá

continuidade à investigação relacionada a distinguidores, generalizando, melhorando e estendendo os resultados em diversas esferas.

Já o segundo método a ser apresentado, baseia-se no uso de *Autômatos Finitos Estendidos* (AFEs). Os AFEs são estruturas de estados, similares aos AFs, mas estendidas com fórmulas associadas às transições de estados do modelo. Essas fórmulas atualizam variáveis após cada transição, de modo que os valores das variáveis passam a compor a semântica dos estados. Assim, ao invés de refinar informações sobre eventos, como no caso dos distinguidores, essa abordagem refina informações sobre os estados de um modelo. O uso desse método remete a vantagens equivalentes às observadas com o refinamento de eventos, tanto em relação à modelagem quanto à síntese de supervisores. Embora o uso de AFEs não seja inédito na TCS, esta tese estrutura a abordagem de maneira mais vantajosa (e, sob certo aspecto, mais simples) em comparação aos trabalhos encontrados na literatura (SKÖLDSTAM; ÅKESSON; FABIAN, 2007; OUEDRAOGO et al., 2011), principalmente em relação ao custo computacional da síntese de supervisores.

Será mostrado, no decorrer da tese, que ambos os métodos usados para refinar o modelo de um SED, distinguidores ou AFEs, permitem:

(i) viabilizar a modelagem de especificações complexas de controle, fazendo-se uso dos eventos refinados ou dos valores das variáveis;

(ii) obter uma solução ótima de controle;

(iii) computar as soluções refinadas com um custo computacional equivalente ao da síntese original (caso essa fosse processada sem refinamentos).

Mesmo em face às claras vantagens, sumarizadas nos itens acima, os métodos propostos não permitem, diretamente, reduzir a complexidade de síntese. Assim sendo, o esforço computacional permanece sendo um obstáculo para a resolução do PCS. Nesse sentido, uma contribuição adicional desta tese é mostrar como os distinguidores e os AFEs podem ser explorados para a construção, respectivamente, de *aproximações* e *abstrações* para o modelo refinado de um SED. Ao serem usadas em síntese, tais aproximações ou abstrações permitem resolver o PCS com ganhos computacionais. Ainda, são derivadas as condições que garantem a controlabilidade, o não-bloqueio e a máxima permissividade da solução de controle, nesses casos.

Como contribuição final desta tese (não necessariamente expostas nessa ordem textual) propõe-se associar a ideia de refinamentos a uma arquitetura descentralizada de controle, aqui representada pelo

Controle Modular Local (CML) (QUEIROZ; CURY, 2000b, 2000a). Como justificativa, argumenta-se que, mesmo em face às vantagens providas pelos métodos de refinamento propostos, a síntese de supervisores permanece sendo processada monoliticamente. Em contrapartida, o CML define um método de síntese descentralizado. Assim, propõe-se estender o CML para incorporar o refinamento de eventos. Essa extensão é feita de forma a se obter as vantagens paralelas de distinguidores, aproximações e CML.

A ideia de refinar o modelo de um SED, como apresentada nesta tese, tem sido adotada na literatura para propósitos específicos. Por exemplo, o refinamento de eventos tem sido usado como um dos passos da *síntese composicional* (MOHAJERANI et al., 2011), para evitar não-determinismo de AFs. Essa estratégia otimiza a construção de abstrações para reduzir o custo computacional de síntese, mas não trata da simplificação da modelagem. Em contrapartida, a simplificação da modelagem é alvo de investigações através do uso de AFEs (OLIVEIRA; CURY; KAESTNER, 2004; CHEN; LIN, 2000; KUMAR; GARG, 2005; OUEDRAOGO et al., 2011). Porém, essas abordagens não implementam a ideia de usar abstrações para reduzir o custo computacional de síntese.

As contribuições desta tese são ilustradas no contexto de diversos exemplos. Em particular, um sistema de manufatura com retrabalho de materiais é introduzido no Capítulo 3 e retomado nos demais capítulos para ilustrar as propostas apresentadas, facilitando, assim, comparálas. Esse exemplo contém uma especificação complexa, que é então modelada com o auxílio de distinguidores e também de AFEs. Então, a síntese é processada em cada abordagem, inicialmente sobre a estrutura refinada e posteriormente sobre aproximações/abstrações para os modelos refinados. Quando se obtém uma solução subótima, mostra-se como a síntese pode ser melhorada para garantir o resultado minimamente restritivo e, ao mesmo tempo, manter importantes ganhos computacionais. Outros exemplos são abordados: o algoritmo de síntese com AFEs é ilustrado através de um exemplo simples de um sistema de manufatura (que permite descrever o passo-a-passo da operação) e um exemplo de um sistema de estocagem também é apresentado para ilustrar a associação entre distinguidores e AFEs.

A estrutura da tese é organizada conforme descrição a seguir.

O Capítulo 2 traz um apanhado geral sobre os SEDs. Discutem-se as características, propriedades e algumas alternativas para a modelagem dessa classe de sistemas, em especial, os Autômatos e as Linguagens, cujos conceitos básicos também são apresentados no capítulo.

O Capítulo 3 apresenta a TCS, conforme originalmente proposta.

Apresenta-se um problema de controle cuja resolução é ilustrada através de um exemplo.

O Capítulo 4 introduz a ideia do refinamento de eventos. O conceito de um *distinguidor* é apresentado, suas propriedades são descritas, e seu uso na TCS é exemplificado. Este capítulo também estrutura o conceito de *D-aproximações*, sua construção, propriedades e seu uso na TCS, para uma síntese computacionalmente eficiente.

O Capítulo 5 apresenta o CML e suas extensões para envolver distinguidores e aproximações. Um exemplo é usado para ilustrar as diversas alternativas de síntese modular local.

O Capítulo 6 descreve a abordagem de síntese com AFEs. Após apresentar suas principais propriedades, mostra-se como o PCS pode ser resolvido a partir de AFEs. Em seguida, introduz-se uma classe de abstrações para serem usadas em síntese. Um algoritmo para a computação de supervisores, a partir de AFEs, também é apresentado e exemplificado.

As abordagens apresentadas nos capítulos 4 e 6 são comparadas no Capítulo 7, e exemplos são apresentados para ilustrar como elas se relacionam.

Finalmente, o Capítulo 8 traz as considerações finais, os objetivos alcançados com a tese, e algumas sugestões de trabalhos futuros.

2 SISTEMAS A EVENTOS DISCRETOS

A modelagem computacional de sistemas é estruturada sobre dois fundamentos básicos:

(i) o conceito de *estado*, que identifica o status do sistema em determinada circunstância; e

(ii) um mecanismo de *transição de estados*, caracterizando a evolução do sistema.

Assim, classificar uma família de sistemas que compartilhe de alguma característica em comum, passa, sobretudo, por caracterizá-los dentro desses dois fundamentos. Por exemplo, alguns sistemas têm em comum o fato de que seus estados são mapeados continuamente e suas estruturas de transição são regidas pelo tempo. Inúmeras grandezas físicas compõem essa categoria, como temperatura, fluxo, pressão, etc. Na modelagem de tais sistemas, os estados podem ser representados por variáveis contínuas e a dinâmica de transição, guiada pelo tempo. Assim, equações diferenciais se tornam alternativas naturais para representá-los.

Em outros casos, a modelagem via equações diferenciais pode não ser adequada, pelo fato de que o espaço de estados de um sistema pode não ser continuo, e a dinâmica de transições pode não depender do tempo, mas de eventos instantâneos e assíncronos. Considere, por exemplo, o processamento de um dado em algum dispositivo computacional. Nesse caso, o conjunto de estados possíveis é claramente discreto, e poderia ser dado, por exemplo, por $E = \{parado,\ processando\}$. Ademais, as transições entre esses dois estados visivelmente não se relacionam ao tempo, mas a eventos assíncronos, como *clock*, *entrada de dados*, *interrupção*, etc.

Sistemas que compartilham dessa dinâmica, i.e., que representam estados por conjuntos discretos, e cujos mecanismos de transição são guiados por eventos ocorridos em pontos discretos do tempo, são denominados *Sistemas a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2008). Exemplos de SEDs incluem sistemas de manufatura, de telecomunicações, de transações eletrônicas, jogos, etc.

A modelagem de SEDs se dá, normalmente, através de diagramas de transições, nos quais é possível mapear a evolução do sistema dentro de um espaço de estados, de forma intuitiva, capturando apenas o que é relevante no contexto de uma dinâmica discreta (CURY, 2001). Alguns

dos formalismos existentes para a modelagem de SEDs são apresentadas a seguir.

2.1 MODELAGEM DE SED

A modelagem de um sistema se justifica por inúmeras razões. Dentre elas, o fato de que nem sempre é possível, ou mesmo seguro, atuar experimentalmente sobre sua estrutura real, tendo em vista os efeitos práticos desse tipo de interferência. Além disso, um modelo permite abstrair detalhes do sistema modelado que podem ser irrelevantes para determinado contexto, facilitando assim, compreender o sistema, manipulá-lo a partir de diferentes perspectivas, entre outras coisas.

Sob o ponto de vista dos SEDs, a atividade de modelagem normalmente se utiliza de estruturas discretas de diagramação que suportam representar problemas de natureza, tamanho e complexidade variados. Dentre os formalismos mais conhecidos estão: as *Redes de Petri* (e suas inúmeras extensões) (MURATA, 1989; DESROCHERS; AL-JAAR, 1994), a *Teoria das Filas* (BOLCH et al., 2006; TRIVEDI, 2002), a *Teoria dos Autômatos e Linguagens* (HOPCROFT; ULLMAN; MOTWANI, 2001; ROSEN, 2007), etc. Nenhuma dessas alternativas formais é, contudo, unanimemente aceita. Isso porque cada formalismo suporta, em geral, representar apenas parcialmente as propriedades de um SED.

Nesta tese, a *Teoria dos Autômatos e Linguagens* é adotada como formalismo de modelagem. A escolha se dá, sobretudo, pelo fato de que as operações sobre Autômatos e Linguagens alicerçam a abordagem a ser introduzida no Capítulo 3, a qual é de relevância central para a tese por formalizar a síntese de controladores ótimos para SEDs. Isso pode não ser trivial, ou mesmo possível, através de outros formalismos, que normalmente se limitam a técnicas de análise formal. Nesse sentido, apresentam-se a seguir os principais fundamentos dos Autômatos e Linguagens, no contexto da modelagem de SEDs.

2.2 TEORIA DAS LINGUAGENS FORMAIS

Considere um SED, cuja evolução se dá em razão da ocorrência de eventos. Naturalmente, o resultado dessa dinâmica remete a uma trajetória formada por sequências de eventos. Para cada uma dessas sequências dá-se o nome de *cadeia*. É intuitivo pensar que modelar um

SED nada mais é, então, do que descrever tais cadeias e, analogamente, processar operações sobre o modelo de um SED consiste em manipular suas cadeias. A seguir, essas ideias são formalizadas e algumas operações básicas são introduzidas.

2.2.1 Evento

O primeiro passo para modelar um SED é identificar os *eventos* que guiam a dinâmica do sistema. Cada um desses eventos é então etiquetado por um símbolo, distinto dos demais.

Identificar eventos em um SED pode ser relativizado ao nível de abstração com que o sistema é interpretado, ou à necessidade de observar determinados eventos para fins práticos. De todo modo, tratam-se de atividades de engenharia e que, portanto, dependem da experiência do engenheiro. É comum, porém, que o sistema naturalmente aponte para um conjunto de eventos, como por exemplo sinais de início e fim de operações em máquinas, sinais providos por sensores ao identificar uma situação particular, início e fim de movimentos robóticos, etc.

2.2.2 Alfabeto

Alfabeto é como é conhecido o conjunto de eventos identificados em um SED. Nesta tese, assume-se que o alfabeto é finito, não-vazio e denotado por Σ.

2.2.3 Cadeia

Define-se por *cadeia* uma sequência finita de eventos pertencentes ao alfabeto Σ. Um caso particular de cadeia é a *cadeia vazia* (denotada por ε), a qual corresponde a uma sequência com nenhum evento.

Denota-se por Σ^* o conjunto de todas as cadeias possíveis de serem compostas com eventos em Σ, incluindo a cadeia ε. Convencionalmente, o tamanho de uma cadeia $s \in \Sigma^*$ é denotada por $|s|$, representando a quantidade de eventos que compõem s. Por exemplo, $|abcd| = 4$, $|\varepsilon| = 0$, e assim por diante.

2.2.4 Linguagem

O conceito de linguagem, formalmente definido, não difere da intuitiva noção de linguagem natural, falada. Considerando-se, por exemplo, o vocabulário relacionado a determinado idioma, existe um conjunto de palavras (cadeias) que são coerentes ou corretas no contexto do idioma. Tais palavras são compostas unicamente por letras (símbolos) do seu alfabeto. Mas, também, é possível realizar a livre composição de novas palavras, embora elas possam não ser reconhecidas no vocabulário. À entidade que define esse vocabulário válido dá-se o nome de *linguagem*.

Formalmente, uma linguagem L é um subconjunto de cadeias em Σ^*, i.e., $L \subseteq \Sigma^*$, que determina quais são as combinações possíveis de eventos em Σ que levam a cadeias reconhecidas em determinado contexto (vocabulário).

Uma característica importante das linguagens formais é que elas podem contemplar um número infinito de cadeias, embora cada cadeia seja finita e oriunda de um alfabeto também finito. Por exemplo, seja $\Sigma = \{a, b\}$. Uma linguagem L que contempla todas as cadeias possíveis em Σ^* corresponde a $L = \{\epsilon, a, b, aa, bb, ab, ba, aab, \cdots\}$. Nesse caso, L contém um número infinito de cadeias finitas.

2.2.5 Operações sobre Linguagens

Sendo uma linguagem um conjunto, por definição, então todas as operações convencionais sobre conjuntos - união, intersecção, potência, entre outras - se aplicam às linguagens. Além dessas, outras operações podem ser definidas especificamente para esse contexto. Algumas delas são apresentadas a seguir.

2.2.5.1 Operações Morfológicas

Seja Σ um alfabeto e $s = uvw$ uma cadeia, com $u, v, w \in \Sigma^*$. Então, diz-se que:

- u é um prefixo de s;
- v é uma subcadeia de s;
- w é um sufixo de s;

2.2.5.2 Concatenação

Para duas linguagens, $L_1, L_2 \subseteq \Sigma^*$, a concatenação de L_1 com L_2, denotada por L_1L_2, é definida por:

$$L_1L_2 = \{s \in \Sigma^* \mid (s = s_1s_2) \wedge (s_1 \in L_1 \wedge s_2 \in L_2)\}.$$

2.2.5.3 Prefixo-fechamento

Para uma linguagem $L \subseteq \Sigma^*$, o prefixo-fechamento de L, denotado por $\overline{L}$, é definido por:

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

Ou seja, $\overline{L}$ consiste na linguagem que contém todos os prefixos de todas as cadeias de L. Sendo assim, se L for prefixo-fechada, então $L = \overline{L}$ e qualquer prefixo, de qualquer cadeia de L, também é uma cadeia em L.

2.2.5.4 Projeção Natural de cadeias e linguagens

De maneira informal, projetar uma cadeia definida em Σ_1^* sobre outro conjunto de cadeias Σ_2^* significa aplicar um mapeamento sobre os eventos da primeira cadeia, de forma a obter como resultado uma versão da cadeia original, em Σ_2^*. Um caso particular é aquele no qual $\Sigma_2 \subseteq \Sigma_1$ e a ação de projeção é a de apagar eventos da cadeia original que não estejam no alfabeto projetado.

Formalmente, sejam dois alfabetos Σ_m e Σ_M, tal que $\Sigma_m \subseteq \Sigma_M$. A projeção natural de uma cadeia em Σ_M^* sobre Σ_m^* é obtida através de um mapa $\mathcal{P} : \Sigma_M^* \to \Sigma_m^*$, definido como segue:

$$\begin{aligned}
\mathcal{P}(\varepsilon) &= \varepsilon; \\
\mathcal{P}(\sigma) &= \begin{cases} \sigma & \text{se } \sigma \in \Sigma_m; \text{ ou} \\ \varepsilon & \text{se } \sigma \in \Sigma_M \setminus \Sigma_m; \end{cases} \\
\mathcal{P}(s\sigma) &= \mathcal{P}(s)\mathcal{P}(\sigma), \text{ para } s \in \Sigma_M^* \text{ e } \sigma \in \Sigma_M.
\end{aligned}$$

As projeções de cadeias diretamente se estendem a linguagens se o mapeamento se aplica a todas as cadeias da linguagem. Para uma

linguagem $L \subseteq \Sigma_M^*$, a projeção natural de L sobre Σ_m^* é definida por:

$$\mathcal{P}(L) = \{t \in \Sigma_m^* \mid t = \mathcal{P}(s), \text{ para algum } s \in L\}. \tag{2.1}$$

Outra forma de projeção se dá através do mapeamento inverso de uma cadeia. Considere um mapa $\mathcal{P}^{-1} : \Sigma_m^* \to 2^{\Sigma_M^*}$, definido de modo tal que:

$$\mathcal{P}^{-1}(t) = \{s \in \Sigma_M^* \mid \mathcal{P}(s) = t\}.$$

De maneira análoga à primeira, a projeção inversa também pode ser estendida para linguagens $L_m \subseteq \Sigma_m^*$, tal que:

$$\mathcal{P}^{-1}(L_m) = \{s \in \Sigma_M^* \mid \mathcal{P}(s) = t, \text{ para algum } t \in L_m\}. \tag{2.2}$$

Como se pode notar, a operação de projeção é aninhada na definição do mapa inverso. Ou seja, a projeção inversa de uma cadeia t com eventos em Σ_m é composta pelo conjunto de todas as cadeias com eventos em Σ_M, cujas projeções resultam na própria cadeia t.

2.2.5.5 Composição síncrona de linguagens

Linguagens podem ser formalmente combinadas através da operação de composição síncrona, que é geralmente denotada pelo símbolo $\|$.

Seja Σ um alfabeto tal que $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$, para $i = 1, \cdots, n$, e seja L_i uma linguagem tal que $L_i \subseteq \Sigma_i^*$. A composição síncrona das linguagens L_i é denotada por $\|_{i=1}^{n} L_i \subseteq \Sigma^*$ e é definida como:

$$\begin{aligned} \|_{i=1}^{n} L_i &= \bigcap_{i=1}^{n} \mathcal{P}_i^{-1}(L_i) \\ &= \{s \in \Sigma^* \mid \bigwedge_{i=1}^{n} \mathcal{P}_i(s) \in L_i\}. \end{aligned} \tag{2.3}$$

2.2.6 Classes de Linguagens

De um modo geral, uma linguagem pode ser classificada conforme duas perspectivas distintas: (i) a regularidade e (ii) a não-regularidade. Uma linguagem é regular quando pode ser expressa por um conjunto finito de recursos (muito embora o número de possíveis cadeias que compõem essa linguagem possa ser infinito). Inversamente, uma lin-

guagem é não-regular quando nem sempre pode ser representada por uma estrutura finita.

Dada a natureza dos aspectos abordados nesta tese, é de interesse particular a exploração da classe das linguagens regulares. De fato, note que a modelagem de SEDs normalmente compreende uma quantidade finita de elementos. Em geral, o comportamento desses elementos pode ser representado por mecanismos formais de estruturação, também finitos.

As Expressões Regulares, por exemplo, são notações algébricas capazes de reconhecer todas (e somente) as linguagens regulares (HOPCROFT; ULLMAN; MOTWANI, 2001). Esse formalismo se assemelha às linguagens de programação, no sentido de que o reconhecimento (ou não) de cadeias de uma linguagem é definido declarativamente, por meio de expressões.

Uma das grandes dificuldades da modelagem de SEDs através de Expressões Regulares é a falta de clareza e a complexidade de notação (CASSANDRAS; LAFORTUNE, 2008). Nem sempre é simples representar sequências de eventos de um SED de uma maneira puramente algébrica. Mecanismos de diagramação como, por exemplo, os *Autômatos Finitos*, podem simplificar substancialmente essa atividade. Os Autômatos fornecem uma notação intuitiva de modelagem sem, ao mesmo tempo, abdicar do rigor formal. Ademais, essas estruturas são mais adequadas à manipulação computacional do que as Expressões Regulares. Nesta tese, os Autômatos (e algumas extensões) são adotados como formalismo de modelagem, e são apresentados a seguir.

2.3 AUTÔMATOS FINITOS

Um *Autômato Finito* (AF) é uma estrutura formal de diagramação que permite representar problemas de diversas naturezas, tamanhos e complexidades, através de um método prático e intuitivo. Um AF pode ser formalmente apresentado como uma quíntupla $\langle \Sigma, Q, q^\circ, Q^\omega, \rightarrow \rangle$, onde:

- Σ é o alfabeto finito;
- Q é o conjunto finito de estados;
- $q^\circ \in Q$ é o estado inicial;
- $Q^\omega \subseteq Q$ é o subconjunto de estados marcados;
- $\rightarrow \subseteq Q \times \Sigma \times Q$ é a relação de transição de estados.

Denota-se por $\mathsf{q}_1 \xrightarrow{\sigma} \mathsf{q}_2$, uma transição do estado q_1 para o estado q_2 com o evento $\sigma \in \Sigma$. Por $AF \xrightarrow{s} q$ denota-se que uma cadeia s é possível em um dado autômato AF.

Na prática, a estrutura formal de um autômato pode ser associada a uma convenção gráfica que facilita sua interpretação e manipulação e, ao mesmo tempo, preserva o seu caráter formal. Usualmente, a interface adotada para representar um autômato é inspirada na teoria dos grafos direcionados. Dois exemplos de autômatos são apresentados na Figura 1.

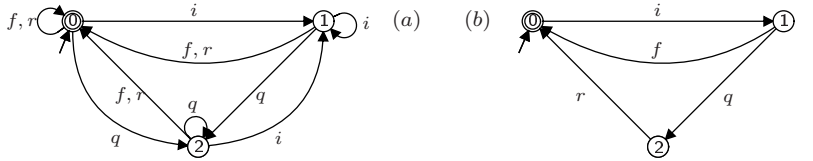


Figura 1 – Exemplo de *Autômato* (a) e de *Gerador* (b).

Os nós do diagrama (círculos) representam os estados do autômato. As mudanças de estado se dão por meio de transições, denotadas por arcos direcionados conectando estados. Cada transição é etiquetada com os eventos que podem desencadeá-la. O estado inicial é indicado por uma seta uniconectada e os estados marcados são denotados por círculos duplos, cuja semântica aponta para um estado aceitável de parada. Em geral, estados marcados são associados à ideia de tarefas completas.

Conceitualmente, as estruturas (a) e (b) da Figura 1 visam modelar o mesmo processo, contemplando o início (evento i) e o fim (evento f) de operação em uma máquina qualquer, admitindo a possibilidade de quebra (evento q) e de reparo (evento r). A diferença é que em (a) a relação $\rightarrow$ é, na verdade, uma função definida para todo elemento de Σ, i.e., em cada um dos estados, todos os eventos são possíveis, ao contrário do que é modelado em (b).

Na modelagem de SEDs, é bastante provável que apenas um subconjunto de eventos esteja habilitado em cada estado. Na Figura 1, por exemplo, estando o sistema no estado inicial $\mathsf{0}$, a transição $\mathsf{0} \xrightarrow{\mathsf{q}} \mathsf{2}$ não deve ser modelada, no sentido de expressar que a quebra de uma máquina não ocorre em seu estado de repouso. Esse raciocínio remete ao conceito de *Gerador* (b), uma classe particular de autômatos para a qual a relação $\rightarrow$ representa uma função parcial, i.e., não é definida para todo evento, em cada estado. Para fins práticos, nesta tese a

menção a autômato incorpora indistintamente o conceito de gerador.

2.3.1 Autômatos como reconhecedores de linguagens

Como visto, um autômato permite implementar a noção de estados, transições e eventos e, assim, se torna uma escolha direta para a modelagem de SEDs. Em muitos casos, o comportamento modelado por um autômato precisa ser adicionalmente interpretado em termos de linguagens, i.e., em razão das possíveis cadeias de eventos produzidas pelo autômato.

Dois tipos de linguagens podem ser definidas a partir de um autômato A qualquer: a *linguagem gerada* e a *linguagem marcada*, definidas respectivamente como segue.

$$\begin{aligned} L(A) &= \{s \in \Sigma^* \mid A \xrightarrow{s} q \in Q\}; \\ L^\omega(A) &= \{s \in \Sigma^* \mid A \xrightarrow{s} q \in Q^\omega\}. \end{aligned}$$

Em palavras, a *linguagem gerada* $L(A)$ contempla o conjunto de todas as cadeias possíveis do autômato A, enquanto que a *linguagem marcada* $L^\omega(A)$ representa o conjunto de todas as cadeias que levam a estados marcados. $L(A)$ e $L^\omega(A)$ sintetizam, em termos de linguagem, o comportamento modelado por AFs.

2.3.2 Composição síncrona de autômatos

Assim como as linguagens, os autômatos também podem ser compostos entre si, resultando um modelo cujo comportamento combina os comportamentos gerado e marcado individuais. Nesta tese, a mesma notação usada para compor linguagens ($\|$), é também adotada para compor autômatos, e é definida como segue.

Sejam dois autômatos, $A = \langle \Sigma_A, Q_A, q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$ e $B = \langle \Sigma_B, Q_B, q_B^\circ, Q_B^\omega, \rightarrow_B \rangle$. A composição síncrona de A e B é definida pelo autômato

$$A \| B = \langle \Sigma_A \cup \Sigma_B, Q_A \times Q_B, (q_A^\circ, q_B^\circ), Q_A^\omega \times Q_B^\omega, \rightarrow \rangle,$$

em que:

- $(q_A, q_B) \xrightarrow{\sigma} (q'_A, q'_B)$, se $\sigma \in \Sigma_A \cap \Sigma_B$;

- $(q_A, q_B) \xrightarrow{\sigma} (q'_A, q_B)$, se $\sigma \in \Sigma_A \setminus \Sigma_B$;
- $(q_A, q_B) \xrightarrow{\sigma} (q_A, q'_B)$, se $\sigma \in \Sigma_B \setminus \Sigma_A$.

A operação $\|$ força a ocorrência simultânea dos eventos compartilhados e intercala, em qualquer ordem, os não compartilhados. Então, dados os autômatos $G^i = \langle \Sigma_i, Q_i, q_i^\circ, Q_i^\omega, \rightarrow_i \rangle$, para $i = 1, \cdots, n$, um modelo global G pode ser obtido por meio da composição síncrona

$$G = \|_{i=1}^{n} G^i, \text{ tal que } \Sigma = \bigcup_{i=1}^{n} \Sigma_i.$$

Pode-se mostrar (WONHAM, 2002) que os comportamentos gerado e marcado são tais que:

$$L(G) = \|_{i=1}^{n} L(G^i) \text{ e } L^\omega(G) = \|_{i=1}^{n} L^\omega(G^i).$$

A composição síncrona é particularmente útil por permitir que se lide individualmente com modelos mais simples, compondo-os *a posteriori*.

Uma classe particular de composição de autômatos pode ser obtida com base na sincronia de eventos. Sejam G^1 e G^2 dois autômatos definidos sobre Σ^{G^1} e Σ^{G^2}, respectivamente. Diz-se que G^1 e G^2 são assíncronos se $\Sigma^{G^1} \cap \Sigma^{G^2} = \emptyset$. Uma forma de se obter autômatos assíncronos (representá-los via *sistema-produto* (RAMADGE, 1989)) se dá por meio da composição síncrona ($\|$) de todos, e somente, os autômatos individuais que são síncronos entre si.

2.4 SUMÁRIO

Neste capítulo foram apresentados os SEDs, uma classe de sistemas cujo comportamento evolui em razão de eventos e não do tempo. Por compartilharem de um conjunto de características discretas, os SEDs são tipicamente modelados por estruturas também discretas, e não via equações diferenciais, por exemplo.

A *Teoria dos Autômatos e Linguagens* foi então apresentada como uma alternativa natural para a modelagem de SEDs.

Por fim, foram introduzidas algumas das operações usualmente aplicadas sobre autômatos e linguagens, que permitem modelar propriedades comuns aos SEDs.

Nos capítulos seguintes, os autômatos e as linguagens são investigados em mais detalhes no contexto da modelagem e controle de SEDs, à luz da *Teoria do Controle Supervisório*.

3 TEORIA DE CONTROLE SUPERVISÓRIO

Lógicas de controle para SEDs podem, em geral, ser implementadas usando-se os próprios formalismos utilizados na modelagem. Na literatura, algumas frentes investigam o controle de SEDs através da Álgebra de Dióides (Max-Plus) (SPACEK et al., 1996; SCHUTTER; BOOM, 2008), outras usam Lógica Temporal (PNUELI, 1977; JIANG; KUMAR, 2006), ainda outras exploram a Teoria das Redes de Petri (GIUA; DICESARE, 1994; DARONDEAU, 2005) e dos Autômatos e Linguagens (CASSANDRAS; LAFORTUNE, 2008; RAMADGE; WONHAM, 1987b), etc.

Na maioria dessas abordagens, a lógica de controle é implementada de maneira empírica e então é checada para a análise das propriedades de interesse, tais como o *não-bloqueio*, a *vivacidade*, a *limitação*, etc. Nesse sentido, a *Teoria do Controle Supervisório* (TCS) diferencia-se por permitir a síntese automática de supervisores (controladores) ótimos para SEDs.

Estruturada fundamentalmente na Teoria dos Autômatos e Linguagens (abordada no Capítulo 2), a TCS permite que o comportamento de um SED, e suas especificações de controle, sejam individualmente descritos por meio de um conjunto de autômatos. Então, operações matemáticas são efetivadas sobre a composição desses autômatos, levando a uma lógica de controle capaz de supervisionar o sistema e interferir no seu comportamento de forma minimamente restritiva e não-bloqueante. A TCS é, em geral, aplicável a toda a classe dos SEDs, muito embora alguns aspectos relativos à, por exemplo, natureza do problema de controle ou ao porte do sistema a ser controlado, apresentem implicações importantes, em termos de complexidade, e são objetos de investigação nesta tese.

Os fundamentos básicos da TCS (RAMADGE; WONHAM, 1987b, 1989) têm sustentado, ao longo dos anos, uma vasta linha de pesquisas, levando a uma série de novas propostas e adendos à teoria inicial. Dentre elas, o Controle Temporizado (BRANDIN; WONHAM, 1993), o Controle Hierárquico (CUNHA; CURY, 2007), o controle com observabilidade parcial do sistema (LIN; WONHAM, 1990), o Controle Modular Clássico (RAMADGE; WONHAM, 1987a), a abordagem Modular Local (QUEIROZ; CURY, 2000b), o uso de autômatos estendidos com variáveis (OUEDRAOGO et al., 2011), o refinamento de eventos (BOUZON; QUEIROZ; CURY, 2009), entre muitos outros.

Este capítulo apresenta a TCS, tal como originalmente proposta por Ramadge e Wonham. Discutem-se os procedimentos, as proprie-

dades e as condições elementares para a obtenção de supervisores para SEDs. Um exemplo, então, ilustra a abordagem. No contexto desse mesmo exemplo, são apontados alguns aspectos que podem limitar a aplicabilidade prática da TCS. São esses aspectos que fundamentam esta tese.

3.1 MODELAGEM DO SISTEMA

O primeiro passo a ser considerado na TCS de SEDs, é a modelagem dos componentes do sistema. Em um SED, esses componentes (ou subsistemas) são individualmente organizados dentro de um arranjo lógico, de modo que a execução conjunta desses elementos determina o comportamento global do sistema, também conhecido como *planta*. Uma planta está em *malha aberta* quando reflete o comportamento do sistema sem qualquer interferência externa, coordenação ou ação de controle.

Modelar a planta de um SED em malha aberta requer, sobretudo, a identificação do comportamento de cada subsistema. Logo, o modelo da planta pode assumir diferentes níveis de abstração e a escolha de qual deles usar é uma tarefa puramente de engenharia. Na TCS, autômatos são empregados como formalismo de modelagem e é usual que os subsistemas sejam individualmente representados. Assim, obter o modelo para a planta de um SED consiste na associação dos autômatos de cada subsistema, normalmente materializada pela operação de composição síncrona ($\|$).

A Figura 2 ilustra um exemplo simples de um SED composto por duas máquinas (subsistemas), M_1 e M_2, modeladas respectivamente por G^1 e G^2, tal que $G = G^1 \| G^2$ representa o modelo da planta em malha aberta.

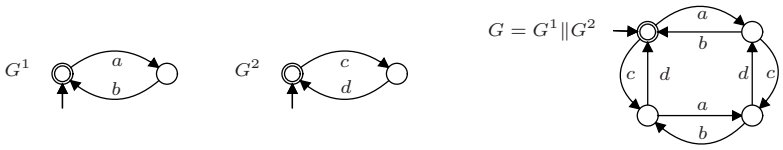


Figura 2 – Modelagem da planta de um SED.

Assume-se que o comportamento das máquinas M_1 e M_2 é descrito em termos de eventos de início e final de operações. Os eventos a e c modelam os respectivos inícios, enquanto b e d modelam os finais

de operações. Quando compostos, G^1 e G^2 resultam no modelo global da planta, G, tal que $L(G)$ representa o comportamentos gerado em malha aberta e $L^{\omega}(G)$ o seu comportamento marcado, representando tarefas completas pelo sistema.

3.2 MODELAGEM DAS ESPECIFICAÇÕES

Uma vez modelada a planta em malha aberta, o próximo passo à luz da TCS é *fechar a malha*. Para isso, são implementadas restrições as quais espera-se que o sistema cumpra quando em funcionamento. Restringir uma planta passa, basicamente, pela definição do que restringir e, em seguida, pela implementação da semântica de cada uma das restrições em forma de especificações de controle.

Na TCS, uma especificação nada mais é do que a representação de uma ação proibitiva no sistema. Quando associada ao modelo da planta, ela interfere em determinados eventos possíveis em malha aberta, adequando o comportamento da planta aos requisitos de controle. Não costuma ser a meta de uma especificação enumerar as possíveis ações no sistema, uma vez que isso, em tese, já faz parte do comportamento da planta. Objetiva-se, por outro lado, cumprir um requisito de controle simplesmente observando eventos na planta e desabilitando aqueles que são considerados proibidos, quando eles puderem ocorrer.

O resultado da ação de uma especificação (ou de um conjunto delas) sobre uma planta remete ao que pode ser informalmente descrito como o comportamento desejado para o sistema sob controle, ou em *malha fechada*. Assim como os modelos de subsistemas, também as especificações podem ser representadas individualmente através de um autômato, expressando os requisitos de controle na forma de desabilitação de eventos em estados. A Figura 3 mostra um exemplo simples de um modelo de especificação para a planta G, apresentada na Figura 2.

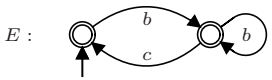


Figura 3 – Modelagem de uma especificação.

Independentemente de relevância prática, a especificação E tem por objetivo controlar a planta G de maneira a evitar o início de operação da máquina M_2 (evento c proibido no estado inicial), sem que

antes M_1 tenha finalizado uma operação (evento b).

3.3 CONTROLE SUPERVISÓRIO

No exemplo anterior, a composição $E \| G$ modela a ação de controle de E sobre G. Logo, é natural pensar em utilizar essa composição para implementar um controlador para G. Na prática, porém, uma planta pode conter eventos que não podem ser diretamente desabilitados. O final de operação de uma máquina ou a quebra de um equipamento, por exemplo, caracterizam eventos cuja ocorrência é involuntária e independe de qualquer política de controle. Logo, ao modelar tais eventos é necessário levar em conta esses aspectos, sob pena de ocasionar uma inconsistência semântica entre o sistema e sua solução de controle.

Nesse sentido, a TCS define uma estrutura de controle para a planta G, dada pelo particionamento de seu conjunto de eventos $\Sigma = \Sigma_c \cup \Sigma_u$, tal que Σ_c é o conjunto de *eventos controláveis*, cuja ocorrência pode ser evitada na planta, e Σ_u é o conjunto contendo os *eventos não-controláveis*, os quais não podem ser diretamente desabilitados.

O elemento da TCS que efetivamente implementa as ações de controle na planta é denominado *supervisor*. Um supervisor é um mapa $S : L(G) \rightarrow 2^{\Sigma}$, associado a uma linguagem $L_S \subseteq L^{\omega}(G)$ que, após qualquer cadeia $s \in L(G)$, observa eventos possíveis na planta e informa, dentre eles, quais devem ser, de fato, habilitados. Nesta tese, S é considerado ser um tipo especial de supervisor, denominado *supervisor marcador* (RAMADGE; WONHAM, 1987b). A ação de controle de um supervisor marcador S sobre G, denotada por S/G, além de habilitar eventos do conjunto $S(s) \subseteq \Sigma$, também marca cadeias $s \in L_S$. A estrutura de supervisão que interage com a planta de forma a fechar a malha de controle, é ilustrada na Figura 4.

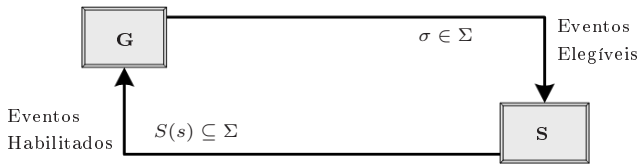


Figura 4 – Fluxo de Controle

Assume-se que eventos $\sigma \in \Sigma$ ocorrem espontaneamente na planta. Quando σ é observado, após uma cadeia qualquer $s \in L(G)$,

o supervisor S atualiza o conjunto $S(s)$ de eventos habilitados. Os eventos possíveis na planta que não pertencem a $S(s)$ são justamente aqueles que estão sendo inibidos pela ação de controle.

O conjunto de cadeias de $L(G)$ que sobrevive sob controle representa o *comportamento gerado em malha fechada*, e é dado pela linguagem $L(S/G)$. O *comportamento marcado em malha fechada*, por sua vez, é dado por $L^{\omega}(S/G) = L(S/G) \cap L_{\mathrm{S}}$. S é chamado *não-bloqueante* quando $L(S/G) = \overline{L^{\omega}(S/G)}$. Finalmente, o problema de controle pode ser enunciado, como segue.

Problema 1 - Problema de Controle Supervisório (PCS) (RAMADGE; WONHAM, 1989): *Dada uma planta G, com eventos em Σ, e uma especificação $E \subseteq \Sigma^*$, definindo um comportamento desejado $K = E \cap L^{\omega}(G)$, encontre um supervisor não-bloqueante S tal que $L^{\omega}(S/G) \subseteq K$.*

Encontrar uma solução para o PCS passa, sobretudo, pelo conceito de *controlabilidade*. Uma linguagem $K \subseteq \Sigma^*$ é dita ser *controlável* em relação a L quando

$$\overline{K}\Sigma_{\mathrm{u}} \cap L \subseteq \overline{K}.$$

Ou seja, após qualquer prefixo de K, se um evento não-controlável é observado em L, a cadeia resultante continua sendo um prefixo de K. Garante-se, então, que nenhum evento não-controlável, possível em L, esteja sendo desabilitado pela ação de controle.

A controlabilidade de uma linguagem não vazia $K \subseteq L^{\omega}(G)$ é uma condição necessária e suficiente para a existência de um supervisor marcador não-bloqueante S, tal que $L^{\omega}(S/G) = K$. Nesse caso, S, tal que $L^{\omega}(S/G) = K$, pode ser implementado por um autômato V, tal que $K = L^{\omega}(V) \cap L^{\omega}(G)$ e $\overline{K} = L(V) \cap L(G)$. A ação de controle de S é implementada pela desabilitação de eventos possíveis em $L(G)$, os quais não são possíveis em $L(V)$, após uma cadeia $s \in L(S/G)$. A função de marcação corresponde a marcar cadeias que são marcadas em ambos V e G.

Seja $\mathcal{C}(K, G) = \{L \subseteq K \mid L \text{ é controlável em relação a } L(G)\}$. $\mathcal{C}(K, G)$ possui um elemento supremo único, denotado por $\sup\mathcal{C}(K, G)$, o qual representa o comportamento menos restritivo possível de ser implementado por um supervisor não-bloqueante S sobre G. Então S, tal que $L^{\omega}(S/G) = \sup\mathcal{C}(K, G)$ e $L(S/G) = \overline{\sup\mathcal{C}(K, G)}$, é uma solução ótima para o PCS.

Quando um problema de controle é resolvido através de uma única estrutura de supervisão, como no PCS, diz-se que a solução é

monolítica. A computação de $\sup\mathcal{C}(K, G)$ é a operação mais complexa do PCS, conforme discutido a seguir.

3.3.1 Complexidade da síntese monolítica

A complexidade computacional, envolvida em uma operação monolítica de síntese, é polinomial no número de estados de G e E (RAMADGE; WONHAM, 1987b). Por conseguinte, tais operações são consideradas computacionalmente tratáveis, já que uma resposta pode ser obtida em um tempo que é limitado por uma função polinomial.

Na prática, porém, o grande empecilho computacional da TCS reside no fato de que a estrutura $G\|E$ pode apresentar um aumento do número de estados de ordem exponencial. Por exemplo, seja um SED modelado por G e restrito por E, compostos como segue.

$$G = \Big\|_{j=1}^{p} G^j \qquad\qquad E = \Big\|_{i=1}^{e} E^i$$

Considerando-se que cada G^j tem no máximo m estados, então G possui m^p estados, no pior caso. Ainda, se cada E^i possui no máximo n estados, então E pode possuir até n^e estados. Logo, a operação monolítica de síntese possui complexidade na ordem de $\mathcal{O}[(n^e.m^p)^2]$.

Portanto, o cálculo de um supervisor monolítico, embora seja polinomial no número de estados de $G\|E$, é exponencial na quantidade de modelos envolvidos no problema. Isso significa que, embora uma operação de síntese seja computável, ela pode não ser tratável na prática. A solução para muitos problemas de controle é inviabilizada por esse entrave computacional, principalmente para aqueles problemas que requerem uma solução centralizada, mas que envolvem múltiplos subsistemas e especificações, o que é típico em SEDs de grande porte.

Na sequência, apresenta-se um exemplo que ilustra a obtenção de uma solução monolítica de controle para um SED, através do qual a complexidade das operações de síntese pode ser dimensionada.

3.4 UM EXEMPLO DE UM SISTEMA DE MANUFATURA

Considere o sistema de manufatura com retrabalho, mostrado na Figura 20.

Nesse exemplo, um robô (R) alimenta com materiais duas má-

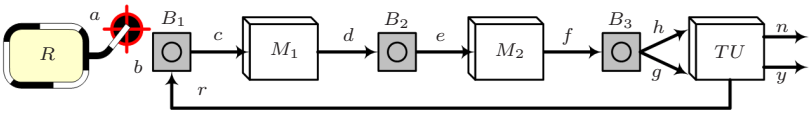


Figura 5 – Pequena fábrica com retrabalho de materiais.

quinas, M_1 e M_2, que produzem peças para serem testadas em uma terceira estação de trabalho, TU. A partir do teste, as peças podem ser aprovadas (evento y), descartadas (evento n) ou retrabalhadas em M_1 (evento r). A interconexão entre os subsistemas se dá por meio de armazéns (*buffers*) intermediários, com capacidade unitária. A Figura 21 apresenta os autômatos G^1, G^2, G^3 e G^4, que modelam, respectivamente, os subsistemas R, M_1, M_2 e TU.

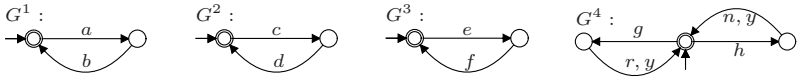


Figura 6 – Modelos para os subsistemas R, M_1, M_2 e TU.

Os modelos G^1, G^2 e G^3 contemplam apenas eventos de *início* e de *fim* de operações em cada respectivo equipamento. Em G^4, ambos os eventos, g e h, são habilitados no estado inicial, modelando que TU pode proceder a duas operações distintas: (i) um teste geral (evento g), que evita um descarte, levando à aprovação ou ao retrabalho das peças em M_2 (respectivos eventos y ou r); ou (ii) um teste final (evento h), que evita o retrabalho e leva as peças a deixarem a linha de manufatura, sendo aprovadas (evento y) ou rejeitadas (evento n).

A planta do sistema é modelada pela composição

$$G = G^1 \| G^2 \| G^3 \| G^4$$

com $\Sigma = \{a, b, c, d, e, f, g, h, r, y, n\}$, onde é assumido que $\Sigma_c = \{a, c, e, g, h\}$ e $\Sigma_u = \{b, d, f, n, y, r\}$.

Como objetivo de controle, considera-se evitar *overflow* e *underflow* nos *buffers* B_1, B_2 e B_3. Para isso, são modeladas, respectivamente, as especificações[1] E_1, E_2 e E_3, conforme apresentadas na Figura 7.

[1] Por clareza, foram removidos das especificações os autolaços contendo os eventos que não são necessários para a modelagem, os quais são assumidos estarem sempre habilitados. Para os resultados que seguem, todavia, considera-se a linguagem da

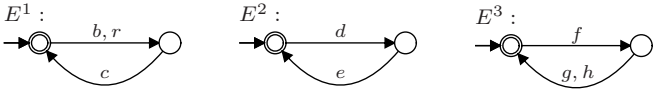


Figura 7 – Modelos para as especificações E_1, E_2 e E_3.

Por exemplo, após a inserção de uma peça no *buffer* B_1 (eventos b ou r) o modelo E_1 evita uma nova inserção (*overflow*) até que a peça seja removida do *buffer* (evento c). Adicionalmente, o modelo evita a retirada de uma peça (*underflow*) sem que antes esta tenha sido depositada no *buffer* (evento c no estado inicial). As demais especificações são semanticamente equivalentes.

Enquanto modelar especificações como E^1, E^2 e E^3 é uma tarefa bastante simples, estruturar autômatos para outros tipos de especificações pode ser substancialmente mais complexo. A seguir, propõe-se uma especificação complementar para esse mesmo exemplo, cujo esforço de modelagem destoa daquele observado nos modelos da Figura 7.

3.4.1 Uma nova especificação de controle

Antes de obter um supervisor para a planta do exemplo, considere expandir os objetivos de controle para envolver uma nova especificação. Suponha que E^4 é uma especificação destinada a controlar a quantidade de ciclos de trabalho a que cada peça é submetida. Assim sendo, E^4 deve: (i) evitar a rejeição de uma peça sem que não tenha sido retrabalhada ao menos uma vez e (ii) evitar que uma mesma peça seja retrabalhada mais de uma vez.

Obter um modelo para E^4 requer, sobretudo, que sejam desabilitados os eventos g ou h após a ocorrência de um evento f. A complexidade inerente a essa decisão está no fato de que escolher entre a desabilitação de g ou h depende do número de vezes em que f ocorre para uma mesma peça, i.e., h é desabilitado após a primeira ocorrência de f e g é desabilitado no caso contrário. Como f ocorre no contexto de um ciclo de execução do sistema, o qual permite o processamento de mais de uma peça em paralelo, então "contar" as vezes em que f ocorre para uma mesma peça requer enumerar suas ocorrências, explicitando todas as combinações possíveis de eventos no ciclo, até que uma decisão

especificação no contexto de Σ^*.

de controle sobre g ou h possa ser tomada.

Um modelo para E^4 pode ser obtido através da estruturação de um autômato com 32 estados. Por razões de clareza, tal autômato não é mostrado, mas a sistemática subjacente a essa construção é explicada no contexto do modelo parcial de E^4, apresentado na Figura 8.

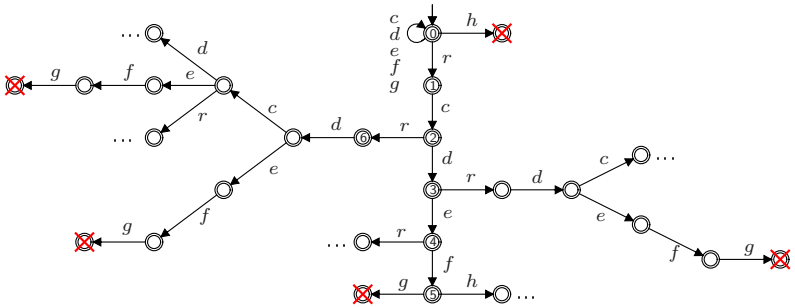


Figura 8 – Modelo parcial para a especificação E^4, com eventos em Σ.

O autômato da Figura 8 apresenta o núcleo da modelagem de E^4. Parte-se da primeira ocorrência de f, no estado inicial, onde então o evento h é desabilitado. Nesse estado, perante a ocorrência de um primeiro retrabalho (evento r), desencadeia-se uma estruturação lógica no autômato, de modo que se possa rastrear todas as combinações possíveis de eventos no sistema até que, para cada combinação, se possa desabilitar o evento g.

Por exemplo, a partir do estado inicial, após r (estado 1), o *buffer* B_1 contém uma peça, cuja retirada é modelada pelo evento c (estado 2). Caso o sistema processe uma única peça, então a única sequência admitida é a que leva aos estados 3, 4 e 5 onde, por fim, proíbe-se o evento g. Como o sistema pode processar mais de uma peça em paralelo, tão logo B_1 seja desocupado (estado 2), um novo evento r (estado 6) é admitido (para uma nova peça). Esse evento acaba por criar um novo fluxo, representando um novo contexto possível de execução que, por sua vez, leva a novos fluxos, e assim por diante. Todos eles precisam ser combinados até que uma segunda ocorrência de f seja observada em quaisquer dos fluxos e, assim, um evento g possa ser proibido.

Usando o modelo completo de E^4, com 32 estados, e os demais modelos de especificações e plantas apresentados anteriormente, resolve-se o PCS para o exemplo proposto. O número de estados dos autômatos, envolvidos na síntese monolítica de um supervisor ótimo

para o exemplo, é apresentado na Tabela 1.

Tabela 1 – Síntese Monolítica de Controle.

Solução para o PCS				
Modelo	$G = \|\|_{i=1}^{4} G^i$	$E = \|\|_{i=1}^{4} E^i$	$K = G\|E$	V
N°de Estados	24	108	1.394	121

V é um autômato para o supervisor $\sup\mathcal{C}(K, G)$ e pode ser usado para implementar uma solução ótima para o PCS.

3.4.2 Um problema motivador

Embora não seja uma tarefa trivial, estruturar um modelo para a especificação E^4, com 32 estados, pode ser considerada uma tarefa manualmente realizável. Esse problema, porém, pode se tornar substancialmente mais complexo se, por exemplo, forem considerados mais ciclos de retrabalho para cada peça ou novos subsistemas forem inseridos dentro desse ciclo.

Suponha, por exemplo, que sejam admitidos agora até 2 ciclos de retrabalho para cada peça. Pode ser mostrado (usando métodos a serem propostos nesta tese) que o modelo para essa especificação requer a estruturação de um autômato com 560 estados. Um aumento para 3 ciclos eleva esse indicador para 15.803 estados. E assim por diante. Isso fornece uma estimativa do esforço que uma tarefa de modelagem como essa pode requerer, ao passo que sua ocorrência prática é bastante razoável.

É sobre essas conjecturas que a proposta desta tese se insere e é distribuída nos capítulos seguintes. A primeira grande meta é prover alternativas para simplificar a modelagem de especificações; adicionalmente, objetiva-se lidar com a complexidade computacional requerida para obter uma solução de controle. Propostas serão apresentadas nesse sentido. Por fim, apresenta-se uma alternativa para modularizar o procedimento de síntese de supervisores, combinando o uso das propostas supracitadas.

3.5 SUMÁRIO

Neste capítulo foi apresentada a Teoria do Controle Supervisório e como obter uma solução monolítica para o Problema de Controle

Supervisório. Os conceitos apresentados foram exemplificados através de um estudo de caso, que permitiu dimensionar o esforço, manual e computacional, requerido para solucionar o problema de controle.

Por fim, foi introduzido um exemplo motivador para evidenciar lacunas existentes na TCS, as quais podem limitar sua adoção prática. Tais inconveniências contextualizam as contribuições desta tese, apresentadas nos capítulos seguintes.

4 TCS COM DISTINGUIDORES

Este capítulo apresenta o conceito de um *Distinguidor*. Distinguidores foram originalmente pensados como tipos especiais de sensores que, ao serem associados ao modelo de um SED, permitem refinar informações sobre certos eventos do sistema (BOUZON; QUEIROZ; CURY, 2008, 2009). Esta tese preserva a ideia central de um distinguidor, em termos de refinamento de eventos, mas a apresenta com uma perspectiva distinta, que estende e melhora a abordagem inicial sob diversas esferas, principalmente em termos de construção, de resultados e de exemplos.

A abordagem com distinguidores, como proposta nesta tese, consiste basicamente das seguintes etapas. Primeiramente, os eventos a serem refinados são selecionados dentre os demais, em razão da dificuldade para modelar alguma especificação de controle. Em seguida, cada evento escolhido é mapeado em um conjunto de refinamentos (novos eventos). Normalmente, cada refinamento renomeia o evento original de modo que se possa identificar uma instância particular em que ele ocorre no sistema. Assim, fazendo-se uso de um alfabeto mais rico, pode-se facilmente expressar a semântica de uma especificação, evitando que, por vezes, extensas sequências de eventos tenham de ser estruturadas. Além disso, mostra-se que os modelos refinados podem ser usados para a síntese de supervisores, sem que isso implique no aumento da complexidade computacional requerida para tratar do problema de controle, em relação ao método sem refinamentos. Por fim, ainda mostra-se como distinguidores podem ser usados para a construção de aproximações para os modelos refinados, as quais permitem sintetizar uma solução para o PCS com ganhos computacionais.

Os resultados a serem apresentados neste capítulo permitem determinar a controlabilidade, a máxima permissividade e o não-bloqueio de uma solução de controle obtida via distinguidores, bem como sua relação com a solução original, não refinada. Por fim, retoma-se o exemplo do sistema de manufatura, a fim de ilustrar os benefícios da abordagem.

4.1 UMA INTRODUÇÃO AOS DISTINGUIDORES

Considere um SED, modelado com eventos em Σ. Na abordagem com distinguidores, assume-se que cada evento $\sigma \in \Sigma$ passa a represen-

tar uma máscara para eventos de um conjunto $\Delta^\sigma \neq \emptyset$. Eventos em Δ^σ são escolhidos de maneira a identificar diferentes instâncias em que σ pode ocorrer no sistema.

Para as máscaras $\sigma_1, \sigma_2 \in \Sigma$, tal que $\sigma_1 \neq \sigma_2$, considera-se que $\Delta^{\sigma_1} \cap \Delta^{\sigma_2} = \emptyset$. Ou seja, cada máscara possui um conjunto particular de refinamentos, distinto dos demais, e não vazio. Então, Σ passa a ser um conjunto de máscaras para eventos do alfabeto refinado $\Delta = \dot{\bigcup}_{\sigma \in \Sigma} \Delta^\sigma$. A relação entre os alfabetos Σ e Δ é estabelecida por um mapa, definido como segue.

Definição 1 (Mapa Mascarador) *Um mapa mascarador* $\Pi : \Delta^* \to \Sigma^*$ *é definido recursivamente por*

$$\begin{aligned} \Pi(\epsilon) &= \epsilon, \\ \Pi(t\delta) &= \Pi(t)\sigma, \text{ para } t \in \Delta^*, \ \delta \in \Delta^\sigma \ \text{ e } \sigma \in \Sigma. \end{aligned}$$

Ou seja, Π é um mapa que atribui, a cada cadeia de eventos refinados, a cadeia original (no alfabeto Σ). Esse mapa pode ser estendido para qualquer linguagem $L_d \subseteq \Delta^*$ por

$$\Pi(L_d) = \{s \in \Sigma^* \,|\, \exists t \in L_d, \, \Pi(t) = s\}.$$

O mapa mascarador é um caso particular do *mapa repórter* (ZHONG; WONHAM, 1990), introduzido no contexto do controle hierárquico para comutar os níveis de *processo* (baixo nível) com o de *gerência* (alto nível). O mapa Π tem como particularidade o fato de que um evento em alto nível (evento original) é sempre a máscara de eventos em baixo nível (refinamentos). Ainda, todo refinamento é reportado para o alto nível, ou seja, não existem transições *silenciosas*.

Enquanto mapa repórter, Π é prefixo-preservante, ou seja, se $s \leq t$, então $\Pi(s) \leq \Pi(t)$. Além disso, Π comuta com o prefixo-fechamento (WONG; WONHAM, 1996), ou seja, para $L_d \subseteq \Delta^*$, $\overline{\Pi(L_d)} = \Pi(\overline{L_d})$.

Definição 2 (Mapa Mascarador Inverso) *O mapa mascarador inverso* $\Pi^{-1} : \Sigma^* \to 2^{\Delta^*}$ *é definido por*

$$\Pi^{-1}(s) = \{t \in \Delta^* \mid \Pi(t) = s\}.$$

Esse mapa também pode ser estendido para qualquer linguagem $L \subseteq \Sigma^*$ por

$$\Pi^{-1}(L) = \{t \in \Delta^* \,|\, \Pi(t) \in L\}.$$

Definida a relação entre os alfabetos Σ e Δ, um distinguidor pode ser formalmente apresentado como segue.

Definição 3 (Distinguidor) *Dado um alfabeto Σ e um alfabeto refinado $\Delta = \dot{\bigcup}_{\sigma\in\Sigma}\Delta^\sigma$, um mapa $\mathrm{D} : \Sigma^* \to 2^{\Delta^*}$ é um distinguidor se $\forall s \in \Sigma^*$, $\sigma \in \Sigma$, $r \in \Delta^*$ e $\delta \in \Delta$*

$$\begin{array}{lcl} i)\ \mathrm{D}(\epsilon) & = & \{\epsilon\}; \\ ii)\ r\delta \in \mathrm{D}(s\sigma) & \Rightarrow & r \in \mathrm{D}(s) \wedge \delta \in \Delta^\sigma; \\ iii)\ r \in \mathrm{D}(s) & \Rightarrow & \exists \delta' \in \Delta^\sigma, r\delta' \in \mathrm{D}(s\sigma). \end{array}$$

Em prosa, um distinguidor D é um mapa prefixo-preservante que substitui cada cadeia em Σ^*, por um conjunto não vazio de cadeias em Δ^*. As cadeias em Δ^* representam instâncias das cadeias originais, no alfabeto refinado. Na prática, tais cadeias podem ser construídas pela substituição de cada máscara que compõe uma cadeia em Σ^* por refinamentos daquela máscara. Note que o mapa Π^{-1} é um caso particular de um distinguidor e $\mathrm{D}(s) \subseteq \Pi^{-1}(s)$, $\forall s \in \Sigma^*$.

Um distinguidor pode ser estendido para qualquer linguagem $L \subseteq \Sigma^*$, por

$$\mathrm{D}(L) = \{r \in \Delta^* \mid \exists s \in L,\, r \in \mathrm{D}(s)\}.$$

A partir da Definição 3, as seguintes proposições podem ser enunciadas.

Proposição 1 *Para um distinguidor $\mathrm{D} : \Sigma^* \to 2^{\Delta^*}$, com um mapa mascarador $\Pi : \Delta^* \to \Sigma^*$ e qualquer linguagem $L \subseteq \Sigma^*$, é verdade que $\Pi(\mathrm{D}(L)) = L$.*

Prova: Como $\mathrm{D}(L) \subseteq \Pi^{-1}(L)$, então $\Pi(\mathrm{D}(L)) \subseteq \Pi(\Pi^{-1}(L)) = L$. Agora, seja $s \in L \subseteq \Sigma^*$ e $t \in \mathrm{D}(s) \subseteq \Pi^{-1}(s)$. Então $t \in \mathrm{D}(L)$ e $s = \Pi(t) \in \Pi(\mathrm{D}(L))$. Logo, $\Pi(\mathrm{D}(L)) = L$. ∎

Proposição 2 *Para um distinguidor $\mathrm{D} : \Sigma^* \to 2^{\Delta^*}$ e qualquer linguagem $L \subseteq \Sigma^*$, é verdade que $\overline{\mathrm{D}(L)} = \mathrm{D}(\overline{L})$.*

Prova: Seja $r \in \overline{\mathrm{D}(L)}$. Existe $u \in \Delta^*$ e $t \in L$ tal que $ru \in \mathrm{D}(t)$. Então, os itens (i) e (ii) da Definição 3 implicam, por indução no tamanho da cadeia r, que existe $s \in \Sigma^*$, com $s \leq t$, tal que $r \in \mathrm{D}(s)$. Como $s \in \overline{L}$, então $r \in \mathrm{D}(\overline{L})$. Agora, seja $r \in \mathrm{D}(\overline{L})$. Existe $s, t \in \Sigma^*$ tal que $st \in L$ e $r \in \mathrm{D}(s)$. Então, pelo item (iii) da Definição 3, tem-se, por indução no tamanho de r, que existe $u \in \Delta^*$ tal que $ru \in \mathrm{D}(st) \subseteq \mathrm{D}(L)$. Logo, $r \in \overline{\mathrm{D}(L)}$. ∎

Uma linguagem, em particular, que pode ser associada com o uso de um distinguidor é L_{D}, definida como segue.

Definição 4 *Para* D*, como na Definição 3,* $L_{\mathrm{D}} \subseteq \Delta^*$ *é uma linguagem distinguidora tal que* $L_{\mathrm{D}} = \mathrm{D}(\Sigma^*)$.

Para um distinguidor D, em particular, $L_{\mathrm{D}} = \overline{L_{\mathrm{D}}} \subseteq \Delta^*$ é uma linguagem prefixo-fechada que contempla todas as possíveis distinções de Σ^*. L_{D} habilita ao menos uma cadeia, em Δ^*, para cada cadeia original, em Σ^*. Note que $\Pi(L_{\mathrm{D}}) = \Sigma^*$ se segue diretamente da Proposição 1.

A partir das definições apresentadas, o efeito de um distinguidor D sobre uma linguagem pode ser reescrito como segue.

Proposição 3 *Seja um distinguidor* D*, como na Definição 3, e uma linguagem* L_{D}*, como na Definição 4. Para uma linguagem* $L \subseteq \Sigma^*$*, o efeito de* D *sobre* L *é tal que*

$$\mathrm{D}(L) = \Pi^{-1}(L) \cap L_{\mathrm{D}}.$$

Prova: Seja $r \in \mathrm{D}(L)$. É verdade que $r \in \Pi^{-1}(L)$ pois, por definição, $\mathrm{D}(L) \subseteq \Pi^{-1}(L)$. Além do mais, existe um $s \in L$, tal que $r \in \mathrm{D}(s)$, o que implica que $r \in L_{\mathrm{D}}$, uma vez que $L_{\mathrm{D}} = \mathrm{D}(\Sigma^*)$ e $s \in L \subseteq \Sigma^*$. Portanto, $\mathrm{D}(L) \subseteq \Pi^{-1}(L) \cap L_{\mathrm{D}}$. Agora, seja $r \in \Pi^{-1}(L) \cap L_{\mathrm{D}}$. De $r \in L_{\mathrm{D}} = \mathrm{D}(\Sigma^*)$, tem-se que $r \in \mathrm{D}(s)$ e $\Pi(r) = s$, para algum $s \in \Sigma^*$. Mas, como também $r \in \Pi^{-1}(L)$, então $\Pi(r) = s \in L \subseteq \Sigma^*$. Assim, como $r \in \mathrm{D}(s)$ e $s \in L$, tem-se que $r \in \mathrm{D}(L)$ e $\mathrm{D}(L) \supset \Pi^{-1}(L) \cap L_{\mathrm{D}}$. ■

A Proposição 3 implica que um distinguidor D pode ser identificado por uma linguagem L_{D}. Como L_{D} é prefixo-fechada (pela Definição 4) e o mapa Π^{-1} é um caso particular de um distinguidor D (pela Definição 3), então (pela Proposição 2) L_{D} e $\Pi^{-1}(L)$ são não-conflitantes, para qualquer linguagem $L \subseteq \Sigma^*$. Essa propriedade é formalizada pela proposição seguinte.

Proposição 4 *Para* L_{D}*, como na Definição 4, e* Π^{-1}*, como na Definição 2, é verdade que, para qualquer linguagem* $L \subseteq \Sigma^*$*,* $\overline{\Pi^{-1}(L) \cap L_{\mathrm{D}}} = \overline{\Pi^{-1}(L)} \cap \overline{L_{\mathrm{D}}}$.

Prova: $\overline{\Pi^{-1}(L) \cap L_{\mathrm{D}}} = \overline{\mathrm{D}(L)} = \mathrm{D}(\overline{L}) = \Pi^{-1}(\overline{L}) \cap L_{\mathrm{D}} = \overline{\Pi^{-1}(L)} \cap \overline{L_{\mathrm{D}}}$. ■

Considerando-se um SED modelado por um autômato G, então $\mathrm{D}(L(G)) = \Pi^{-1}(L(G)) \cap L_{\mathrm{D}}$ e $\mathrm{D}(L^\omega(G)) = \Pi^{-1}(L^\omega(G)) \cap L_{\mathrm{D}}$ repre-

sentam o efeito de D sobre G. Ou seja, o distinguidor refina cadeias de $L(G)$ em cadeias em Δ^*, através do mapa inverso $\Pi^{-1}(L(G))$, e distingue a ocorrência dessas cadeias de acordo com a linguagem distinguidora L_{D}.

Note que o efeito do mapa inverso Π^{-1} pode ser representado em autômatos, simplesmente substituindo os eventos de cada transição pelos conjuntos correspondentes de eventos refinados. Equivalentemente, o efeito do mapa Π também pode ser representado, desta vez substituindo-se os eventos refinados pelas respectivas máscaras. Um guia para a construção de um distinguidor D é apresentado na Subseção 4.1.2 e um exemplo de como estruturar um autômato H_{D} que modela D, de tal modo que $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$, é apresentado na Subseção 4.2.2.

Uma estrutura que ilustra a introdução de um distinguidor em um sistema modelado por G é apresentada na Figura 9. Nesse esquema, a interação entre o distinguidor e a planta é contextualizada em termos de fluxo de eventos, observados através de uma sucessão de estágios.

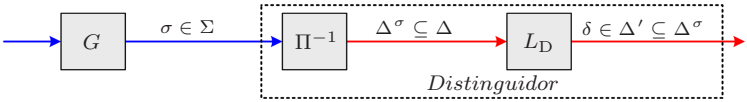


Figura 9 – Fluxo de eventos ao usar um distinguidor na planta.

Assume-se que eventos $\sigma \in \Sigma$ ocorrem espontaneamente em G. Então, cada evento σ é mapeado em um subconjunto Δ^σ de refinamentos, através do mapa inverso Π^{-1}, os quais são filtrados pela linguagem distinguidora L_{D}, estabelecendo quais deles são de fato elegíveis na planta.

4.1.1 Classes de distinguidores

Distinguidores podem ser classificados sob duas diferentes perspectivas, como segue.

Definição 5 (Distinguidor Preditível) *Um distinguidor* $\mathrm{D} : \Sigma^* \to 2^{\Delta^*}$ *é preditível se, para todo* $s \in \Sigma^*$, $|\mathrm{D}(s)| = 1$. *Do contrário,* D *é dito ser não-preditível.*

Em prosa, D é preditível se qualquer cadeia em Σ^* é mapeada em um conjunto com exatamente uma cadeia em Δ^*, i.e., se após cada

cadeia $r \in L_{\mathrm{D}}$ existe um, e apenas um, evento refinado elegível para cada máscara $\sigma \in \Sigma$. Assim, a linguagem L_{D} permite sempre *predizer* a próxima distinção de qualquer máscara. Os exemplos apresentados na sequência ilustram essas duas classes de distinguidores.

4.1.1.1 Exemplos simples de distinguidores

Seja G o modelo da planta de um dispositivo que retira peças de um *buffer* B. Seja $\Sigma = \{\alpha\}$, tal que α representa o evento *"máquina removeu uma peça de B"*. O comportamento do sistema é, então, dado por $L(G) = \Sigma^*$. Em seguida, mostram-se dois exemplos de distinguidores para esse caso particular, um preditível e um não-preditível.

Distinguidor preditível: Considere modelar um distinguidor para identificar a remoção de peças do *buffer*, de maneira a sinalizar toda a vez que um número *par* de peças foi retirado. Então, pode-se definir $\Delta = \Delta^{\alpha} = \{\alpha_P, \alpha_I\}$ a fim de refinar a máscara α nos seguintes eventos:

- α_P, *"máquina removeu um número* **par** *de peças de B"*; e
- α_I, *"máquina removeu um número* **ímpar** *de peças de B"*.

Agora, seja a linguagem distinguidora dada por $L_{\mathrm{D}} = (\alpha_I\alpha_P)^*(\epsilon + \alpha_I)$. Nesse caso, D é claramente *preditível*, uma vez que para qualquer $s \in \Sigma^*$, $|\mathrm{D}(s)| = 1$, i.e., qualquer cadeia em Σ^* possui exatamente uma respectiva cadeia em Δ^*.

Por exemplo, para uma cadeia $s = \alpha\alpha$, D(s) é tal que

$$\mathrm{D}(s) = \{\alpha_I\alpha_P\}.$$

Assim, após a cadeia $\alpha\alpha$, é possível *predizer* que a próxima ocorrência de α será α_I.

Distinguidor não-preditível: Suponha agora que, ao invés de identificar a remoção de um número par de peças do *buffer* B, um distinguidor é usado para identificar as cores das peças removidas. Assuma que existam duas cores aleatoriamente possíveis: *vermelho* e *azul*.

Sendo assim, a ocorrência do evento α é refinada de maneira tal que $\Delta = \Delta^{\alpha} = \{\alpha_V, \alpha_A\}$. Nesse caso, os refinamentos possuem a seguinte semântica:

- α_V, *"máquina removeu uma peça* **vermelha** *de B"*; e

- α_A, *"máquina removeu uma peça* **azul** *de B".*

Para esse exemplo, considere que a linguagem do distinguidor é dada por $L_{\mathrm{D}} = (\alpha_B + \alpha_W)^*$. Esse distinguidor é claramente *não-preditível*, pois existe um $s \in \Sigma^*$ tal que $|\mathrm{D}(s)| > 1$. Por exemplo, para $s = \alpha\alpha$, D(s) é tal que

$$\mathrm{D}(s) = \{\alpha_V\alpha_V, \alpha_V\alpha_A, \alpha_A\alpha_V, \alpha_A\alpha_A\}.$$

4.1.2 Construção de um distinguidor

Vem sendo argumentado, ao longo desta tese, que, para muitos problemas práticos de controle, pode ser mais simples modelar especificações usando eventos de um alfabeto refinado Δ do que eventos diretamente observáveis na planta, os quais compõem Σ. Essa ideia é, então, formalizada através do conceito de distinguidores.

Contudo, a construção de um distinguidor pressupõe algumas etapas de modelagem, as quais precisam ser detalhadas. É o caso, por exemplo, da identificação dos eventos a serem refinados, como eles são refinados e como atribuir uma semântica a cada refinamento. Essas tarefas explicitam, de modo prático, o formalismo sobre o qual um distinguidor é fundado.

Ocorre que a construção de um distinguidor envolve, por natureza, tarefas de engenharia. Como tais, elas dependem, sobretudo, da capacidade humana para serem realizadas, de modo que se torna inconveniente sistematizá-las por completo. Ainda assim, alguns passos dessa construção podem ser evidenciados. Apresenta-se, na sequência, o que poderia ser usado como um guia para obter um distinguidor.

(i) *Identificação de um conjunto inicial de eventos $\sigma \in \Sigma$ a serem refinados:* o primeiro passo para a construção de um distinguidor é identificar quais eventos em Σ, quando refinados, poderiam simplificar a modelagem de uma dada especificação de controle.

(ii) *Definição das instâncias de refinamentos $\delta \in \Delta^\sigma$:* a partir do passo (i), deve-se determinar quais instâncias devem ser associadas a cada um dos eventos selecionados para o refinamento. Tais instâncias modelam diferentes circunstâncias em que o evento original pode ocorrer. Assim, cada instância também carrega uma semântica em particular, que precisa ser implementada pelo modelo distinguidor (passo (iv)).

(iii) *Complementação do conjunto* Δ*:* dado o conjunto inicial de eventos a serem refinados e seus respectivos conjuntos de instâncias, a construção de um modelo que distingue a ocorrência das diferentes instâncias de cada conjunto pode depender de outros refinamentos. De fato, o significado de uma certa instância de um evento pode se materializar apenas quando o sentido de outro evento é conhecido. Logo, tal evento também deve ser refinado e suas instâncias distinguidas. Tratar dessas dependências semânticas requer revisitar os passos (i) e (ii) e, assim, corretamente caracterizar o alfabeto refinado Δ.

(iv) *Modelagem do distinguidor* H_{D}*:* finalmente, aproveitando-se os demais passos, constrói-se um modelo que estabelece as interdependências entre as instâncias de refinamentos em Δ. Essa tarefa consiste na estruturação de um autômato H_{D}, tal que $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$, normalmente simples em termos de modelagem e, em geral, pode ser projetado modularmente.

Ao longo desta tese serão mostrados exemplos de como aplicar os passos acima para construir um distinguidor, bem como formas de modelá-lo através de autômatos e métodos de modularização. Na próxima seção, é discutida a relação entre modelos construídos sobre alfabetos refinados e não refinados, no contexto de um problema de controle.

4.2 PCS COM DISTINGUIDORES

Representar o efeito de um distinguidor D sobre o PCS requer considerar dois diferentes aspectos: primeiro, o fato de que o conjunto de eventos Σ é expandido em um conjunto de eventos refinados Δ e, segundo, que as informações trazidas pelo distinguidor precisam ser levadas em conta ao modelar as especificações de controle.

Sob o ponto de vista de modelagem, denota-se por G_{d} (ou $\mathrm{D}(G)$) um autômato tal que $L(G_{\mathrm{d}}) = \mathrm{D}(L(G))$ e $L^{\omega}(G_{\mathrm{d}}) = \mathrm{D}(L^{\omega}(G))$, e por E_{d} uma especificação definida em Δ. E_{d} deve expressar os mesmos requisitos expressos por E no PCS, porém, agora, usando informações adicionais providas pelo distinguidor, i.e., E_{d} é tal que $E_{\mathrm{d}} \cap L^{\omega}(G_{\mathrm{d}}) = K_{\mathrm{d}} = \mathrm{D}(K)$, onde $K = E \cap L^{\omega}(G)$. Nesse caso, a especificação $K \subseteq L^{\omega}(G)$ seria dada por $K = \Pi(E_{\mathrm{d}} \cap \Pi^{-1}(L^{\omega}(G)) \cap L_{\mathrm{D}})$.

Já sob o ponto de vista de controle, embora o refinamento de eventos possa enriquecer as informações providas pela planta, a con-

trolabilidade dos eventos refinados deve permanecer equivalente nesse novo cenário, i.e., se $\sigma \in \Sigma_{\mathrm{u}}$, então $\delta \in \Delta_{\mathrm{u}}$, para todo $\delta \in \Delta^{\sigma}$. Portanto, a estrutura de controle definida sobre o alfabeto refinado é tal que:

$$\Delta = \Delta_{\mathrm{c}} \dot{\cup} \Delta_{\mathrm{u}}$$

para

$$\Delta_{\mathrm{u}} = \bigcup_{\sigma \in \Sigma_{\mathrm{u}}} \Delta^{\sigma} \qquad \text{e} \qquad \Delta_{\mathrm{c}} = \bigcup_{\sigma \in \Sigma_{\mathrm{c}}} \Delta^{\sigma}.$$

Além disso, as ações de controle definidas no PCS original devem ser consistentemente expressas no problema com refinamentos, i.e., sempre que um evento é habilitado em Δ, todos os eventos com a mesma máscara também devem ser habilitados. Por exemplo, se após uma cadeia $s \in L(G_{\mathrm{d}})$ um evento $\delta \in \Delta^{\sigma}$ é habilitado na planta, então todo evento $\delta' \in \Delta^{\sigma}$, tal que $s\delta' \in L(G_{\mathrm{d}})$, também será habilitado. Essa ideia leva à definição da seguinte classe de supervisores.

Definição 6 (Supervisor não-ambíguo) *Um supervisor* $S_{\mathrm{d}} : L(G_{\mathrm{d}}) \rightarrow 2^{\Delta}$ *é não-ambíguo se* $\forall s \in L(G_d)$ *e* $\forall \sigma \in \Pi(S_{\mathrm{d}}(s))$, *tem-se que* $s\Delta^{\sigma} \cap L(G_{\mathrm{d}}) \subseteq sS_{\mathrm{d}}(s)$.

Finalmente, pode-se introduzir um novo problema de controle, correspondendo ao PCS original, modificado para considerar o uso de um distinguidor.

Problema 2 - PCS com Distinguidores (PCS-D): *Dado* G *e* K, *como no PCS, e um distinguidor* D, *seja* G_{d} *tal que* $L(G_{\mathrm{d}}) = \mathrm{D}(L(G))$ *e* $L^{\omega}(G_{\mathrm{d}}) = \mathrm{D}(L^{\omega}(G))$. *Para uma especificação* $E_{\mathrm{d}} \subseteq \Delta^{*}$, *definindo um comportamento esperado* $K_{\mathrm{d}} = E_{\mathrm{d}} \cap L^{\omega}(G_{\mathrm{d}})$, *onde* $K_{\mathrm{d}} = \mathrm{D}(K)$, *encontre um supervisor não-ambíguo e não-bloqueante* $S_{\mathrm{d}} : \Delta^{*} \rightarrow 2^{\Delta}$, *tal que* $L^{\omega}(S_{\mathrm{d}}/G_{\mathrm{d}}) \subseteq K_{\mathrm{d}}$.

No PCS-D, a ideia de que E_{d} deve ser uma especificação expressando os mesmos requisitos que E no PCS é capturada por $K_{\mathrm{d}} = \mathrm{D}(K)$, i.e., K_d corresponde, na verdade, ao modelo K distinguido por D. Na sequência, é assumido que garantir a equivalência de E_{d} em relação a E é uma tarefa de modelagem, tal que se pode considerar o PCS-D bem-definido a partir do problema original, o PCS. A Figura 10 contextualiza a arquitetura do problema de controle supervisório com distinção de eventos.

Quando a ocorrência de um evento δ é observada na planta G_{d}, após uma cadeia qualquer $r \in L(G_{\mathrm{d}})$, o supervisor não-ambíguo S_{d}

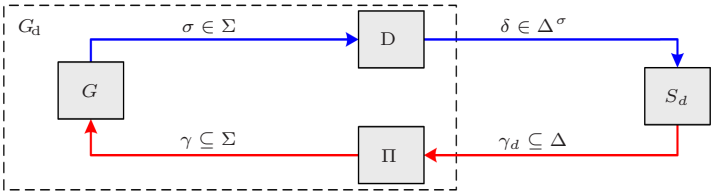


Figura 10 – Arquitetura de Controle Supervisório com supervisores não-ambíguos.

atualiza o conjunto γ_d de eventos cujas máscaras no conjunto γ são permitidas na planta. Observe que se G_{d} é uma planta resultante da distinção de G por um distinguidor preditível, então qualquer supervisor que iniba somente eventos elegíveis em L_{D} é um supervisor não-ambíguo. Como consequência, qualquer supervisor implementado por um autômato V, tal que $L(V) \subseteq L_{\mathrm{D}}$, é um supervisor não-ambíguo.

Os resultados apresentados na sequência evidenciam o fato de que uma solução ótima para o PCS pode ser obtida através da resolução do PCS-D, e vice-versa.

Teorema 1 *Dados G e K, como no PCS, e um distinguidor* D*, seja G_{d} um autômato para G distinguido por* D *e $K_{\mathrm{d}} = \mathrm{D}(K)$, então*

$$\Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) = \sup\mathcal{C}(K, G) \tag{4.1}$$

e

$$\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}) = D(\sup\mathcal{C}(K, G)) \tag{4.2}$$

Os lemas seguintes auxiliam na demonstração do Teorema 1.

Lema 1 *Para G e K, como no PCS, e G_{d} e K_{d}, como no PCS-D, tem-se que*

$$\Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) \in \mathcal{C}(K, G).$$

Prova: Primeiro, observe que

$$\begin{array}{rcl} \sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}) & \subseteq & K_{\mathrm{d}} \\ & \Rightarrow & \\ \Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) & \subseteq & \Pi(K_{\mathrm{d}}) = K. \end{array}$$

Agora, para $\sigma_u \in \Sigma_{\rm u}$, seja s tal que

$$\begin{aligned} s &\in \overline{\Pi(\sup\mathcal{C}(K_{\rm d}, G_{\rm d}))}\sigma_u \cap L(G) \\ &= \Pi(\overline{\sup\mathcal{C}(K_{\rm d}, G_{\rm d})})\sigma_u \cap L(G). \end{aligned}$$

Ainda, seja $t \in \overline{\sup\mathcal{C}(K_{\rm d}, G_{\rm d})}$ tal que $s = \Pi(t)\sigma_u$.
Como $L_{\rm D} = {\rm D}(\Sigma^*)$ e

$$t \in \overline{K_{\rm d}} \quad = \quad \overline{\Pi^{-1}(K) \cap L_{\rm D}} \quad \subseteq \quad L_{\rm D},$$

então $\exists \delta_u \in \Delta^{\sigma_u}$, tal que $t\delta_u \in L_{\rm D}$. Mas,

$$\Pi(t\delta_u) \quad = \quad \Pi(t)\sigma_u \quad = \quad s \in L(G).$$

Portanto,

$$t\delta_u \quad \in \quad \Pi^{-1}(L(G)) \cap L_{\rm D} \quad = \quad L(G_{\rm d}).$$

Pela controlabilidade de $\sup\mathcal{C}(K_{\rm d}, G_{\rm d})$ e.r.a. $L(G_{\rm d})$, tem-se que $t\delta_u \in \overline{\sup\mathcal{C}(K_{\rm d}, G_{\rm d})}$. Logo

$$\begin{aligned} s &= \Pi(t\delta_u) \\ &\in \Pi(\overline{\sup\mathcal{C}(K_{\rm d}, G_{\rm d})}) \\ &= \overline{\Pi(\sup\mathcal{C}(K_{\rm d}, G_{\rm d}))}, \end{aligned}$$

o que é suficiente para provar o Lema 1. ■

Lema 2 *Para G e K, como no PCS, e $G_{\rm d}$ e $K_{\rm d}$, como no PCS-D, também tem-se que*

$${\rm D}(\sup\mathcal{C}(K, G)) \in \mathcal{C}(K_{\rm d}, G_{\rm d}).$$

Prova: Primeiro, observe que

$$\begin{aligned} {\rm D}(\sup\mathcal{C}(K, G)) &= \Pi^{-1}(\sup\mathcal{C}(K, G)) \cap L_{\rm D} \\ &\subseteq \Pi^{-1}(K) \cap L_{\rm D} \\ &= K_{\rm d}. \end{aligned}$$

Então, para $\delta_u \in \Delta_{\mathrm{u}}$, com $\Pi(\delta_u) = \sigma_u$, tem-se

$$
\begin{aligned}
& \{\overline{\mathrm{D}(\mathrm{sup}\mathcal{C}(K,G))}\}\delta_u \cap L(G_{\mathrm{d}}) \\
= \; & \{\mathrm{D}(\overline{\mathrm{sup}\mathcal{C}(K,G)})\}\delta_u \cap L(G_{\mathrm{d}}) \text{ (pela Proposição 2)} \\
= \; & \{\Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)}) \cap L_{\mathrm{D}}\}\delta_u \cap L(G_{\mathrm{d}}) \text{ (pela Proposição 3)} \\
\subseteq \; & \{\Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)})\}\delta_u \cap L(G_{\mathrm{d}}) \\
\subseteq \; & \Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)})\Pi^{-1}(\sigma_u) \cap L(G_{\mathrm{d}}) \\
= \; & \Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)\sigma_u}) \cap L(G_{\mathrm{d}}) \\
= \; & \Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)\sigma_u}) \cap \Pi^{-1}(L(G)) \cap L_{\mathrm{D}} \\
= \; & \Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)\sigma_u} \cap L(G)) \cap L_{\mathrm{D}} \\
\subseteq \; & \Pi^{-1}(\overline{\mathrm{sup}\mathcal{C}(K,G)}) \cap L_{\mathrm{D}} \text{ (pela controlabilidade de } \mathrm{sup}\mathcal{C}(K,G)) \\
= \; & \mathrm{D}(\overline{\mathrm{sup}\mathcal{C}(K,G)}) \text{ (pela Proposição 3)} \\
= \; & \overline{\mathrm{D}(\mathrm{sup}\mathcal{C}(K,G))} \text{ (pela Proposição 2).}
\end{aligned}
$$

■

Lema 3 *Para um distinguidor* $\mathrm{D} : \Sigma^* \to 2^{\Delta^*}$*, um mapa mascarador* $\Pi : \Delta^* \to \Sigma^*$ *e uma linguagem qualquer* $L_d \subseteq L_{\mathrm{D}} \subseteq \Delta^*$*, é verdade que* $L_d \subseteq \mathrm{D}(\Pi(L_d))$.

Prova: Seja $t \in L_d \subseteq L_{\mathrm{D}}$ e $s = \Pi(t)$. Então, $t \in \Pi^{-1}(s) \cap L_{\mathrm{D}} = \mathrm{D}(s) \subseteq \mathrm{D}(\Pi(L_d))$. ■

Prova: Agora prova-se o Teorema 1.
(4.1) Do Lema 1, tem-se que

$$\Pi(\mathrm{sup}\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) \subseteq \mathrm{sup}\mathcal{C}(K, G).$$

Da Proposição 1 e do Lema 2, é verdade que

$$
\begin{aligned}
\mathrm{sup}\mathcal{C}(K,G) &= \Pi(\mathrm{D}(\mathrm{sup}\mathcal{C}(K,G))) \\
&\subseteq \Pi(\mathrm{sup}\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})).
\end{aligned}
$$

Portanto,

$$\Pi(\mathrm{sup}\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) = \mathrm{sup}\mathcal{C}(K, G).$$

(4.2) Do Lema 2, tem-se que

$$\mathrm{D}(\mathrm{sup}\mathcal{C}(K,G)) \subseteq \mathrm{sup}\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}).$$

Dos lemas 1 e 3, é verdade que

$$\begin{aligned} \sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}) &\subseteq \mathrm{D}(\Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}))) \\ &= \mathrm{D}(\sup\mathcal{C}(K, G)). \end{aligned}$$

Portanto,

$$\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}}) = \mathrm{D}(\sup\mathcal{C}(K, G)).$$

■

Sob as condições do Teorema 1, resolver o PCS-D leva a uma solução de controle equivalente à resolução do problema original, o PCS, e vice-versa.

4.2.1 Complexidade de síntese com distinguidores

Para o PCS e o PCS-D, seja n (n_d) e m (m_d), respectivamente, o número de estados dos autômatos que modelam a planta G (G_{d}) e a especificação E (E_{d}). A solução de controle para esses problemas, ou seja, a computação da máxima linguagem controlável, possui ordem de crescimento de $\mathcal{O}[(n.m)^2]$ ($\mathcal{O}[(n_d.m_d)^2]$) no tempo.

Observe que, em geral, $m_d < m$. Porém, $n_d > n$, i.e., se por um lado usar refinamento de eventos permite simplificar tarefas de modelagem de especificações, por outro lado a planta refinada se torna mais complexa pelo fato de que precisa ser associada ao modelo do distinguidor. Em geral, espera-se que $\mathcal{O}[(n.m)^2] = \mathcal{O}[(n_d.m_d)^2]$, i.e., que ambos os problemas, PCS e PCS-D, requeiram o mesmo esforço computacional para serem resolvidos.

4.2.2 Sistema de manufatura modelado com Distinguidores

Considere o exemplo do sistema de manufatura com retrabalho, apresentado no Capítulo 3 e reproduzido na Figura 11.

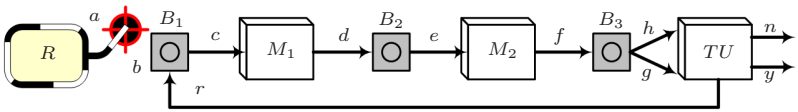


Figura 11 – Pequena fábrica com retrabalho de materiais.

Sobre esse processo foram definidas quatro especificações de controle: E^1, E^2 e E^3, destinadas a evitar o *overflow* e o *underflow* de cada respectivo *buffer*; e E^4, modelada para limitar o número de ciclos de retrabalho. Foi mostrado que a modelagem de E^1, E^2 e E^3 é simples e envolve autômatos com apenas 2 estados cada. Por outro lado, mostrou-se que modelar E^4 é substancialmente mais complexo e essa dificuldade aumenta com a escalabilidade do processo, como, por exemplo, aumentando o número permitido de ciclos.

Nessa seção, mostra-se que modelar E^4 se torna trivial se feito através de um alfabeto refinado, coordenado por um distinguidor. Primeiramente, seguem-se os passos descritos na Subseção 4.1.2 para construir um distinguidor. Então, apresenta-se a modelagem da planta e das especificações em Δ e, por fim, mostra-se a estatística de síntese.

4.2.2.1 Modelagem do distinguidor

Para o exemplo apresentado, o distinguidor pode ser construído como segue:

(i) *Identificação de um conjunto inicial de eventos a serem refinados:* Modelar E^4 consiste em desabilitar eventos g ou h após a ocorrência de f. A complexidade dessa atividade reside no fato de que escolher entre a desabilitação de g ou h depende do número de vezes em que f ocorre para uma mesma peça no sistema, i.e., h é desabilitado após a primeira ocorrência de f, enquanto g é desabilitado do contrário. Portanto, simplificar E^4, claramente, requer refinar o evento f.

(ii) *Definição das instâncias de refinamentos:* Esse passo trata da definição de quantas e quais serão as instâncias de refinamentos para o evento f, identificado no passo anterior. Tal atividade está diretamente associada à especificação, ou seja, requer saber o limite de ciclos permitidos para, então, definir quantas vezes f pode ocorrer para uma mesma peça. No exemplo da Seção 3.4, por simplicidade, foi considerado apenas um ciclo de retrabalho. Suponha agora considerarmos o caso introduzido no exemplo motivador (Seção 3.4.2), onde até dois ciclos são permitidos. É fácil notar que, nesse caso, o evento f pode ocorrer até três vezes para uma mesma peça: uma vez no primeiro ciclo e mais duas vezes nos retrabalhos. Logo, refinar f consiste em associá-lo a três instâncias refinadas, identificadas por f_0, f_1 e f_2, i.e., $\Delta^f = \{f_0, f_1, f_2\}$;

semanticamente, f_0 modela a ocorrência de f pela primeira vez (0 retrabalhos), enquanto f_1 e f_2 modelam f ocorrendo, respectivamente, no primeiro e no segundo retrabalho. Agora, com o refinamento de f, a especificação E^4 pode ser facilmente modelada pelo autômato apresentado na Figura 12.

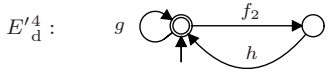


Figura 12 – Modelo preliminar para a especificação E^4_{d}.

(iii) *Complementação do conjunto* Δ*:* Note que distinguir as instâncias de f depende da identificação de cada ciclo. Um ciclo começa em B_1, com os eventos b (um novo ciclo) ou r (reiniciando um ciclo). Naturalmente, os eventos sucessores de b e r devem conduzir a semântica do ciclo identificado, ao longo do sistema, até que por fim as instâncias de f possam ser distinguidas. Nesse sentido, os seguintes eventos devem também ser refinados: $\Delta^c = \{c_0, c_1, c_2\}$; $\Delta^d = \{d_0, d_1, d_2\}$; $\Delta^e = \{e_0, e_1, e_2\}$; $\Delta^g = \{g_0, g_1\}$; $\Delta^r = \{r_1, r_2\}$; $\Delta^\sigma = \{\sigma\}, \forall \sigma \in \{a, b, h, n, y\}$. Para estes novos refinamentos, a semântica equivale àquela definida para as instâncias de f. Por fim, o alfabeto refinado é dado por $\Delta = \dot{\bigcup}_{\sigma \in \Sigma} \Delta^\sigma$.

(iv) *Modelagem do distinguidor:* Resta, para este exemplo, modelar a relação entre os refinamentos em Δ. Este é o papel desempenhado pelo modelo distinguidor, o qual visa apontar qual instância de cada evento deve ser habilitada em cada ciclo de trabalho, sem que para isso sejam impostas quaisquer restrições sobre o sistema. As restrições impostas pelo distinguidor são apenas em termos de "escolha"por uma instância dentre as possíveis, para refinamentos com a mesma máscara. Para $x \in \{c, d, e, f, g, r\}$, modela-se um distinguidor modular H_x, como mostrado na Figura 13.

O módulo H_c distingue o evento c_0 de c_1 e c_2 no estado inicial, e distingue c_1 e c_2 dos demais, após o primeiro e o segundo retrabalho (r_1 e r_2), respectivamente. Isso significa, por exemplo, que quando uma peça é descarregada no sistema pela primeira vez (pelo robô R no *buffer* B_1 com o evento b), a única instância de c habilitada é c_0; se a peça é oriunda do primeiro retrabalho (evento r_1), então c_1 é habilitado; se a origem é o segundo retrabalho (evento r_2), então o evento habilitado é c_2.

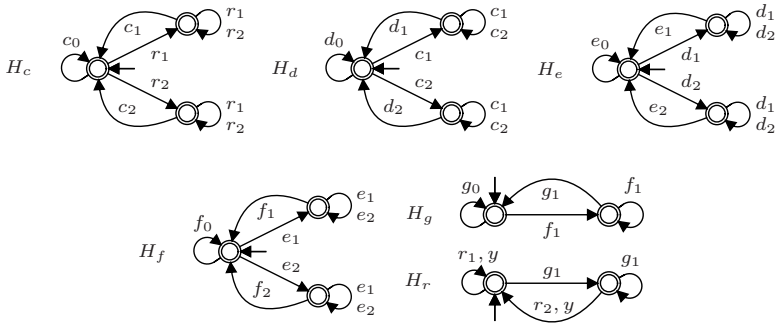


Figura 13 – Módulos distinguidores H_{D}.

Por sua vez, o módulo H_d distingue as instâncias d_0, d_1 e d_2 conforme a ocorrência dos predecessores c_0, c_1 ou c_2. Equivalentemente, os módulos H_e, H_f, H_g e H_r distinguem os eventos com máscaras em e, f, g e r, guiados pela ocorrência dos eventos predecessores.

Note que o distinguidor proposto na Figura 13 é não-proibitivo, no sentido de que cada módulo habilita, em cada estado, ao menos uma instância (exatamente uma nesse caso particular) dos eventos a quem objetiva-se distinguir, e todas as instâncias dos demais eventos.

Compondo-se os autômatos modulares da Figura 13 obtém-se o modelo do distinguidor, denotado por $H_{\mathrm{D}} = H_c \parallel H_d \parallel H_e \parallel H_f \parallel H_g \parallel H_r \parallel H_\Delta$, onde $L^\omega(H_\Delta) = \Delta^*$ e $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$. H_Δ é um autômato com apenas um estado, inicial e marcado, contendo uma transição em autolaço, etiquetada com os eventos Δ, complementando o alfabeto em H_{D}. Para este exemplo em particular, H_{D} possui 308 estados. Como este modelo preserva, em cada estado, um único refinamento habilitado para cada máscara, então ele representa, na verdade, um distinguidor preditível.

4.2.2.2 Modelagem das especificações

Fazendo uso da abordagem de refinamentos proposta, pode-se modificar o autômato da Figura 12 para obter o modelo da especificação E^4, agora sobre o alfabeto Δ, a qual é denotada E^4_{d} e é mostrada na Figura 14.

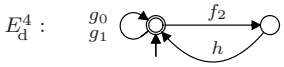


Figura 14 – Modelo para a especificação E^4_{d}, construído com eventos em Δ.

Esse modelo proíbe um teste geral (eventos g_0 e g_1) quando o *buffer* B_3 possui uma peça que já tenha sido rejeitada duas vezes (após f_2); e proíbe um teste final do contrário (evento h no estado inicial).

As demais especificações, E^1, E^2 e E^3, as quais foram inicialmente modeladas em Σ, são agora projetadas em Δ, i.e., $E^1_{\mathrm{d}} = \Pi^{-1}(E^1)$, $E^2_{\mathrm{d}} = \Pi^{-1}(E^2)$ e $E^3_{\mathrm{d}} = \Pi^{-1}(E^3)$. Esses novos modelos são apresentados na Figura 15.

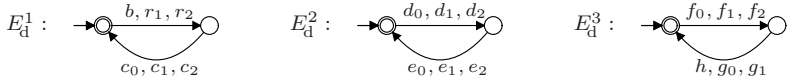


Figura 15 – Modelo para as especificações E^1_{d}, E^2_{d} e E^3_{d} em Δ.

A implementação do mapa inverso Π^{-1} sobre autômatos pode ser concretizada pela simples substituição de cada máscara, em cada transição, pelo respectivo conjunto de refinamentos.

4.2.2.3 Modelagem da planta

Similarmente às especificações, também as plantas que foram modeladas em Σ, na Seção 3.4, são agora projetadas no novo alfabeto Δ, i.e., $G^1_{\mathrm{a}} = \Pi^{-1}(G_1)$, $G^2_{\mathrm{a}} = \Pi^{-1}(G_2)$, $G^3_{\mathrm{a}} = \Pi^{-1}(G_3)$ e $G^4_{\mathrm{a}} = \Pi^{-1}(G_4)$. Esses modelos são apresentados na Figura 24.

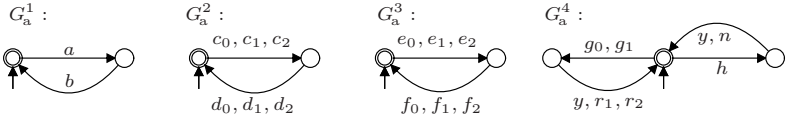


Figura 16 – Modelos para os subsistemas R, M_1, M_2 e TU, em Δ.

Assim, a planta refinada do sistema é modelada pela composição $G_{\mathrm{a}} = G^1_{\mathrm{a}} \parallel G^2_{\mathrm{a}} \parallel G^3_{\mathrm{a}} \parallel G^4_{\mathrm{a}}$, e a controlabilidade dos eventos refinados

segue a coerência em relação à respectiva máscara. Note que G_{a} contempla o mapeamento dos eventos da planta original, em Σ, para o alfabeto Δ, sem implementar nenhuma coordenação entre eles, i.e., $G_{\mathrm{a}} = \Pi^{-1}(G)$. Para fins práticos, G_{a} tem de ser associada com o distinguidor, o que é mostrado a seguir, no contexto da síntese de controle.

4.2.2.4 Solução de Controle

Agora, obtém-se a solução ótima de controle para o exemplo apresentado. A segunda linha da Tabela 9 mostra a solução para o PCS-D, contextualizada em termos do número de estados dos autômatos usados para modelar:

- a planta distinguida $G_{\mathrm{d}} = G_{\mathrm{a}} \| H_{\mathrm{D}}$, tal que $L(G_{\mathrm{d}}) = \Pi^{-1}(L(G)) \cap L_{\mathrm{D}}$;
- a especificação $E_{\mathrm{d}} = E_{\mathrm{d}}^1 \| E_{\mathrm{d}}^2 \| E_{\mathrm{d}}^3 \| E_{\mathrm{d}}^4$;
- a especificação global $K_{\mathrm{d}} = E_{\mathrm{d}} \cap L^{\omega}(G_{\mathrm{d}})$; e
- o supervisor $\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})$, modelado pelo autômato V_{d} tal que $L^{\omega}(V_{\mathrm{d}}) = \sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})$.

A solução para o PCS também é apresentada, para fins comparativos. Os autômatos correspondentes, usados na síntese do PCS, têm número de estados como mostrado na primeira linha da Tabela 9.

Tabela 2 – Número de estados dos autômatos usados na síntese do PCS e do PCS-D.

G	E	K	V
24	1080	**7220**	212
G_{d}	E_{d}	K_{d}	V_{d}
2160	12	**7220**	212

Observe que os modelos das especificações globais (K e K_{d}), bem como o dos supervisores (V e V_{d}), possuem o mesmo número de estados. Tecnicamente, a única diferença é que, no PCS-D, a carga computacional é movida da especificação (uma tarefa de modelagem) para a planta (uma composição). Quando associados, no entanto, planta e especificação levam a um modelo globalmente equivalente.

Salienta-se que ao resolver o PCS não foi possível obter o modelo da especificação E^4 diretamente, i.e., usando apenas os eventos em Σ. De fato, essa modelagem envolve um raciocínio similar àquele usado no Capítulo 3, Figura 8, para obter o modelo de E^4, porém, é ainda mais complexa por considerar um ciclo extra de retrabalho. Portanto, para modelar essa especificação utiliza-se a própria abordagem com distinguidores. Note que uma vez que E^4_{d} foi modelada em Δ, se torna possível obter uma especificação correspondente em Σ, através da composição do autômato de E^4_{d} com o distinguidor H_{D}. Então, mapeia-se essa composição para o alfabeto Σ, tal que $E^4 = \Pi(E^4_{\mathrm{d}} \| L_{\mathrm{D}})$. Nesse caso em particular, o modelo para E^4 possui 560 estados.

Um aspecto relevante em relação ao uso de distinguidores é que, enquanto o modelo de uma especificação pode se tornar bastante complexo quando o sistema escala, o esforço para modelar um distinguidor permanece essencialmente o mesmo. Por exemplo, para o caso de um ciclo adicional de retrabalho (4 no total), foi dito que o modelo da especificação envolve mais de 15 mil estados. Por outro lado, H_{D} continuaria sendo representado por autômatos modulares de, no máximo, 5 estados. Alguns autores apontam ainda alternativas para modularizar o distinguidor, tornando a modelagem ainda mais simples em termos de números de estados (AGUIAR, 2013; CURY et al., 2012).

4.3 USANDO APROXIMAÇÕES NO PCS-D

Vem sendo discutido, neste capítulo, como reformular o PCS para considerar a introdução de um distinguidor. Essa abordagem permite expressar complexas especificações de maneira simples e, ainda assim, leva a uma solução de controle equivalente à solução do PCS original. Um exemplo vem sendo apresentado para ilustrar essa abordagem.

Mesmo em face de claros benefícios, mostrou-se que o PCS-D não provê economias computacionais no procedimento de síntese. De fato, os ganhos resultantes do uso de modelos mais simples de especificações são compensados pela introdução do modelo distinguidor na planta. Desse modo, o custo computacional total para tratar de ambas as versões dos problemas, com e sem refinamentos, é em geral equivalente.

Sob certas condições, no entanto, pode ser possível obter uma solução para o PCS-D, de maneira computacionalmente mais simples. A ideia básica é fazer uso de especificações modeladas sobre um alfa-

beto refinado, como no PCS-D, substituindo, na síntese, a planta G_{d} por uma aproximação G_{a}. A pergunta que surge, então, e que motiva os resultados a serem apresentados na sequência é:

Dado um problema de controle com distinguidores (PCS-D), sob quais condições é possível resolver esse problema e, consequentemente, o problema original, usando uma aproximação G_{a} para a planta G_{d}, tal que se possa também prover vantagens computacionais na síntese?

Os resultados apresentados a seguir objetivam responder a essa pergunta. Inicialmente, propõe-se uma forma de construir aproximações G_{a} para a planta G_{d}, assumindo que G_{d} é tal que $L(G_{\mathrm{d}}) = \mathrm{D}(L(G))$, em que D é um distinguidor preditível. Então, discute-se como tais aproximações levam a uma solução para o PCS-D, obtida com um menor esforço computacional. Finalmente, apresentam-se condições sob as quais se pode determinar a máxima permissividade de supervisores obtidos via aproximações.

4.3.1 Construindo Aproximações

Esta seção formaliza uma maneira de se construir uma aproximação G_{a} para uma planta G_{d}. É de interesse desta tese explorar uma classe particular de aproximações, com propriedades que as tornam adequadas para a síntese de supervisores. Essa classe é definida como segue.

Definição 7 (D-aproximação) *Sejam* D *um distinguidor preditível,* D_{a} *um distinguidor tal que* $L_{\mathrm{D}} \subseteq L_{\mathrm{Da}} \subseteq \Delta^*$ *e* G_{d} *uma planta tal que* $L(G_{\mathrm{d}}) = \mathrm{D}(L(G))$. G_{a}*, tal que* $L(G_{\mathrm{a}}) = \mathrm{D}_{\mathrm{a}}(L(G))$ *e* $L^{\omega}(G_{\mathrm{a}}) = \mathrm{D}_{\mathrm{a}}(L^{\omega}(G))$*, é dita ser uma D-aproximação para* G_{d}*.*

A Definição 7 estabelece uma classe de distinguidores D_{a} com a característica de preservar toda cadeia distinguida por D, sendo D preditível. Quando aplicado sobre uma planta G de um SED, D_{a} produz uma aproximação G_{a} para G_{d}, tal que $L(G_{\mathrm{d}}) \subseteq L(G_{\mathrm{a}})$ e $L^{\omega}(G_{\mathrm{d}}) \subseteq L^{\omega}(G_{\mathrm{a}})$.

Resta mostrar como D_{a} pode ser modelado de modo que, ao ser associado a G, produza uma *D-aproximação* G_{a} como na Definição 7. Assumindo que o autômato H_{D}, tal que $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$, é obtido pela composição de um conjunto de autômatos modulares (como exemplificado na Seção 3.4), então a composição de qualquer subconjunto desses

autômatos modela um possível $\mathrm{D_a}$, uma vez que $L_{\mathrm{D}} \subseteq L_{\mathrm{Da}}$.

Um caso particular é, por exemplo, modelar $\mathrm{D_a}$ usando apenas o módulo H_Δ. Nessa situação, $L_{\mathrm{Da}} = \Delta^*$ representa, na verdade, a *D-aproximação* G_{a} mais "relaxada" possível de ser produzida para G_{d}, pois $L(G_{\mathrm{a}}) = \Pi^{-1}(L(G))$. Outras *D-aproximações* podem ser construídas através da composição incremental de módulos do distinguidor. O uso de regras heurísticas (AGUIAR, 2013) pode ajudar na escolha de módulos específicos a serem compostos, em razão de resultados de síntese.

4.3.2 Propriedades das *D-Aproximações*

Esta seção formaliza a relação entre uma *D-aproximação* e o modelo correspondente, não aproximado. Os resultados a serem apresentados se baseiam no conceito de Λ-*Preservância*, apresentado como segue.

Definição 8 (Λ-Preservância) *Dadas duas linguagens* $L_1 \subseteq L_2 \subseteq \Delta^*$, *um mapa mascarador* $\Pi : \Delta^* \to \Sigma^*$ *e um conjunto de máscaras* $\Lambda \subseteq \Sigma$, L_1 *é* Λ-*Preservante e.r.a.* L_2 *se* $\forall \sigma \in \Lambda$ *e* $t \in \overline{L_1}$,

$$t\Delta^\sigma \cap \overline{L_2} \neq \emptyset \;\Rightarrow\; t\Delta^\sigma \cap \overline{L_1} \neq \emptyset.$$

Ou seja, L_1 é Λ-*Preservante e.r.a.* L_2 se, para toda máscara $\sigma \in \Lambda$, sempre que um evento refinado pertencente a Δ^σ é habilitado em $\overline{L_2}$, pelo menos uma instância em Δ^σ é também habilitada em $\overline{L_1}$.

Para as linguagens $L_1 \subseteq L_2 \subseteq \Delta^*$ e $\Lambda \subseteq \Sigma$, define-se o conjunto

$$\mathcal{P}_\Lambda(L_1, L_2) = \{K \subseteq L_1 \mid K \text{ é } \Lambda\text{-}\textit{Preservante} \text{ e.r.a. } L_2\}.$$

Pode ser mostrado que $\mathcal{P}_\Lambda(L_1, L_2)$ possui um elemento supremo, denotado por $\sup\mathcal{P}_\Lambda(L_1, L_2)$, que pode ser computado em um tempo limitado por uma função polinomial, com grau correspondente ao número de estados dos autômatos que modelam L_1 e L_2. Um algoritmo para computar $\sup\mathcal{P}_\Lambda(L_1, L_2)$ é apresentado em (BOUZON; QUEIROZ; CURY, 2008).

A partir da Definição 8, a relação entre G_{a} e G_{d} pode ser estabelecida como segue.

Proposição 5 *Para dois distinguidores quaisquer* $\mathrm{D}_1, \mathrm{D}_2 : \Sigma^* \to 2^{\Delta^*}$

e uma linguagem $L \subseteq \Sigma^*$ *tal que* $\mathrm{D}_1(L) \subseteq \mathrm{D}_2(L)$,

$$\mathrm{D}_1(L) \text{ é } \Sigma\text{-}\textit{Preservante} \quad \textit{e.r.a.} \quad \mathrm{D}_2(L).$$

Prova: Primeiro, note que $\forall\ \sigma \in \Sigma$ e $\forall\ t \in \overline{\mathrm{D}_1(L)}$, $t \in \mathrm{D}_1(\Pi(t))$ e, pela Definição 3 (iii), existe $\delta \in \Delta^\sigma$, tal que $t\delta \in \mathrm{D}_1(\Pi(t)\sigma)$. Então, $t\Delta^\sigma \cap \overline{\mathrm{D}_2(L)} \neq \emptyset$ implica que $\Pi(t)\sigma \in \overline{L}$ e, então, existe $\delta \in \Delta^\sigma$, tal que $t\delta \in \mathrm{D}_1(\Pi(t)\sigma) \subseteq \overline{\mathrm{D}_1(L)}$. Portanto, $t\Delta^\sigma \cap \overline{\mathrm{D}_1(L)} \neq \emptyset$. ■

A partir da Proposição 5, pode-se enunciar o seguinte lema.

Lema 4 *Seja* G_d *como no PCS-D e* G_a *uma D-aproximação para* G_d, *tal que* $L(G_\mathrm{d}) \subseteq L(G_\mathrm{a})$, *então* $L(G_\mathrm{d})$ *é* Σ_u*-Preservante e.r.a.* $L(G_\mathrm{a})$.

Prova: Pela Proposição 5, $L(G_\mathrm{d})$ é Σ-*Preservante e.r.a.* $L(G_\mathrm{a})$. Pela Definição 8, é verdade que para duas linguagens quaisquer $L_1 \subseteq L_2 \subseteq \Delta^*$, se L_1 é Λ-*Preservante e.r.a.* L_2, então L_1 também é Λ'-*Preservante* e.r.a. L_2, para qualquer subconjunto de máscaras $\Lambda' \subseteq \Lambda$. Como $\Sigma_u \subseteq \Sigma$, então é verdade que $L(G_\mathrm{d})$ é Σ_u-*Preservante e.r.a.* $L(G_\mathrm{a})$. ■

4.3.3 Resolvendo o PCS-D usando Aproximações

O teorema seguinte mostra que uma solução para o PCS-D pode ser obtida através de uma *D-aproximação* G_a para a planta G_d. O teorema também provê um meio de checar se uma solução aproximada é a menos restritiva, sem que o resultado $\sup\mathcal{C}(K_\mathrm{d}, G_\mathrm{d})$ seja, de fato, computado.

Teorema 2 *Dados* G_d *e* E_d, *como no PCS-D, seja* G_a *uma D - aproximação para* G_d. *Para* $K_\mathrm{a} = E_\mathrm{d} \cap L^\omega(G_\mathrm{a})$, *se* $\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a})$ *e* L_D *forem não-conflitantes, então um supervisor* S_a *que implementa* $\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a}) \cap L_\mathrm{D}$ *é uma solução para o PCS-D. Além disso,*

$$\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a}) \cap L_\mathrm{D} \subseteq \sup\mathcal{C}(K_\mathrm{d}, G_\mathrm{d}) \subseteq \sup\mathcal{P}_{\Sigma_\mathrm{u}}(K_\mathrm{a}, L(G_\mathrm{a})) \cap L_\mathrm{D}.$$

Os lemas 5 e 6 auxiliam na prova do Teorema 2.

Lema 5 *Dadas as linguagens* $L_1, L_2 \subseteq \Delta^*$, *com* $L_1 \subseteq L_2$, *se* L_1 *é controlável e.r.a.* $\overline{L_2}$, *então* L_1 *é* Σ_u*-Preservante e.r.a.* L_2.

Prova: Da estrutura de controle, introduzida para o alfabeto refinado, tem-se que para cada $\Delta^\sigma \subseteq \Delta_u$, $\Pi(\Delta^\sigma) \in \Sigma_u$. Seja L_1 controlável e.r.a. $\overline{L_2}$. Então, para cada $\Delta^\sigma \subseteq \Delta_u$ e $s \in \overline{L_1}$, $s\Delta^\sigma \cap \overline{L_2} \subseteq \overline{L_1}$. Isso implica que $s\Delta^\sigma \cap \overline{L_2} = s\Delta^\sigma \cap \overline{L_2} \cap \overline{L_1} = s\Delta^\sigma \cap \overline{L_1}$. Assim, $s\Delta^\sigma \cap \overline{L_2} \neq \emptyset$ implica que $s\Delta^\sigma \cap \overline{L_1} \neq \emptyset$, provando o Lema 5. ∎

Lema 6 *Considere o mapa* $\Pi : \Delta^* \to \Sigma^*$, *um alfabeto* $\Lambda \subseteq \Sigma$ *e as linguagens* $L_1 \subseteq L_2 \subseteq L_3 \subseteq \Delta^*$. *Se* L_1 *é* Λ-*Preservante e.r.a.* L_2 *e* L_2 *é* Λ-*Preservante e.r.a.* L_3, *então* L_1 *também é* Λ-*Preservante e.r.a.* L_3.

Prova: Para todo $s \in \overline{L_1}$ e todo $\sigma \in \Lambda$, é verdade que

$$\begin{aligned} &s\Delta^\sigma \cap \overline{L_3} \neq \emptyset \Rightarrow \\ &s\Delta^\sigma \cap \overline{L_2} \neq \emptyset \Rightarrow \\ &s\Delta^\sigma \cap \overline{L_1} \neq \emptyset. \end{aligned}$$

∎

Prova: (para o Teorema 2).

- $\sup\mathcal{C}(K_d, G_d) \subseteq \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a)) \cap L_D$:

Primeiro, note que $\sup\mathcal{C}(K_d, G_d)$ é controlável e.r.a. $L(G_d)$. Então, pelo Lema 5, $\sup\mathcal{C}(K_d, G_d)$ é Σ_u-*Preservante e.r.a.* $L(G_d)$. Além disso, observe que sendo G_a uma *D-aproximação* para G_d então, pela Definição 7, $L(G_d) \subseteq L(G_a)$ e, pelo Lema 4, $L(G_d)$ é Σ_u-*Preservante* e.r.a. $L(G_a)$.

Assim, como $\sup\mathcal{C}(K_d, G_d) \subseteq L(G_d) \subseteq L(G_a)$ então, pelo Lema 6, é verdade que $\sup\mathcal{C}(K_d, G_d)$ é Σ_u-*Preservante e.r.a.* $L(G_a)$. Como também $\sup\mathcal{C}(K_d, G_d) \subseteq K_a$, então $\sup\mathcal{C}(K_d, G_d) \subseteq \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a))$. Por fim, $\sup\mathcal{C}(K_d, G_d) \subseteq L(G_d)$ implica que

$$\begin{aligned} \sup\mathcal{C}(K_d, G_d) &\subset \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a)) \cap L(G_d) \\ &= \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a)) \cap \Pi^{-1}(L(G)) \cap L_D \cap L_{Da} \\ &= \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a)) \cap L(G_a) \cap L_D \\ &= \sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a)) \cap L_D. \end{aligned}$$

- $\sup\mathcal{C}(K_a, G_a) \cap L_D \subseteq \sup\mathcal{C}(K_d, G_d)$:

Note que $L(G_{\rm d}) \subseteq L(G_{\rm a})$ implica na controlabilidade de $\sup\mathcal{C}(K_{\rm a}, G_{\rm a})$ e.r.a. $L(G_{\rm d})$, pois

$$\begin{aligned}\overline{\sup\mathcal{C}(K_{\rm a}, G_{\rm a})}\Delta_{\rm u} \cap L(G_{\rm d}) &\subseteq \overline{\sup\mathcal{C}(K_{\rm a}, G_{\rm a})}\Delta_{\rm u} \cap L(G_{\rm a}) \\ &\subseteq \overline{\sup\mathcal{C}(K_{\rm a}, G_{\rm a})}.\end{aligned}$$

Ainda, como $L(G_{\rm d}) \subseteq L_{\rm D}$, então $L_{\rm D}$ é controlável e.r.a. $L(G_{\rm d})$. Logo, pelo não conflito entre $\sup\mathcal{C}(K_{\rm a}, G_{\rm a})$ e $L_{\rm D}$, tem-se que $\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ é controlável e.r.a. $L(G_{\rm d})$. Além do mais,

$$\begin{aligned}\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D} &\subseteq K_{\rm a} \cap L_{\rm D} \\ &= K_{\rm d}.\end{aligned}$$

Portanto, pela inclusão de $\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ em $K_{\rm d}$ e pela sua controlabilidade e.r.a. $L(G_{\rm d})$,

$$\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D} \subseteq \sup\mathcal{C}(K_{\rm d}, G_{\rm d}).$$

- $\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ é uma solução para o PCS-D:

Esse resultado segue da controlabilidade de $\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ e.r.a. $L(G_{\rm d})$, do não conflito entre $\sup\mathcal{C}(K_{\rm a}, G_{\rm a})$ e $L_{\rm D}$, e da inclusão

$$\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D} \subseteq \sup\mathcal{C}(K_{\rm d}, G_{\rm d}) \subseteq K_{\rm d}.$$

Portanto, existe um supervisor $S_{\rm a}$ tal que

$$\begin{aligned}L^{\omega}(S_{\rm a}/G_{\rm d}) &= \sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D} \\ &\subseteq K_{\rm d}.\end{aligned}$$

Além disso, como $S_{\rm a}$ pode ser implementado por um autômato $V_{\rm a}$ com $L^{\omega}(V_{\rm a}) = \sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ e como $L(V_{\rm a}) = \overline{\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}} \subseteq L_{\rm D}$, então $S_{\rm a}$ é um supervisor não-ambíguo, e uma solução para o PCS-D. ∎

O Teorema 2 mostra que um supervisor $S_{\rm a}$ implementado por um autômato $V_{\rm a}$, tal que $L^{\omega}(V_{\rm a}) = \sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ e $L(V_{\rm a}) = \overline{L^{\omega}(V_{\rm a})}$, é uma solução para o PCS-D, embora essa solução possa ser mais restritiva do que uma implementada por $\sup\mathcal{C}(K_{\rm d}, G_{\rm d})$.

A proximidade entre $\sup\mathcal{C}(K_{\rm a}, G_{\rm a})$ e $\sup\mathcal{P}_{\Sigma_{\rm u}}(K_{\rm a}, L(G_{\rm a}))$ fornece uma ideia do quão distante $\sup\mathcal{C}(K_{\rm a}, G_{\rm a}) \cap L_{\rm D}$ pode estar da solução ótima $\sup\mathcal{C}(K_{\rm d}, G_{\rm d})$. Em particular, se $\sup\mathcal{C}(K_{\rm a}, G_{\rm a})$ é igual a

$\sup\mathcal{P}_{\Sigma_u}(K_a, L(G_a))$, então a solução obtida com a *D-aproximação* G_a é ótima, como sugere o Corolário seguinte.

Corolário 1 *Dados* G_d, E_d *e* G_a, *como no Teorema 2, se* $\sup\mathcal{C}(K_a, G_a)$ *e* L_D *forem não-conflitantes, então*

$$\sup\mathcal{C}(K_a, G_a) = sup P_{\Sigma_u}(K_a, L(G_a))$$

$$\Rightarrow$$

$$\sup\mathcal{C}(K_d, G_d) = \sup\mathcal{C}(K_a, G_a) \cap L_D.$$

O Corolário 2 provê outra condição suficiente para garantir que a solução obtida através de uma *D-aproximação* é ótima.

Corolário 2 *Dados* G_d, E_d, G_a *e* K_a, *como no Teorema 2, se* K_a *é controlável e.r.a.* G_a *e não-conflitante com* L_D, *então* $K_d = K_a \cap L_D$ *é controlável e.r.a.* G_d *e, assim, um supervisor* S_a *que implementa* $K_a \cap L_D$ *é uma solução ótima para o PCS-D.*

Prova: Primeiro, note que

$$\begin{aligned} K_a \cap L_D &= E_d \cap \Pi^{-1}(L^\omega(G)) \cap L_{Da} \cap L_D \\ &= E_d \cap \Pi^{-1}(L^\omega(G)) \cap L_D \\ &= E_d \cap L^\omega(G_d) \\ &= K_d. \end{aligned}$$

Como K_a é controlável e.r.a. G_a e não conflitante com L_D, então

$$\begin{aligned} K_d &= K_a \cap L_D \\ &= \sup\mathcal{C}(K_a, G_a) \cap L_D \\ &\subseteq \sup\mathcal{C}(K_d, G_d) \text{ (pelo Teorema 2).} \end{aligned}$$

Como, por definição, também $\sup\mathcal{C}(K_d, G_d) \subseteq K_d$, então $\sup\mathcal{C}(K_d, G_d) = K_d$. Portanto, K_d é controlável e $L^\omega(S_a/G_d) = K_a \cap L_D = K_d$. ■

De modo geral, o Teorema 2 mostra que o PCS-D pode ser resolvido usando uma *D-aproximação* G_a para a planta G_d. Se, eventualmente (pelo Corolário 1), essa abordagem levar a uma solução subótima, o resultado pode ser melhorado por intermédio da distinção incremental da aproximação.

Note que ao considerar $L_{Da} = \Delta^*$ têm-se a vantagem de modelar especificações em Δ, sem que isso introduza complexidade alguma ao

modelo da planta. Nesse caso, G_{a}, tal que $L(G_{\mathrm{a}}) = \Pi^{-1}(L(G))$, é um autômato com o mesmo número de estados de G, modelado em Σ.

Por outro lado, ao considerar $L_{\mathrm{Da}} = L_{\mathrm{D}}$ têm-se vantagens equivalentes na modelagem, mas sem nenhum ganho computacional na síntese já que, nesse caso, $G_{\mathrm{a}} = G_{\mathrm{d}}$. Quando o modelo H_{D} do distinguidor é construído modularmente (como no exemplo anterior), pode-se construir aproximações intermediárias através da introdução gradual de módulos de H_{D} na planta. O exemplo seguinte ilustra essa abordagem.

Vale notar que computar um supervisor S_{a}, que resolve o PCS-D, não requer calcular explicitamente a intersecção $\sup\mathcal{C}(K_{\mathrm{a}}, G_{\mathrm{a}}) \cap L_{\mathrm{D}}$. Ao invés disso, o controlador pode ser implementado por dois autômatos concorrentes, V_{a} e H_{D}, de maneira tal que $L^{\omega}(V_{\mathrm{a}}) = \sup\mathcal{C}(K_{\mathrm{a}}, G_{\mathrm{a}})$ e $L^{\omega}(H_{\mathrm{D}}) = L_{\mathrm{D}}$. Esses autômatos são dirigidos por eventos gerados por G_{d} e a ação de controle resultante é tal que, após uma cadeia $s \in L(S_{\mathrm{a}}/G_{\mathrm{d}})$, os eventos elegíveis são aqueles habilitados em ambos $L^{\omega}(H_{\mathrm{D}})$ e $L^{\omega}(V_{\mathrm{a}})$. Como $\sup\mathcal{C}(K_{\mathrm{a}}, G_{\mathrm{a}})$ e L_{D} são não-conflitantes, então S_{a} é não-bloqueante.

Finalmente, note que o teste de não conflito é a única operação que de fato requer a intersecção com a linguagem L_{D}. Simplificar esse teste não é objetivo desta tese, mas algumas abordagens permitem executá-lo com menor esforço computacional, como por exemplo as técnicas de projeções de linguagens (PENA; CURY; LAFORTUNE, 2009) ou de verificação composicional (FLORDAL; MALIK, 2009).

4.3.4 Sistema de manufatura modelado com aproximações

Esta seção revisita o exemplo do sistema de manufatura apresentado na Seção 3.4. Naquela ocasião, foi ilustrado como um distinguidor pode ser útil para a modelagem de especificações de controle, mas não foram explorados aspectos de redução de complexidade de síntese. Nesta seção, exemplifica-se como o uso de uma *D-aproximação* permite manter as vantagens de modelagem e, ao mesmo tempo, obter economia computacional.

4.3.4.1 Exemplo de síntese com uma *D-aproximação*

Ao usar uma *D-aproximação* para resolver um problema de controle, é intuitivo pensar em construir um modelo que permita o máximo possível de vantagens computacionais. Nesse sentido, a primeira

D-aproximação a ser considerada neste exemplo é uma representada pela linguagem $L_{\mathrm{D}_{\mathrm{a}}^{1}} = \Delta^*$. Esse efeito pode ser reproduzido na planta através da distinção de G por um distinguidor $\mathrm{D}_{\mathrm{a}}^{1}$, modelado por H_Δ, um autômato com apenas um estado, inicial e marcado, contendo uma transição em autolaço, etiquetada com os eventos Δ.

Nesse caso, $L_{\mathrm{D}} \subseteq L_{\mathrm{D}_{\mathrm{a}}^{1}} = \Delta^*$ e, pela Definição 7, $G_{\mathrm{a}1}$, tal que $L(G_{\mathrm{a}1}) = \mathrm{D}_{\mathrm{a}}^{1}(L(G)) = \Pi^{-1}(L(G))$, é uma *D-aproximação* para G_{d}. Então, a síntese de controle contempla a *D-aproximação* $G_{\mathrm{a}1}$, a especificação E_{d} e a especificação global $K_{\mathrm{a}1} = E_{\mathrm{d}} \cap L^{\omega}(G_{\mathrm{a}1})$, levando à obtenção de um supervisor $\sup\mathcal{C}(K_{\mathrm{a}1}, G_{\mathrm{a}1})$, modelado pelo autômato $V_{\mathrm{a}1}$. A segunda linha da Tabela 3 apresenta o número de estados desses autômatos. Para fins comparativos, a primeira linha da tabela reproduz a síntese sem aproximações, tal como apresentada anteriormente.

Tabela 3 – Síntese de controle usando *D-aproximações*.

G_{d}	E_{d}	K_{d}	V_{d}	$V_{\mathrm{d}} \| H_{\mathrm{D}}$
2160	12	7220	212	**212**
$G_{\mathrm{a}1}$	E_{d}	$K_{\mathrm{a}1}$	$V_{\mathrm{a}1}$	$V_{\mathrm{a}1} \| H_{\mathrm{D}}$
24	12	228	62	***196***
$G_{\mathrm{a}2}$	E_{d}	$K_{\mathrm{a}2}$	$V_{\mathrm{a}2}$	$V_{\mathrm{a}2} \| H_{\mathrm{D}}$
48	12	576	92	**212**

A síntese de $V_{\mathrm{a}1}$ requer um esforço computacional muito menor do que o necessário para computar V_{d}, uma vez que o autômato para $K_{\mathrm{a}1}$ possui apenas 228 estados, enquanto o modelo de K_{d} possui 7.220 estados. Além do mais, é possível mostrar que $\sup\mathcal{C}(K_{\mathrm{a}1}, G_{\mathrm{a}1})$ e L_{D} são não-conflitantes. Portanto, $V_{\mathrm{a}1}$ e H_{D} (com $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$) podem ser usados para implementar um supervisor não-ambíguo $S_{\mathrm{a}1}$ para a planta G distinguida por D.

Entretanto, pode também ser mostrado que

$$\sup\mathcal{C}(K_{\mathrm{a}1}, G_{\mathrm{a}1}) \neq \sup\mathcal{P}_{\Sigma_{\mathrm{u}}}(K_{\mathrm{a}1}, L(G_{\mathrm{a}1})).$$

Como essa diferença não satisfaz as condições do Corolário 1, então nada pode ser dito a respeito da otimidade da solução de controle, i.e., $S_{\mathrm{a}1}$ pode ser mais restritivo do que S_{d}.

Para compreender a razão pela qual $S_{\mathrm{a}1}$ pode ser subótimo, considere uma análise informal do processo mostrado na Figura 11. Note que a especificação E_{d}^{4} modela a proibição dos refinamentos de g ou de

h conforme qual instância refinada de f é elegível. Uma vez que a distinção dessas instâncias foi omitida da síntese, então toda vez que um evento f é possível na planta, as três instâncias, f_0, f_1 e f_2, se tornam habilitadas. Assim, não tendo o supervisor S_{a1} a informação de qual instância será de fato elegível, ele pode precisar prevenir o ingresso de peças no sistema (por razões de *overflow* em B_1), até que uma delas de fato ocorra e assim permita ao supervisor efetivar a ação de controle especificada em E_{d}^4, com a garantia de *não-overflow* em B_1. Como consequência, S_{a1} é de fato mais restritivo que S_{d}.

Contudo, suponha que a distinção dos refinamentos de f seja incorporada à síntese. Isso implica que imediatamente após a ocorrência de um dos refinamentos sobre a máscara e, já se saiba qual instância de f será habilitada. Logo, a ação de controle já pode ser prevista e o ingresso de peças no sistema pode ser liberado, quando esse for o caso. Nem sempre distinguir as instâncias de uma única máscara melhora o resultado de síntese, mas fornece perspectivas para tal. Além do mais, distinções podem ser incorporadas gradualmente à síntese, até que resultem em melhorias.

4.3.4.2 Exemplo de síntese com uma *D-aproximação* melhorada

Suponha agora prover à síntese de controle, informações acerca da distinção dos refinamentos sobre a máscara f. Mais precisamente, considere incorporar o módulo H_f ao modelo da planta. Então, seja G_{a2} uma nova *D-aproximação* para G_{d}, construída pela aplicação de um distinguidor $\mathrm{D}_{\mathrm{a}}^2$ sobre G. $\mathrm{D}_{\mathrm{a}}^2$ é, nesse caso, modelado pelo autômato $H_f \| H_\Delta$. Então, para $L_{\mathrm{D}_{\mathrm{a}}^2} = L(H_f \| H_\Delta)$, é verdade que $L_{\mathrm{D}} \subseteq L_{\mathrm{D}_{\mathrm{a}}^2} \subseteq \Delta^*$ e, portanto, G_{a2} é uma *D-aproximação* como na Definição 7 e $L(G_{\mathrm{a2}}) = \mathrm{D}_{\mathrm{a}}^2(L(G)) = \Pi^{-1}(L(G)) \cap L_{\mathrm{D}_{\mathrm{a}}^2}$.

A síntese é então complementada pelos autômatos da especificação E_{d} e a especificação global $K_{\mathrm{a2}} = E_{\mathrm{d}} \cap L(G_{\mathrm{a2}})$, resultando em um supervisor V_{a2} tal que $L^\omega(V_{\mathrm{a2}}) = \sup\mathcal{C}(K_{\mathrm{a2}}, G_{\mathrm{a2}})$. O número de estados desses modelos é mostrado na última linha da Tabela 3.

A síntese de V_{a2} envolve, agora, um modelo K_{a2} com 576 estados e, portanto, passa a ser mais complexa do que a síntese de V_{a1}, efetuada sobre o modelo K_{a1}, com 228 estados. Contudo, essa complexidade é ainda substancialmente menor do que a observada para computar V_{d}, a qual envolve um modelo com 7.220 estados.

Como $\sup\mathcal{C}(K_{\mathrm{a2}}, G_{\mathrm{a2}})$ e L_{D} são não-conflitantes, $\sup\mathcal{C}(K_{\mathrm{a2}}, G_{\mathrm{a2}}) \cap L_{\mathrm{D}}$ é uma solução para o PCS-D, pelo Teorema 2. Além disso, agora é

verdade que

$$\sup\mathcal{C}(K_{a2}, G_{a2}) = \sup\mathcal{P}_{\Sigma_u}(K_{a2}, L(G_{a2}))$$

Pelo Corolário 1, isso garante que V_{a2} e H_D podem ser usados para implementar uma solução ótima de controle para o sistema de manufatura, tal que

$$\begin{aligned} L^{\omega}(S_{a2}/G_d) &= \sup\mathcal{C}(K_{a2}, G_{a2}) \cap L_D \\ &= \sup\mathcal{C}(K_d, G_d). \end{aligned}$$

4.4 SUMÁRIO

Este capítulo apresenta o conceito de distinguidor, proposto para simplificar a modelagem de especificações e, ao mesmo tempo, levar a uma solução de controle equivalente ao método sem distinguidores.

O preço a ser pago por essa simplificação é que o modelo da planta se torna mais complexo pela incorporação do modelo distinguidor, de modo que, computacionalmente, o esforço para obter uma solução de controle é equivalente, usando ou não distinguidores.

Por isso, foi introduzida uma classe de aproximações que podem substituir a planta complexa da síntese, proporcionando importantes ganhos computacionais, enquanto as especificações simples podem ainda ser usadas. Foram mostradas as condições sob as quais a solução de controle, computada através de aproximações, permanece equivalente à original, sem distinguidores.

As vantagens da abordagem proposta neste capítulo foram ilustradas no contexto de um exemplo de um sistema de manufatura.

5 USANDO DISTINGUIDORES NO CONTROLE MODULAR LOCAL

Em alguns problemas de controle, principalmente naqueles envolvendo sistemas de grande porte, compostos por inúmeros subsistemas e especificações, a obtenção de um supervisor monolítico pode ser computacionalmente inviável. Isso, porque uma operação monolítica requer a composição, em uma única estrutura, dos autômatos considerados no problema, o que pode levar à *explosão* no espaço de estados, tornando o PCS intratável.

Alternativamente, nas extensões da TCS têm-se sugerido a modularização do problema de controle, de modo que partes possam ser tratadas individualmente e, em tese, de maneira mais simples. Então, condições são providas para mostrar que a ação conjunta das soluções modulares resolve o problema global de controle. São inúmeras as formas de modularização sugeridas, incluindo o particionamento baseado em estágios composicionais (MOHAJERANI et al., 2011), em eventos gerados pelo sistema (CAI; WONHAM, 2010), em especificações de controle (RAMADGE; WONHAM, 1987a), em plantas e especificações (QUEIROZ; CURY, 2000a), etc.

Contudo, modularizar a resolução de um problema não implica, diretamente, em viabilizar tal resolução, pois, como vem sendo discutido ao longo desta tese, modelar certos problemas de controle pode ser uma tarefa irrealizável. Nesse sentido, este capítulo propõe a associação de uma das abordagens de síntese apresentadas anteriormente, em particular o uso de distinguidores, a uma estrutura modular. Os resultados a serem apresentados permitem avaliar os efeitos dessa associação, bem como combinar as vantagens de cada método.

5.1 CONTROLE MODULAR CLÁSSICO

O *Controle Modular Clássico* (CMC) (RAMADGE; WONHAM, 1987a; WONHAM; RAMADGE, 1988) representa uma alternativa para minimizar a sobrecarga computacional observada em um procedimento monolítico de síntese. No CMC, ao invés de se construir um único supervisor S que atenda a um conjunto global de especificações $I = \{1, \cdots, m\}$, é sintetizado um supervisor modular S_i para cada especificação $i \in I$. A Figura 17 ilustra a arquitetura do CMC.

Toda vez que, após uma cadeia $s \in \Sigma^*$ qualquer, um evento

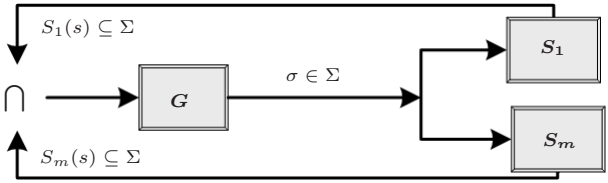


Figura 17 – Estrutura de controle no CMC.

$\sigma \in \Sigma$ for observado na planta, cada supervisor modular mapeia um subconjunto $S_i(s) \subseteq \Sigma$ de eventos a serem habilitados. Assim, se um evento é observado e não fizer parte de ambos os subconjuntos $S_i(s)$, ele é desabilitado na planta.

Um dos principais benefícios do CMC é a flexibilidade agregada pela resolução de um problema de controle por módulos, o que permite alterar, inserir ou remover um requisito em particular, sem afetar o sistema como um todo. Além disso, as operações individuais de síntese são, em geral, mais simples se comparadas à uma operação monolítica.

5.1.1 Complexidade de síntese no CMC

Seja um SED modelado por $p+q$ subsistemas, com no máximo m estados cada, de modo que m^{p+q} é o número de estados do autômato que modela a planta G. Ainda, seja G restrito pelas especificações $E^i,\ i \in I = \{1, \cdots, e\}$, tendo E^i no máximo n estados. Considerando que cada E^i restringe apenas uma parte p dos subsistemas $p+q$, e que q compõe os subsistemas que não são afetados por nenhuma das especificações, então q não precisa fazer parte da síntese dos supervisores modulares.

Logo, a complexidade de síntese no CMC é na ordem de $\mathcal{O}[(n.m^p)^2]$ e, portanto, é claramente mais simples do que no PCS, onde é na ordem de $\mathcal{O}[(n^e.m^{p+q})^2]$. Na prática, as vantagens do CMC em relação ao PCS vertem de duas fontes principais: (i) do fato de que as especificações não são compostas entre si; e (ii) do uso de um modelo parcial da planta na síntese, o qual não contempla os subsistemas assíncronos com as especificações.

Porém, note que é pouco razoável que um subsistema que não precise ser controlado (não compartilhe eventos com nenhuma especificação) faça parte do problema de controle. Logo, pode-se assumir que a planta G compõe, na verdade, todos os $p+q$ subsistemas. Como o

fator $p + q$ determina o crescimento de G, então o CMC se torna degradado pela impossibilidade de tratar de problemas de grande porte. Uma extensão do CMC, o Controle Modular Local, revitaliza o conceito de controle supervisório modular, e é apresentada na sequência.

5.2 CONTROLE MODULAR LOCAL

O *Controle Modular Local* (CML) (QUEIROZ; CURY, 2000a, 2000b) contorna uma das principais lacunas do CMC: o fato de que, embora a síntese seja descentralizada, o modelo da planta considerado para a síntese é ainda monolítico, o que pode ser um empecilho computacional para resolver problemas de controle.

Similarmente à abordagem clássica, o CML também projeta a construção de um supervisor para cada especificação $i \in I$ envolvida no problema. A diferença é que cada supervisor é obtido a partir de uma versão parcial da planta e, logo, exerce a função de um *supervisor local* para o sistema. A Figura 18 sumariza a arquitetura do CML.

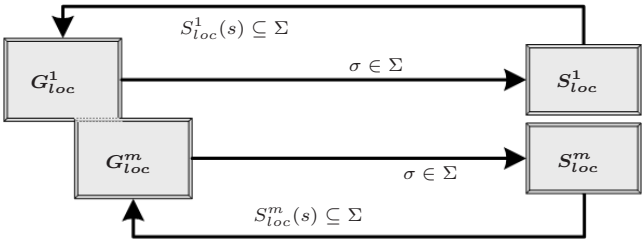


Figura 18 – Estrutura de controle no CML.

Nessa arquitetura, plantas locais são denotadas por G^i_{loc} e são construídas por composições específicas de modelos de subsistemas. Quando um evento $\sigma \in \Sigma$ é observado localmente, o respectivo supervisor S^i_{loc} mapeia um subconjunto de eventos $S^i_{loc}(s) \subseteq \Sigma$ a serem habilitados na planta local. Eventos possíveis em G^i_{loc} que não fazem parte de $S^i_{loc}(s) \subseteq \Sigma$ são inibidos pela ação de controle.

A escolha por quais plantas devem ser compostas localmente é regida pela intersecção entre os alfabetos das plantas modulares e o de uma especificação em particular. A Figura 19 exemplifica a composição de plantas locais para um caso específico envolvendo $m = 3$ modelos de especificações e $n = 4$ modelos de plantas assíncronas.

Na figura, os conectivos entre as especificações e as plantas de-

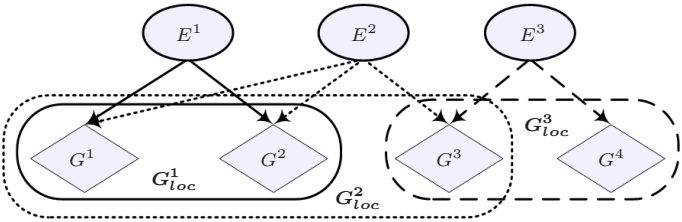


Figura 19 – Composição de plantas locais no CML.

notam o compartilhamento de eventos entre eles. Note que as plantas locais G^1_{loc}, G^2_{loc} e G^3_{loc}, construídas respectivamente para as especificações E^1, E^2 e E^3, são compostas por todas e somente as plantas que possuem eventos em comum com a especificação. Essa ideia pode ser formalizada como segue.

Definição 9 *Para um SED cuja planta é modelada por autômatos assíncronos G^j, $j \in J = \{1, \cdots, n\}$, definidos em Σ^{G^j}, seja E^i, $i \in I = \{1, \cdots, m\}$ um conjunto de especificações modeladas em Σ^{E^i}. Para cada E^i, defina uma planta local G^i_{loc} tal que:*

$$\begin{aligned} G^i_{loc} &= \Big\|^{j \in J_i} G^j \\ J_i &= \{j \in J \mid \Sigma^{G^j} \cap \Sigma^{E^i} \neq \emptyset\}. \end{aligned}$$

O CML define, então, o uso de G^i_{loc} na síntese de supervisores locais. Na sequência, o PCS é reintroduzido para incorporar essa ideia.

Problema 3 - PCS Modular Local (PCS-ML): *Dada uma planta local G^i_{loc} para uma especificação $E^i \subseteq \Sigma^{E^{i^*}} \subseteq \Sigma^*$, definindo um comportamento desejado $K^i_{loc} = E^i \cap L^\omega(G^i_{loc})$, encontre um supervisor local não-bloqueante S^i_{loc}, tal que $L^\omega(S^i_{loc}/G^i_{loc}) \subseteq K^i_{loc}$.*

Dois aspectos relevantes surgem no contexto do PCS-ML. O primeiro é a necessidade de se conhecer as implicações do uso de uma planta local (parcial) para obter um supervisor S^i_{loc}, em substituição de uma planta monolítica. O segundo aspecto remete à relação global de equivalência entre o conjunto de soluções locais para o PCS-ML e uma solução monolítica para o PCS. Esses dois aspectos são apresentados a seguir.

5.2.1 Uma análise local do PCS-ML

Como cada supervisor S^i_{loc} é obtido a partir de uma planta local, é fundamental investigar a equivalência de S^i_{loc} em relação a um supervisor correspondente, caso esse fosse obtido usando o modelo completo da planta. Mostrar tal equivalência passa, sobretudo, por mostrar que o "complemento" da planta, ou seja, os modelos que não fazem parte da planta local, não é de fato necessário à síntese. Essa equivalência é formalizada a seguir e se apoia no conceito de planta complementar.

Definição 10 - *Para E^i, G^j e G^i_{loc}, como na Definição 9, seja C^i tal que:*

$$C^i = \{j \in J \mid \Sigma^{G^j} \cap \Sigma^{E^i} = \emptyset\}.$$

Uma planta complementar G^i_c é definida para G^i_{loc} como:

$$G^i_c = \begin{cases} \|_{j \in C^i} \; G^j & se \; C^i \neq \emptyset \\ \\ G^i_{loc} & do \; contrário \; . \end{cases}$$

Em prosa, uma planta complementar para G^i_{loc} compõe todo G^j que não faz parte da planta local. Se G^i_{loc} acopla todo G^j, então não há complemento de planta e G^i_c é assumida como sendo a própria planta local. A seguinte proposição mostra o efeito de desconsiderar uma planta complementar da síntese de supervisores locais.

Proposição 6 (QUEIROZ; CURY, 2000a) - *Para E^i e G^i_{loc}, como na Definição 9 e G^i_c, como na Definição 10, é verdade que*

$$\sup\mathcal{C}(E^i, G^i_{loc}) \parallel L^\omega(G^i_c) = \sup\mathcal{C}(E^i, G^i_{loc} \parallel G^i_c).$$

Ou seja, se um modelo de subsistema não é afetado por eventos de uma especificação, então ele pode ser removido da síntese local sem que isso interfira no cálculo do supervisor. O efeito disso, globalmente, é investigado como segue.

5.2.2 Uma análise global do PCS-ML

Ainda que supervisores locais sejam não-bloqueantes, a ação conjunta desses supervisores leva a um comportamento global que pode ser conflitante. Isso porque, sendo cada supervisor local obtido com base em uma especificação particular, sua síntese não leva em conta os de-

mais aspectos de controle, definidos por outras especificações. Assim sendo, cada supervisor observa e controla localmente a tomada de recursos do sistema, pelos subsistemas. Logo, toda vez que um recurso é compartilhado por mais de um subsistema, nasce a possibilidade de conflitos, os quais seriam naturalmente prevenidos pela síntese monolítica. A condição necessária e suficiente para a garantia do não-conflito é:

$$\|_{i=1}^{m} \overline{L^{\omega}(S^i_{loc}/G^i_{loc})} = \overline{\|_{i=1}^{m} L^{\omega}(S^i_{loc}/G^i_{loc})}.$$

Ou seja, sempre que um prefixo é compartilhado pelos supervisores locais, ao menos uma cadeia marcada, contendo aquele prefixo, também é compartilhada.

O teste de não-conflito é a etapa mais complexa para resolver o PCS-ML, já que requer a composição de todos os supervisores locais. Isso se equipara, na verdade, à complexidade para tratar do PCS monolítico, o que reverte parte das vantagens do CML. Algumas técnicas, porém, permitem reduzir a complexidade desse teste. O uso de abstrações de linguagens (PENA; CURY; LAFORTUNE, 2009) e procedimentos de verificação composicional (FLORDAL; MALIK, 2009), por exemplo, permitem que o teste de não-conflito seja equivalentemente processado sem necessitar, entretanto, da plena composição dos supervisores locais.

A Proposição 7 mostra que, se o supervisores locais forem globalmente não-conflitantes, então a ação conjunta desses supervisores leva a um comportamento equivalente ao do supervisor monolítico correspondente.

Proposição 7 (QUEIROZ; CURY, 2000b) *Dados G e K, como no PCS, e G^i_{loc} e E^i, como no PCS-ML, se o conjunto* $\{\sup\mathcal{C}(E^i, G^i_{loc}),\ i = 1, \cdots, m\}$ *for não-conflitante, então*

$$\sup\mathcal{C}(K, G) = \|_{i=1}^{m} \sup\mathcal{C}(E^i, G^i_{loc}).$$

Os casos em que essa condição não é observada requerem políticas de resolução de conflitos. A centralização de módulos locais específicos, por exemplo, é uma alternativa natural para isso. Outras alternativas incluem técnicas de coordenação de eventos (WONG; WONHAM, 1998), de prioridade de execução (CHEN; LAFORTUNE; LIN, 1995), etc.

5.2.3 Complexidade da síntese Modular Local

Seja um SED modelado por p subsistemas com no máximo m estados cada. Então, m^p é o número máximo de estados para o autômato que modela a planta. Ainda, seja esse sistema restrito por e especificações modeladas com no máximo n estados. Como cada especificação restringe apenas uma parte k dos p subsistemas, então o cálculo de um supervisor modular local tem complexidade na ordem de

$$\mathcal{O}[(n.m^k)^2].$$

Comparativamente, no CMC a complexidade é na ordem de

$$\mathcal{O}[(n.m^p)^2]$$

e, no caso monolítico, na ordem de

$$\mathcal{O}[(n^e.m^p)^2].$$

Claramente, a síntese no CML tende a ser computacionalmente mais vantajosa do que no CMC e, consequentemente, do que no procedimento monolítico. De fato, o fator de complexidade p é diretamente dependente do número de subsistemas envolvidos no problema, enquanto que o fator k varia apenas em razão do número de subsistemas afetados pela especificação. Isso torna o porte do sistema, sob certo aspecto, irrelevante no CML.

5.2.4 CML aplicado ao sistema de manufatura

Retoma-se aqui o exemplo do sistema de manufatura que vem sendo discutido ao longo desta tese. No capítulo anterior, o exemplo foi explorado no sentido de ilustrar como uma solução para o PCS-D pode ser obtida por meio de um procedimento mais simples, tanto de modelagem quanto de síntese, ao mesmo tempo em que se preservava a equivalência em relação ao PCS original. Mesmo em face dos claros benefícios, o tratamento do problema era processado monoliticamente, i.e., sobre uma composição única de plantas e especificações.

Nesta seção, parte-se do problema original e obtém-se uma solução de controle, também equivalente à do PCS, mas computada de modo descentralizado, através da resolução do PCS-ML. Posteriormente, retoma-se o exemplo para ilustrar a resolução do PCS-MLD,

a extensão do PCS-ML com distinguidores.

5.2.4.1 Uma solução para o PCS-ML

Considere o sistema de manufatura da Figura 20 do Capítulo 4, reproduzido a seguir.

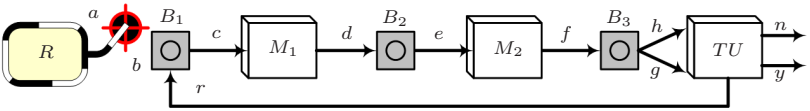


Figura 20 – Pequena fábrica com retrabalho de materiais.

A planta do sistema é modelada pelos autômatos G^1, G^2, G^3 e G^4 da Figura 21.

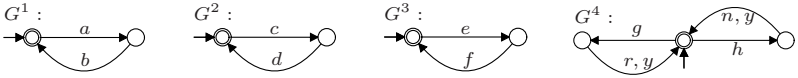


Figura 21 – Modelos para os subsistemas R, M_1, M_2 e TU.

Sobre esse sistema, foram definidas quatro especificações de controle, sendo E^1, E^2 e E^3 destinadas a evitar o *overflow* e o *underflow* de cada respectivo *buffer*, e E^4 visando limitar o número de ciclos de trabalho, o que neste exemplo é assumido ser de, no máximo, 3 (2 retrabalhos). Os autômatos modelando essas especificações são apresentados na Figura 22.

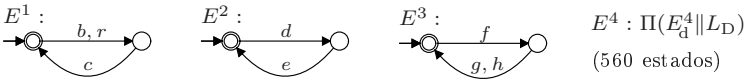


Figura 22 – Modelos para as especificações E^1, E^2, E^3 e E^4.

Por clareza, o modelo de E^4, com 560 estados, não é mostrado. Tal como discutido no Capítulo 4, esse modelo foi novamente obtido via refinamento de eventos, com projeção, *a posteriori*, em Σ. Resolver o PCS-ML consiste, então, em obter um supervisor local S^i_{loc} para cada uma das especificações E^i, $i \in I = \{1, 2, 3, 4\}$, usando para isso uma planta G^i_{loc}, construída conforme a Definição 9. As plantas locais e os demais autômatos usados na síntese, são assim compostos:

- Plantas locais:
  - $G^1_{loc} = G^1 \| G^2 \| G^4$;
  - $G^2_{loc} = G^2 \| G^3$;
  - $G^3_{loc} = G^3 \| G^4$;
  - $G^4_{loc} = G^1 \| G^2 \| G^3 \| G^4$.
- Especificações locais:
  - $K^i_{loc} = E^i \| G^i_{loc}$, para $i \in I$, é um modelo para $L^\omega(E^i \| G^i_{loc})$.
- Supervisores locais:
  - V^i_{loc}, para $i \in I$, é um autômato para $\sup\mathcal{C}(E^i, G^i_{loc})$.

A Tabela 6 apresenta a estatística da síntese de cada um dos supervisores locais. A resolução do PCS também é apresentada, para efeitos comparativos.

Tabela 4 – Uma solução para o PCS-ML.

$i \in I$	G^i_{loc}	E^i	K^i_{loc}	V^i_{loc}
1	12	2	24	14
2	4	2	8	6
3	6	2	12	9
4	12	**560**	1.872	1.656
Teste de não-conflito: $\|_{i\in I}\, V^i_{loc} =$				276
Resolução do PCS				
Monolítico	$G = 24$	$E = 1080$	$K = 7.220$	V =212

Nesse exemplo em particular, o teste de não-conflito falha, i.e., $(\|_{i=1}^m\ V^i_{loc}) \neq V$. Note que V^1_{loc} é um supervisor local que previne o *overflow* e o *underflow* no *buffer* B_1 desconsiderando, para isso, o restante do comportamento do sistema. Sendo assim, é possível que V^1_{loc} permita ocupar B_1 tão logo ele esteja livre. Agora suponha que os demais *buffers* também estejam ocupados. Nesse caso, nenhuma máquina pode começar a operar, pois os eventos de fim de operação são não-controláveis e não há recurso algum de *buffer* disponível. Então, um bloqueio se caracteriza (a menos que se trate do último ciclo, no qual o fim de operação de TU não leva a um retrabalho). Contudo, um supervisor monolítico não-bloqueante reserva o recurso B_1 toda a vez que houver a possibilidade de que uma peça do retrabalho necessite desse recurso.

A resolução efetiva deste bloqueio foge ao escopo desta tese, mas pode ser feita através da centralização da síntese de módulos específicos. Objetiva-se, na sequência, manter a condição de conflito estável sob as diferentes abordagens de síntese a serem aplicadas no exemplo, i.e., a seguir busca-se obter um mesmo modelo com 276 estados ao compor os supervisores locais.

Note que a operação de síntese mais complexa no PCS-ML é a computação de V^4_{loc}, envolvendo um autômato K^4_{loc} com 1.872 estados. Contudo, no PCS, a síntese é conduzida sobre um modelo K com 7.220 estados, o que, de certa forma, dimensiona os ganhos que podem ser alcançados pela substituição do procedimento monolítico pelo modular local.

O principal inconveniente é que a modelagem de E^4, com eventos em Σ, é uma tarefa irrealizável, na prática. Isso sugere a associação de distinguidores ao CML, o que permitiria resolver todos os problemas locais sobre um alfabeto refinado. A seção seguinte formaliza essa ideia e ilustra o efeito disso sobre esse mesmo exemplo.

5.3 CML COM DISTINGUIDORES

Dada a estrutura do CML, como apresentada na Seção 5.2, considere a introdução de um distinguidor para simplificar alguma tarefa de modelagem. A questão que surge, então, é: qual o efeito disso na obtenção de supervisores locais? Esta seção visa responder a essa pergunta.

Inicialmente, o CML é estendido para considerar um alfabeto refinado e a introdução de um modelo distinguidor. Posteriormente, o PCS-ML é reintroduzido à luz desse novo cenário. Então, alguns resultados são apresentados para mostrar a relação entre soluções locais e globais de controle, com e sem distinguidores. Por fim, o exemplo é retomado para ilustrar a abordagem.

Definição 11 *Para um conjunto de especificações $I = \{1, \cdots, m\}$ e um conjunto de plantas $J = \{1, \cdots, n\}$, sejam:*

- G^j_{a}, $j \in J$, *modelos assíncronos dos subsistemas de um SED, tal que $L(G^j_{\mathrm{a}}) = \Pi^{-1}(L(G^j))$, com $\Delta^{G^j_{\mathrm{a}}} \subseteq \Delta$ e $\Sigma^{G^j} \subseteq \Sigma$;*

- E^i_{d}, $i \in I$, *especificações modeladas em $\Delta^{E^i_{\mathrm{d}}} \subseteq \Delta$, expressando requisitos equivalentes a E^i no PCS, com $\Sigma^{E^i} \subseteq \Sigma$;*

- H_{D}, *o modelo de um distinguidor* D, *definido sobre* $\Delta^{H_{\mathrm{D}}}$, *tal que* $L(H_{\mathrm{D}}) = L_{\mathrm{D}} \subseteq \Delta^*$.

O acoplamento entre G_{a}^j *e* H_{D} *decorre de eventos compartilhados entre eles, o que é sintetizado pelos autômatos* $G_{\mathrm{d}}^{'j}$, $j \in J$, *construídos como segue:*

- $G_{\mathrm{d}}^{'j} = \begin{cases} G_{\mathrm{a}}^j \parallel H_{\mathrm{D}} & \text{se } (\Delta^{G_{\mathrm{a}}^j} \cap \Delta^{H_{\mathrm{D}}}) \neq \emptyset; \\ \\ G_{\mathrm{a}}^j & \text{do contrário.} \end{cases}$

  Então, para $L = \{1, \cdots, p\}$, *defina:*

- G_{d}^l, $l \in L$, *como a estrutura assíncrona dos modelos* $G_{\mathrm{d}}^{'j}$, *i.e., construída através da composição de modelos síncronos em* J.

Uma planta local ${G_{\mathrm{d}}}_{loc}^{\,i}$, *usada para sintetizar um supervisor para cada especificação* E_{d}^i, *é construída como segue:*

$$\begin{aligned} {G_{\mathrm{d}}}_{loc}^{\,i} &= \Big\|^{l \in L_i} G_{\mathrm{d}}^l \\ &\mid \quad L_i = \{l \in L \mid (\Delta^{G_{\mathrm{d}}^l} \cap \Delta^{E_{\mathrm{d}}^i}) \neq \emptyset\}. \end{aligned}$$

Por fim, o PCS-ML é reintroduzido para incorporar essa nova definição.

Problema 4 - PCS-ML Distinguido (PCS-MLD): *Dadas uma planta local* ${G_{\mathrm{d}}}_{loc}^{\,i}$ *e uma especificação* $E_{\mathrm{d}}^i \subseteq \Delta^{E_{\mathrm{d}}^{i*}} \subseteq \Delta^*$, *definindo um comportamento desejado* ${K_{\mathrm{d}}}_{loc}^{\,i} = E_{\mathrm{d}}^i \cap L^{\omega}({G_{\mathrm{d}}}_{loc}^{\,i})$, *encontre um supervisor local não-bloqueante* ${S_{\mathrm{d}}}_{loc}^{\,i}$, *tal que* $L^{\omega}({S_{\mathrm{d}}}_{loc}^{\,i}/{G_{\mathrm{d}}}_{loc}^{\,i}) \subseteq {K_{\mathrm{d}}}_{loc}^{\,i}$.

Encontrar uma solução local para o PCS-MLD tende a ser mais complexo do que no PCS-ML. Isso porque, ao serem distinguidas por H_{D}, as plantas refinadas passam a ter mais chances de ser acopladas em ${G_{\mathrm{d}}}_{loc}^{\,i}$, do que as não refinadas em G_{loc}^i.

Por exemplo, suponha que uma planta G^j não compartilha eventos com uma dada especificação E^i e, portanto, não compõe G_{loc}^i. Agora, seja G_{a}^j um modelo para $\Pi^{-1}(L(G^j))$, o qual também não compartilha eventos com o correspondente E_{d}^i. Porém, G_{a}^j é distinguido por H_{D}. Como é bastante provável que H_{D} seja síncrono com algum evento de E_{d}^i, então G_{a}^j tende a compor ${G_{\mathrm{d}}}_{loc}^{\,i}$ indiretamente, por efeito do modelo distinguidor. O que se observa, em geral, é

$$L({G_{\mathrm{d}}}_{loc}^{\,i}) \subseteq \mathrm{D}(L(G_{loc}^i)) \qquad \text{e} \qquad {K_{\mathrm{d}}}_{loc}^{\,i} \subseteq \mathrm{D}(K_{loc}^i).$$

A seguir, um exemplo ilustra esse efeito.

5.3.1 CML refinado aplicado ao sistema de manufatura

Considere o sistema de manufatura mostrado na Figura 20 da Subseção 5.2.4, para o qual foram definidas as especificações E^1, E^2, E^3 e E^4. O problema complexo de modelagem reside na especificação E^4, e é aqui modelado por meio de um alfabeto refinado. As demais especificações mantêm a mesma estrutura, sendo apenas projetadas no novo alfabeto. A Figura 23 apresenta esses modelos.

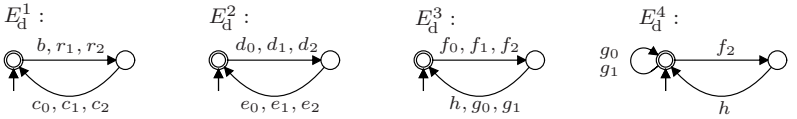


Figura 23 – Modelos das especificações em Δ.

É assumido que as estruturas de refinamentos e de distinções de eventos são como as apresentadas no Capítulo 4. Por clareza, a planta refinada é reapresentada na Figura 24.

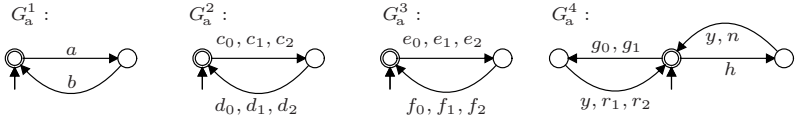


Figura 24 – Modelos dos subsistemas R, M_1, M_2 e TU em Δ.

Resolver o PCS-MLD consiste, então, em obter um supervisor local $S_{\mathrm{d}}{}^{i}_{loc}$ para cada uma das especificações E^i_{d}, $i \in I = \{1, 2, 3, 4\}$, usando para isso uma planta local $G_{\mathrm{d}}{}^{i}_{loc}$, definida conforme a Definição 11. As plantas locais e os demais modelos usados na síntese refinada são compostos como segue.

- Relações entre plantas e distinguidor:
  - $G'^1_{\mathrm{d}} = G^1_{\mathrm{a}}$;
  - $G'^2_{\mathrm{d}} = G^2_{\mathrm{a}} \parallel H_{\mathrm{D}}$;
  - $G'^3_{\mathrm{d}} = G^3_{\mathrm{a}} \parallel H_{\mathrm{D}}$;
  - $G'^4_{\mathrm{d}} = G^4_{\mathrm{a}} \parallel H_{\mathrm{D}}$.

- Representação assíncrona dos modelos $G_{\mathrm{d}}^{\prime\, i}$:
  - $G_{\mathrm{d}}^{1} = G_{\mathrm{d}}^{\prime\, 1} = G_{\mathrm{a}}^{1}$;
  - $G_{\mathrm{d}}^{2} = G_{\mathrm{d}}^{\prime\, 2} \parallel G_{\mathrm{d}}^{\prime\, 3} \parallel G_{\mathrm{d}}^{\prime\, 4} = G_{\mathrm{a}}^{2} \parallel G_{\mathrm{a}}^{3} \parallel G_{\mathrm{a}}^{4} \parallel H_{\mathrm{D}}$.
- Plantas locais refinadas:
  - $G_{\mathrm{d}}{}_{loc}^{1} = G_{\mathrm{d}}^{1} \parallel G_{\mathrm{d}}^{2}$;
  - $G_{\mathrm{d}}{}_{loc}^{2} = G_{\mathrm{d}}{}_{loc}^{3} = G_{\mathrm{d}}{}_{loc}^{4} = G_{\mathrm{d}}^{2}$.
- Especificações locais:
  - $K_{\mathrm{d}}{}_{loc}^{i} = E_{\mathrm{d}}^{i} \| G_{\mathrm{d}}{}_{loc}^{i}$ é um modelo para $L^{\omega}(E_{\mathrm{d}}^{i} \| G_{\mathrm{d}}{}_{loc}^{i})$.
- Supervisores locais:
  - $V_{\mathrm{d}}{}_{loc}^{i}$ é um modelo para $\sup\mathcal{C}(E_{\mathrm{d}}^{i}, G_{\mathrm{d}}{}_{loc}^{i})$.

A Tabela 6 apresenta a estatística da síntese de cada supervisor local refinado. A resolução do PCS-D é reapresentada para efeitos comparativos.

Tabela 5 – Uma solução para o PCS-MLD.

$i \in I$	$G_{\mathrm{d}}{}_{loc}^{i}$	E_{d}^{i}	$K_{\mathrm{d}}{}_{loc}^{i}$	$V_{\mathrm{d}}{}_{loc}^{i}$
1	2.160	2	2.928	1.044
2	1.080	2	1.428	600
3	1.080	2	1.800	900
4	1.080	2	1.872	1.656
Teste de não-conflito: $\|_{i\in I}\ V_{\mathrm{d}}{}_{loc}^{i} =$				276
Resolução do PCS-D				
Monolítico	$G_{\mathrm{d}} = 2.160$	$E_{\mathrm{d}} = 12$	$K_{\mathrm{d}} = 7.220$	V_{d} =212

Primeiramente, note que E_{d}^{4} é agora modelada por um autômato de apenas 2 estados. Além disso, a condição global de conflito entre os supervisores locais permanece inalterada, i.e., $\Pi(\|_{i=1}^{m}\ V_{\mathrm{d}}{}_{loc}^{i}) = \|_{i=1}^{m}\ V_{loc}^{i}$, como era o objetivo. Por fim, a síntese no PCS-MLD é mais simples, em comparação à monolítica, já que a operação mais complexa envolve um autômato com 2.928 estados, ao invés de 7.220, como no PCS-D.

No entanto, usar o distinguidor para resolver os problemas que eram originalmente simples, como os dos módulos 1, 2 e 3, agregou complexidade à síntese local. Note que, no PCS-ML, os modelos para

K^1_{loc}, K^2_{loc} e K^3_{loc} continham apenas 24, 8 e 12 estados, respectivamente, e agora, compõem plantas complementares acopladas pelo modelo H_{D}. A abordagem de síntese combinada é apresentada a seguir como uma maneira de contornar esse problema.

5.4 COMBINANDO LOCALMENTE PCS-ML E PCS-MLD

Se por um lado resolver o PCS-MLD é mais complexo do que resolver o PCS-ML, por outro lado distinguidores podem ser necessários para viabilizar o tratamento de alguns problemas de controle. Nesse sentido, esta seção mostra como as vantagens de ambos os métodos podem ser combinadas localmente, de maneira que uma solução global combinada seja equivalente à do PCS-ML original.

Plantas complementares são redefinidas, agora em um novo contexto, para formalizar a relação entre o PCS-ML e o PCS-MLD.

Definição 12 *Seja E^i, G^j e G^i_{loc}, como no PCS-ML, H_{D} e G^j_{a}, como na Definição 11, e C^i_{d}, definido como segue*

$$C^i_{\mathrm{d}} = \{j \in J \quad | \quad \Sigma^{G^j} \cap \Sigma^{E^i} = \emptyset \quad \wedge \quad \Delta^{G^j_{\mathrm{a}}} \cap \Delta^{H_{\mathrm{D}}} \neq \emptyset\}.$$

Uma planta complementar G^i_{cd}, para G^i_{loc} e.r.a. ${G_{\mathrm{d}}}^i_{loc}$, é tal que

$$G^i_{\mathrm{cd}} = \begin{cases} \|_{j \in C^i_{\mathrm{d}}} \; G^j & \text{se } C^i_{\mathrm{d}} \neq \emptyset; \\ \\ G^i_{loc} & \text{do contrário.} \end{cases}$$

Ou seja, G^i_{cd} é composta por toda planta não-refinada G^j, que não compõe G^i_{loc}, mas cuja planta refinada correspondente, G^j_{a}, compõe ${G_{\mathrm{d}}}^i_{loc}$ por efeito do modelo H_{D}. A partir da Definição 12, fica fácil notar que

$$L({G_{\mathrm{d}}}^i_{loc}) = \mathrm{D}(L(G^i_{loc}) \| L(G^i_{\mathrm{cd}})) \qquad \text{e} \qquad {K_{\mathrm{d}}}^i_{loc} = \mathrm{D}(K^i_{loc} \| L^\omega(G^i_{\mathrm{cd}})).$$

A proposição seguinte usa essa ideia para mostrar a relação local entre o PCS-ML e o PCS-MLD.

Proposição 8 (Equivalência local) *: Seja E^i e G^i_{loc}, como no PCS-ML, E^i_{d} e ${G_{\mathrm{d}}}^i_{loc}$, como no PCS-MLD e G^i_{cd}, como na Definição 12. Então,*

$$\Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, {G_{\mathrm{d}}}^i_{loc})) = \sup\mathcal{C}(E^i, G^i_{loc}) \| L^\omega(G^i_{\mathrm{cd}}).$$

Prova:

$$\begin{aligned}
\Pi(\sup\mathcal{C}(E_{\rm d}^i, G_{\rm d}{}_{loc}^{\,i})) &= \Pi(\sup\mathcal{C}(E_{\rm d}^i, {\rm D}(G_{loc}^i \parallel G_{\rm cd}^i))) \\
&= \Pi(\sup\mathcal{C}(E_{\rm d}^i, {\rm D}(G_{loc}^i) \parallel {\rm D}(G_{\rm cd}^i))) \\
&= \Pi(\sup\mathcal{C}(E_{\rm d}^i, {\rm D}(G_{loc}^i))) \parallel \Pi({\rm D}(L^\omega(G_{\rm cd}^i))) \\
&\quad \text{(pela Proposição 6)} \\
&= \sup\mathcal{C}(E^i, G_{loc}^i) \parallel L^\omega(G_{\rm cd}^i),
\end{aligned}$$

provando a Proposição 8. ∎

Os resultados da Proposição 8 sugerem que cada supervisor local possa ser computado usando distinguidores ou não, sempre que apropriado. O Teorema 3 mostra que essa combinação local não altera o comportamento global, ou seja, o resultado da síntese que combina o PCS-ML e o PCS-MLD leva a um comportamento global equivalente à solução para o PCS-ML.

Teorema 3 (Equivalência global) *: Para E^i e G_{loc}^i, como no PCS-ML e $E_{\rm d}^i$ e $G_{\rm d}{}_{loc}^{\,i}$, como no PCS-MLD, seja $I' \subset I = \{1, \cdots, m\}$ e $I'' = I - I'$. Então,*

$$\Big\|_{i\in I} \sup\mathcal{C}(E^i, G_{loc}^i) \underset{=}{} (\Big\|_{i\in I'} \sup\mathcal{C}(E^i, G_{loc}^i)) \parallel (\Big\|_{i\in I''} \Pi(\sup\mathcal{C}(E_{\rm d}^i, G_{\rm d}{}_{loc}^{\,i}))).$$

Prova:

$$\begin{aligned}
&\Big\|_{i\in I'} \sup\mathcal{C}(E^i, G_{loc}^i) \parallel \Big\|_{i\in I''} \Pi(\sup\mathcal{C}(E_{\rm d}^i, G_{\rm d}{}_{loc}^{\,i})) \\
= &\Big\|_{i\in I'} \sup\mathcal{C}(E^i, G_{loc}^i) \parallel \Big\|_{i\in I''} \sup\mathcal{C}(E^i, G_{loc}^i) \parallel L^\omega(G_{\rm cd}^i) \\
&\text{(pela Proposição 8)} \\
= &\Big\|_{i\in I} \sup\mathcal{C}(E^i, G_{loc}^i) \parallel L^\omega(G_{\rm cd}^i) \\
= &\Big\|_{i\in I} \sup\mathcal{C}(E^i, G_{loc}^i),
\end{aligned}$$

provando o Teorema 3. ∎

O Teorema 3 mostra que o conjunto de especificações pode ser

particionado em dois diferentes subconjuntos, a fim de que possam ser individualmente resolvidos conforme o PCS-ML e o PCS-MLD. As especificações que originalmente não requerem o uso de um distinguidor compõem o subconjunto I', enquanto as complexas, para as quais um modelo depende do distinguidor, definem o subconjunto I''.

Na abordagem combinada, tanto a complexidade de síntese quanto as soluções de controle são equivalentes em relação ao PCS-ML, local e globalmente.

5.4.1 Síntese combinada aplicada ao sistema de manufatura

Retoma-se aqui o exemplo que vem sendo abordado neste capítulo, no sentido de ilustrar a síntese local combinada. Seguindo a ideia do Teorema 3, considere particionar o conjunto de especificações $I = \{1, 2, 3, 4\}$, tal que $I' = \{1, 2, 3\} \subset I$ contém as especificações simples, que podem ser resolvidas conforme o PCS-ML, enquanto $I'' = \{4\} \subset I$ contém a especificação complexa, que requer o uso do distinguidor.

Então, as plantas locais e os demais modelos usados na síntese combinada são compostos conforme a resolução de cada um dos problemas mostrados anteriormente (Subseções 5.2.4 e 5.2.4), cujos resultados são sumarizados na Tabela 6.

Tabela 6 – Uma solução para o PCS-ML obtida via síntese combinada.

$i \in I'$	G^i_{loc}	E^i	K^i_{loc}	V^i_{loc}
1	12	2	24	14
2	4	2	8	6
3	6	2	12	9
$i \in I''$	${G_{\mathrm{d}}}^i_{loc}$	E^i_{d}	${K_{\mathrm{d}}}^i_{loc}$	${V_{\mathrm{d}}}^i_{loc}$
4	1.080	2	1.872	1.656
Teste de não-conflito: $(\|\|_{i=1}^{3} V^i_{loc}) \parallel (\Pi({V_{\mathrm{d}}}^4_{loc})) =$				276

Ao usar o distinguidor para tratar somente do problema no subconjunto I'', evita-se complexificar a síntese em I'. Ao mesmo tempo, viabiliza-se a modelagem de E^4_{d} e, logo, a computação de ${V_{\mathrm{d}}}^4_{loc}$.

Finalmente, o teste de não-conflito pode ser naturalmente executado, Π-*projetando-se* o supervisor ${V_{\mathrm{d}}}^4_{loc}$ no alfabeto Σ, tal que a composição $(\|_{i=1}^{3} V^i_{loc}) \parallel (\Pi({V_{\mathrm{d}}}^4_{loc}))$ possui, novamente, 276 estados.

5.5 CML COM APROXIMAÇÕES

Combinar o PCS-ML e o PCS-MLD é vantajoso por duas razões principais. A primeira, é que o esforço computacional de síntese se equipara ao PCS-ML, que é muito menor do que, por exemplo, em uma operação centralizada ou no CMC. A segunda, é que os benefícios de modelagem são aproveitados localmente sem influenciar desnecessariamente outros módulos.

Contudo, essa abordagem não reduz o esforço de síntese em relação ao PCS-ML original. Vantagens computacionais podem ser adicionadas a módulos da síntese combinada, através do uso de aproximações, para lidar com os problemas pertencentes ao subconjunto I'', i.e., aqueles para os quais um distinguidor é usado.

As aproximações foram apresentadas no Capítulo 4, no contexto do uso de distinguidores na síntese monolítica. Nesta seção, elas são associadas ao CML e as implicações disso são discutidas. Os resultados a serem apresentados permitem preservar a controlabilidade, o não-bloqueio e a máxima permissividade de uma solução obtida usando aproximações, tanto local quanto globalmente. Começa-se estendendo o CML para envolver aproximações, conforme a seguir.

Definição 13 *Para um conjunto de plantas* $G^j,\ j \in J = \{1, \cdots, n\}$ *e de especificações* $E^i,\ i \in I = \{1, \cdots, m\}$*, sejam:*

- $G^j_{\mathrm{a}},\ j \in J$*, modelos assíncronos dos subsistemas de um SED, tal que* $L(G^j_{\mathrm{a}}) = \Pi^{-1}(L(G^j))$*, com* $\Delta^{G^j_{\mathrm{a}}} \subseteq \Delta$ *e* $\Sigma^{G^j} \subseteq \Sigma$*;*
- $E^i_{\mathrm{d}},\ i \in I$*, especificações modeladas em* $\Delta^{E^i_{\mathrm{d}}} \subseteq \Delta$*;*
- H_{D}*, o modelo de um distinguidor preditível* D*, com* $L(H_{\mathrm{D}}) = L_{\mathrm{D}} \subseteq \Delta^*$*;*
- H_{Da}*, o modelo de um distinguidor* $\mathrm{D_a}$ *tal que* $L(H_{\mathrm{Da}}) = L_{\mathrm{Da}} \supseteq L_{\mathrm{D}}$ *e* $\Delta^{H_{\mathrm{D}}} = \Delta^{H_{\mathrm{Da}}}$*.*

O acoplamento entre G^j_{a} *e* H_{Da} *decorre dos eventos compartilhados entre esses dois modelos, o que é sintetizado pelos autômatos* $G'_{\mathrm{a}\,j},\ j \in J$*, construídos como segue:*

- $$G'_{\mathrm{a}\,j} = \begin{cases} G^j_{\mathrm{a}} \parallel H_{\mathrm{Da}} & \text{se } (\Delta^{G^j_{\mathrm{a}}} \cap \Delta^{H_{\mathrm{Da}}}) \neq \emptyset; \\ G^j_{\mathrm{a}} & \text{do contrário.} \end{cases}$$

Então, para $L = \{1, \cdots, p\}$*, defina:*

- G_{a}^{l}, $l \in L$, *como a estrutura assíncrona dos modelos* $G'_{\mathrm{a}\,j}$, *i.e.,* G_{a}^{l} *é construída pela composição de modelos síncronos em* J.

Uma planta local $G_{\mathrm{a}}{}^{i}_{loc}$, *usada para sintetizar um supervisor para cada especificação* E_{d}^{i}, $i \in I$, *é construída como segue:*

$$\begin{aligned} G_{\mathrm{a}}{}^{i}_{loc} &= \Big\|^{l \in L_i} G_{\mathrm{a}}^{l} \\ &\mid \quad L_i = \{l \in L \mid (\Delta^{G_{\mathrm{a}}^{l}} \cap \Delta^{E_{\mathrm{d}}^{i}}) \neq \emptyset\}. \end{aligned}$$

A partir da Definição 13, a relação entre as plantas locais $G_{\mathrm{a}}{}^{i}_{loc}$ e $G_{\mathrm{d}}{}^{i}_{loc}$, em termos de acoplamento de plantas modulares, pode ser mostrada como na Proposição 9.

Proposição 9 *Sejam* E_{d}^{i}, G_{a}^{j}, $G_{\mathrm{d}}{}^{i}_{loc}$ *e* $G_{\mathrm{a}}{}^{i}_{loc}$, *como nas definições 11 e 13. Então,*

$$\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{d}}{}^{i}_{loc}} \neq \emptyset \quad \leftrightarrow \quad \Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} \neq \emptyset.$$

Prova: Por definição, $\Delta^{H_{\mathrm{D}}} = \Delta^{H_{\mathrm{Da}}}$. Suponha que exista uma planta G_{a}^{j}, tal que:

$$\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{d}}{}^{i}_{loc}} \neq \emptyset \quad \wedge \quad \Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} = \emptyset \tag{5.1}$$

ou

$$\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{d}}{}^{i}_{loc}} = \emptyset \quad \wedge \quad \Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} \neq \emptyset. \tag{5.2}$$

Em (5.1), $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{d}}{}^{i}_{loc}} \neq \emptyset$ implica que $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{H_{\mathrm{D}}} \neq \emptyset$ pois, do contrário, $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{E_{\mathrm{d}}^{i}} \neq \emptyset$, o que implicaria $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} \neq \emptyset$, contradizendo (5.1). Mas, então, $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{H_{\mathrm{Da}}} = \Delta^{H_{\mathrm{D}}} \neq \emptyset$ e, portanto, $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} \neq \emptyset$, e (5.1) é falso. Pelo mesmo raciocínio, $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{a}}{}^{i}_{loc}} \neq \emptyset$ em (5.2) leva a $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{H_{\mathrm{Da}}} \neq \emptyset$, implicando que $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{H_{\mathrm{D}}} = \Delta^{H_{\mathrm{Da}}} \neq \emptyset$ e, então, $\Delta^{G_{\mathrm{a}}^{j}} \cap \Delta^{G_{\mathrm{d}}{}^{i}_{loc}} \neq \emptyset$, contradizendo (5.2). ∎

A Proposição 9 mostra que $G_{\mathrm{d}}{}^{i}_{loc}$ e $G_{\mathrm{a}}{}^{i}_{loc}$ compõem localmente as mesmas plantas modulares. Note, contudo, que isso não implica que a complexidade das plantas locais seja a mesma. De fato, como $L_{\mathrm{D}} \subseteq L_{\mathrm{Da}}$, então H_{Da} tende a ser modelado por um autômato mais simples do que H_{D}, em termos de número de estados. Isso implica que, além de $L(G_{\mathrm{d}}{}^{i}_{loc}) \subseteq L(G_{\mathrm{a}}{}^{i}_{loc})$, também o autômato $G_{\mathrm{a}}{}^{i}_{loc}$ é, em geral,

mais simples do que $G_{\mathrm{d}}{}^{i}_{loc}$.

Em face disso, a Proposição 10 mostra outra relação entre as plantas locais, desta vez em termos de Σ-*Preservância* de eventos.

Proposição 10 *Sejam* $G_{\mathrm{d}}{}^{i}_{loc}$ *e* $G_{\mathrm{a}}{}^{i}_{loc}$*, respectivamente como nas definições 11 e 13, tal que* $L(G_{\mathrm{d}}{}^{i}_{loc}) \subseteq L(G_{\mathrm{a}}{}^{i}_{loc})$*. Então,*

$$L(G_{\mathrm{d}}{}^{i}_{loc}) \text{ é } \Sigma\text{-Preservante } \textit{ e.r.a. } L(G_{\mathrm{a}}{}^{i}_{loc}).$$

Prova: Segue diretamente de $L(G_{\mathrm{d}}{}^{i}_{loc}) \subseteq L(G_{\mathrm{a}}{}^{i}_{loc})$ e da aplicação da Proposição 5 (Capítulo 4) a um caso local. ■

Resta, então, investigar o efeito de substituir uma planta $G_{\mathrm{d}}{}^{i}_{loc}$ por uma aproximação $G_{\mathrm{a}}{}^{i}_{loc}$ na síntese local do PCS-MLD. Os resultados a serem apresentados na sequência estendem o uso de aproximações (Capítulo 4) para um caso local. A partir disso, tem-se uma condição suficiente para determinar se uma solução local, obtida através de uma aproximação, é ótima.

Proposição 11 *Sejam* $G_{\mathrm{d}}{}^{i}_{loc}$ *e* E^{i}_{d}*, como no PCS-MLD, e* $G_{\mathrm{a}}{}^{i}_{loc}$ *uma a aproximação local para* $G_{\mathrm{d}}{}^{i}_{loc}$*. Se* $\sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc})$ *e* L_{D} *forem não-conflitantes, então um supervisor* $S_{\mathrm{a}}{}^{i}_{loc}$ *que implementa* $\sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}}$ *é uma solução local para o PCS-MLD. Além disso,*

$$\begin{aligned}\sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}} &\subseteq \sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{d}}{}^{i}_{loc}) \\ &\subseteq \sup\mathcal{P}_{\Sigma_{\mathrm{u}}}(E^{i}_{\mathrm{d}}, L(G_{\mathrm{a}}{}^{i}_{loc})) \cap L_{\mathrm{D}}.\end{aligned}$$

Prova: Segue diretamente dos resultados do Capítulo 4, aplicados a um caso local. ■

Da Proposição 11, tem-se que $S_{\mathrm{a}}{}^{i}_{loc}$ é uma solução local, controlável e não-bloqueante para o PCS-MLD. Quando, além disso, as condições do Corolário 3 são satisfeitas, então a solução também é localmente ótima.

Corolário 3 *Sobre as mesmas condições da Proposição 11,*

$$\sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) = \sup\mathcal{P}_{\Sigma_{\mathrm{u}}}(E^{i}_{\mathrm{d}}, L(G_{\mathrm{a}}{}^{i}_{loc}))$$

$$\Rightarrow$$

$$\sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{d}}{}^{i}_{loc}) = \sup\mathcal{C}(E^{i}_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}}.$$

Caso as condições do Corolário 3 não sejam satisfeitas ao se usar uma dada planta local $G_{\mathrm{a}}{}^{i}_{loc}$, a aproximação pode ser reconstruída de maneira a parcialmente considerar o modelo H_{D}. Quando o Corolário 3 é satisfeito, então pode-se enunciar os seguintes resultados.

Proposição 12 *Sobre as condições da Proposição 11 e do Corolário 3, é verdade que*

$$\Big|\Big|_{i\in I} \sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{d}}{}^{i}_{loc}) \quad = \quad \Big|\Big|_{i\in I} (\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}}).$$

Prova:

$$\begin{aligned}\Big|\Big|_{i\in I} \sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{d}}{}^{i}_{loc}) \quad &= \quad \Big|\Big|_{i\in I} \sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc} \| H_{\mathrm{D}}) \\ &= \quad \Big|\Big|_{i\in I} \sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}} \\ &\qquad \big(\text{pela Proposição 11 e Corolário 3}\big).\end{aligned}$$

■

Ou seja, se as soluções locais aproximadas satisfazem individualmente as condições do Corolário 3, então elas resolvem globalmente o PCS-MLD. Além do mais, nesse caso, o seguinte corolário pode ser enunciado.

Corolário 4 *Sobre as condições do Corolário 3, também é verdade que*

$$\Big|\Big|_{i\in I} \Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}}) \quad = \quad \Big|\Big|_{i\in I} \sup\mathcal{C}(E^i, G^i_{loc}).$$

Prova: Decorre da Proposição 12 e do Teorema 3. ■

A combinação dos corolários 3 e 4 mostra que o uso de aproximações locais resolve também o PCS-ML original. No entanto, a Proposição 8 mostra que

$$\Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{d}}{}^{i}_{loc})) = \sup\mathcal{C}(E^i, G^i_{loc} \| L^{\omega}(G^i_{\mathrm{cd}}))$$

e, pressupondo o Corolário 3, então

$$\Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^{i}_{loc}) \cap L_{\mathrm{D}}) = \sup\mathcal{C}(E^i, G^i_{loc} \| L^{\omega}(G^i_{\mathrm{cd}})).$$

Ou seja, a síntese com aproximações ainda é localmente mais complexa do que no PCS-ML. Uma alternativa seria usar, na síntese combinada, os módulos locais resolvidos com aproximações, o que é investigado a seguir.

5.5.1 Combinando CML, Distinguidores e Aproximações

Mostra-se, aqui, como combinar as vantagens do PCS-ML, do PCS-MLD e das aproximações, para resolver um mesmo problema de controle. A abordagem se assemelha à síntese combinada, apresentada anteriormente, mas a estende no sentido de definir uma nova partição, I''', do conjunto de especificações I. $I''' \subseteq I''$ é definido para permitir separar de I'' certas especificações, as quais são então tratadas usando aproximações.

Em um caso geral, poderia ser assumido que $I'' = I'''$. No entanto, o fato de que a construção de uma aproximação insere um passo adicional na resolução do problema de controle, e que então esse passo deveria ser evitado, sempre que possível, justifica a abordagem de particionamentos adotada a seguir.

Teorema 4 *Seja $I = \{1, \cdots, m\}$, $I' \subset I$, $I'' = I - I'$ e $I''' \subset I''$. Sob as condições do Corolário 3, para os problemas $i \in I'''$, é verdade que*

$$\begin{array}{c} \Big\| \limits_{i \in I} \sup\mathcal{C}(E^i, G^i_{loc}) \\ = \\ \Big\| \limits_{i \in I'} \sup\mathcal{C}(E^i, G^i_{loc}) \\ \Big\| \\ \Big\| \limits_{i \in I'' \setminus I'''} \Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{d}}{}^i_{loc})) \\ \Big\| \\ \Big\| \limits_{i \in I'''} \Pi(\sup\mathcal{C}(E^i_{\mathrm{d}}, G_{\mathrm{a}}{}^i_{loc}) \| L_{\mathrm{D}}). \end{array}$$

Prova 1 *Decorre diretamente das provas Teorema 3 e da Proposição 12.*

O Teorema 4 mostra como integrar os três tipos de soluções locais apresentadas, de modo a resolver o problema original de controle, o PCS-ML.

5.5.2 Nova abordagem combinada aplicada ao sistema de manufatura

No exemplo da síntese combinada, apresentado na Subseção 5.4.1, foi aplicada a ideia do Teorema 3 para particionar o conjunto de especificações $I = \{1, 2, 3, 4\}$ em $I' = \{1, 2, 3\} \subset I$ e $I'' = \{4\} \subset I$. Assim, cada subconjunto foi tratado localmente, sem e com distinguidores, respectivamente.

Agora, considere que $I'' = I'''$ e que o problema 4 é tratado usando-se uma aproximação local $G_{\mathrm{a}\,loc}^{\,4}$ ao invés de usar $G_{\mathrm{d}\,loc}^{\,4}$. Pela Definição 13, tal aproximação é construída distinguindo-se as plantas modulares, através de um distinguidor $\mathrm{D_a}$, tal que $L_{\mathrm{Da}} \supseteq L_{\mathrm{D}}$, onde L_{D} é a linguagem de um distinguidor preditível. Assim, o primeiro passo desse exemplo é definir $\mathrm{D_a}$. Considere inicialmente usar um distinguidor $\mathrm{D}_{\mathrm{a}}^{1}$, tal que $L_{\mathrm{D}_{\mathrm{a}}^{1}} = \Delta^*$. O distinguidor, a planta local e os demais elementos usados na síntese do módulo 4, são assim estruturados:

- $H_{\mathrm{D}_{\mathrm{a}}^{1}} = H_{\Delta}$, tal que $L^{\omega}(H_{\Delta}) = \Delta^*$;
- $G_{\mathrm{a}\,loc}^{1\,4} = G_{\mathrm{a}}^{2} \| G_{\mathrm{a}}^{3} \| G_{\mathrm{a}}^{4} \| H_{\mathrm{D}_{\mathrm{a}}^{1}}$;
- $K_{\mathrm{a}\,loc}^{1\,4} = E_{\mathrm{d}}^{4} \| G_{\mathrm{a}\,loc}^{1\,4}$ é um modelo para $L^{\omega}(E_{\mathrm{d}}^{4} \cap G_{\mathrm{a}\,loc}^{1\,4})$; e
- $V_{\mathrm{a}\,loc}^{1\,4}$ é um modelo para o supervisor $\sup\mathcal{C}(K_{\mathrm{a}\,loc}^{1\,4}, G_{\mathrm{a}\,loc}^{1\,4})$.

A Tabela 7, primeira linha, mostra a estatística de síntese de $V_{\mathrm{a}\,loc}^{1\,4}$. Comparativamente, a última linha traz a síntese de $V_{\mathrm{d}\,loc}^{\,4}$, usando um distinguidor preditível. Note que a síntese de $V_{\mathrm{a}\,loc}^{1\,4}$ é processada sobre um autômato de 24 estados, enquanto $V_{\mathrm{d}\,loc}^{\,4}$ é obtido, no PCS-MLD, a partir de um modelo com 1.872 estados.

Como $\sup\mathcal{C}(K_{\mathrm{a}\,loc}^{1\,4}, G_{\mathrm{a}\,loc}^{1\,4})$ é controlável, e não-bloqueante em relação a L_{D}, então um supervisor $S_{\mathrm{a}\,loc}^{1\,4}$, que implementa $\sup\mathcal{C}(K_{\mathrm{a}\,loc}^{1\,4}, G_{\mathrm{a}\,loc}^{1\,4}) \cap L_{\mathrm{D}}$, é uma solução local para o módulo 4 do PCS-MLD (pela Proposição 11). Nesse caso, o controlador pode ser implementado pela execução concorrente dos autômatos $V_{\mathrm{a}\,loc}^{1\,4}$ e H_{D}. Além disso, ao considerar uma aproximação para resolver esse módulo, a condição global de conflito permanece inalterada (276 estados).

Porém, pode ser mostrado que

$$\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{1\,4}, G_{\mathrm{a}\ loc}^{1\,4}) \neq \sup\mathcal{P}_{\Sigma_{\mathrm{u}}}(K_{\mathrm{a}\ loc}^{1\,4}, G_{\mathrm{a}\ loc}^{1\,4}).$$

De fato, enquanto o autômato $V_{\mathrm{a}\ loc}^{1\,4}\|H_{\mathrm{D}}$ possui 1.248 estados, ambos $V_{\mathrm{d}\ loc}^{4}$ e V_{loc}^{4} possuem 1.656 estados. Isso sugere que a solução local aproximada pode ser mais restritiva. As razões para isso são exatamente as que vem sendo discutidas ao longo deste documento, ou seja, ao omitir o distinguidor da síntese, a ação de controle previne o ingresso simultâneo de peças no sistema, a fim de coordenar adequadamente cada ciclo de retrabalho.

Tabela 7 – Solução para o módulo 4 usando aproximações locais.

Módulo	$G_{\mathrm{a}\ loc}^{1\,4}$	E_{d}^{4}	$K_{\mathrm{a}\ loc}^{1\,4}$	$V_{\mathrm{a}\ loc}^{1\,4}$	$V_{\mathrm{a}\ loc}^{1\,4}\|H_{\mathrm{D}}$
4	12	2	24	18	***1.248***
Não-conflito: $(\|\|_{i\in I'}\ S_{loc}^{i})\ \|\|\ (\Pi(S_{\mathrm{a}\ loc}^{1\,4}\|L_{\mathrm{D}})\ =$					276
Módulo	$G_{\mathrm{a}\ loc}^{2\,4}$	E_{d}^{4}	$K_{\mathrm{a}\ loc}^{2\,4}$	$V_{\mathrm{a}\ loc}^{2\,4}$	$V_{\mathrm{a}\ loc}^{2\,4}\|H_{\mathrm{D}}$
4	24	2	48	42	**1.656**
Não-conflito: $(\|\|_{i\in I'}\ S_{loc}^{i})\ \|\|\ (\Pi(S_{\mathrm{a}\ loc}^{2\,4}\|L_{\mathrm{D}})\ =$					276
Módulo	$G_{\mathrm{d}\ loc}^{4}$	E_{d}^{4}	$K_{\mathrm{d}\ loc}^{4}$	$V_{\mathrm{d}\ loc}^{4}$	
4	1.080	2	1.872	**1.656**	

Nesse sentido, a mesma alternativa adotada no exemplo do Capítulo 4 é considerada, aqui, para melhorar a aproximação, i.e., introduz-se o módulo H_f do distinguidor na síntese. H_f traz informações sobre as distinções das ocorrências do evento f na planta, fazendo com que a ação de controle resultante passe a ser mais permissiva.

Para isso, considere a construção de um novo distinguidor, denominado $\mathrm{D}_{\mathrm{a}}^{2}$, tal que $L_{\mathrm{D}_{\mathrm{a}}^{2}} = L(H_f)\|\Delta^*$. O distinguidor, a planta local e os demais elementos da síntese combinada são modelados como segue:

- $H_{\mathrm{D}_{\mathrm{a}}^{2}} = H_f\|H_{\Delta}$, com $L^{\omega}(H_{\Delta}) = \Delta^*$;
- $G_{\mathrm{a}\ loc}^{2\,4} = G_{\mathrm{a}}^{2}\|G_{\mathrm{a}}^{3}\|G_{\mathrm{a}}^{4}\|H_{\mathrm{D}_{\mathrm{a}}^{2}}$;
- $K_{\mathrm{a}\ loc}^{2\,4} = E_{\mathrm{d}}^{4}\|G_{\mathrm{a}\ loc}^{2\,4}$ é um modelo para $L^{\omega}(E_{\mathrm{d}}^{4} \cap G_{\mathrm{a}\ loc}^{2\,4})$; e
- $V_{\mathrm{a}\ loc}^{2\,4}$ é um modelo para o supervisor $\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4})$.

A Tabela 7, última linha, mostra a estatística de síntese do supervisor $V_{\mathrm{a}\ loc}^{2\,4}$, agora computado sobre um autômato $K_{\mathrm{a}\ loc}^{2\,4}$, com 48

estados. Esse modelo é mais complexo do que $K_{\mathrm{a}\ loc}^{1\,4}$, com 24 estados, mas é muito mais simples do que $K_{\mathrm{d}\,loc}^{\ 4}$, com 1.872 estados.

Do não-conflito entre $\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4})$ e L_{D}, tem-se que um supervisor $S_{\mathrm{a}\ loc}^{2\,4}$, implementando $\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4}) \cap L_{\mathrm{D}}$, é uma solução local para o módulo 4 do PCS-MLD. Também, ao considerar essa nova solução aproximada, a condição de conflito entre os supervisores locais, permanece globalmente inalterada (276 estados). Ademais, agora tem-se que

$$\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4}) = \sup\mathcal{P}_{\Sigma_{\mathrm{u}}}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4}),$$

o que pode ser checado pelo número de estados dos autômatos $V_{\mathrm{d}\,loc}^{\ 4}$ e $V_{\mathrm{a}\ loc}^{2\,4} \| H_{\mathrm{D}}$. Então, pelo Corolário 3, $V_{\mathrm{a}\ loc}^{2\,4}$ e H_{D} podem ser usados para implementar um supervisor $S_{\mathrm{a}\ loc}^{2\,4}$, tal que

$$\sup\mathcal{C}(K_{\mathrm{a}\ loc}^{2\,4}, G_{\mathrm{a}\ loc}^{2\,4}) \cap L_{\mathrm{D}} = \sup\mathcal{C}(K_{\mathrm{d}\,loc}^{\ 4}, G_{\mathrm{d}\,loc}^{\ 4})$$

representando, portanto, uma solução localmente ótima para o módulo 4.

Finalmente, pelo Teorema 4, $S_{\mathrm{a}\ loc}^{2\,4}$ e S_{loc}^{i}, $i = 1, 2, 3$ podem ser combinados para resolver o PCS-ML. A Tabela 8 resume a estatística de síntese, considerando-se a combinação de módulos locais e do uso de aproximações.

Tabela 8 – Solução para o PCS-ML através da síntese combinada com aproximações.

$i \in I'$	G_{loc}^{i}	E^{i}	K_{loc}^{i}	V_{loc}^{i}
1	12	2	24	14
2	4	2	8	6
3	6	2	12	9
$i \in I'''$	$G_{\mathrm{a}\ loc}^{2\,i}$	E_{d}^{i}	$K_{\mathrm{a}\ loc}^{2\,i}$	$V_{\mathrm{a}\ loc}^{2\,i}$
4	24	2	48	42
Teste de não-conflito: $(\|_{i\in I'}\ V_{loc}^{i}) \parallel (\Pi(V_{\mathrm{a}\ loc}^{2\,4} \| H_{\mathrm{D}})) =$				276

Como pode ser notado, esse método de síntese ainda preserva a condição de conflito, ao mesmo tempo em que retém apenas as vantagens providas pelo CML e pelo uso de distinguidores.

5.6 SUMÁRIO

Neste capítulo foram apresentados os conceitos preliminares relacionados ao CMC e ao CML, alternativas imediatas para reduzir a carga computacional procedente da síntese monolítica de supervisores para SEDs.

Foi mostrado que a modularização de um problema de controle, em geral, simplifica o procedimento de síntese, embora não altere a forma como as especificações são modeladas. Isso poderia representar um entrave quando se lida com especificações complexas.

Nesse sentido, o CML foi estendido com a incorporação de distinguidores, possibilitando tratar dos aspectos de modelagem. Porém, foi mostrado que essa abordagem é computacionalmente mais custosa, se comparada ao CML original.

Então, a síntese combinada é introduzida como alternativa para equilibrar a relação entre custo computacional e benefícios de modelagem, porém ainda sem ganhos (computacionais) em relação ao procedimento original.

Isso é contornado através da incorporação do uso de aproximações ao CML com distinguidores. A síntese combinada é, então, ajustada para admitir a resolução de problemas locais com aproximações e, ao mesmo tempo, preservar o comportamento global controlado.

6 TCS COM AUTÔMATOS FINITOS ESTENDIDOS

No mesmo sentido que a abordagem com distinguidores, apresentada no Capítulo 4, este capítulo propõe uma segunda alternativa para simplificar a modelagem de especificações e a síntese de supervisores para SEDs. Trata-se do uso de *Autômatos Finitos Estendidos* (AFEs). AFEs têm sido usados na TCS e em outras técnicas computacionais (como verificação formal), basicamente como uma forma de simplificar autômatos (SKÖLDSTAM; ÅKESSON; FABIAN, 2007; OUEDRAOGO et al., 2011). Nesta tese, a abordagem é reestruturada e as seguintes contribuições são derivadas: (i) mostra-se que o uso de AFEs na TCS leva, na verdade, a uma solução para o PCS original; (ii) apresenta-se um novo método de síntese que pode ser conduzido com vantagens computacionais; (iii) define-se um algoritmo para computar supervisores a partir de AFEs. Por fim, um comparativo entre o uso de AFEs e o uso de distinguidores também é apresentado no Capítulo 7.

A primeira parte do capítulo apresenta os AFEs, as principais operações e como a TCS pode ser estendida para incorporar esse novo contexto. O exemplo que vem sendo objeto desta tese é novamente usado para ilustrar a abordagem, bem como para compará-la às demais, apresentadas nos capítulos anteriores. Na segunda parte do capítulo, investiga-se a redução da complexidade computacional da síntese com AFEs, pelo uso de abstrações. Um algoritmo para a computação de supervisores, a partir de AFEs, é proposto e exemplificado. Finalmente, o exemplo do sistema de manufatura volta a ser explorado para ilustrar os ganhos na síntese com abstrações. Por razões de escopo, a associação entre AFEs e o CLM não é promovida nessa tese, embora decorra implicitamente dos resultados aqui apresentados.

6.1 AUTÔMATOS FINITOS ESTENDIDOS

AFEs são estruturas de estados, similares aos AFs convencionais, mas estendidas com *fórmulas* associadas às transições (CHEN; LIN, 2000; SKÖLDSTAM; ÅKESSON; FABIAN, 2007; OUEDRAOGO et al., 2011). Uma fórmula é uma estrutura lógica que atualiza valores de *variáveis* após a transição sobre a qual está implementada, de modo que tais valores passam a compor a semântica de cada estado alcançado[1].

[1] Note, então, que um único estado de um SED pode passar a ser representado por inúmeros estados em um AFE, conforme os valores das variáveis, i.e., um es-

Nesta tese, considera-se que uma variável v é uma entidade associada a um domínio finito, denotado por $\mathrm{dom}(v)$, e um valor inicial, denotado por $v^\circ \in \mathrm{dom}(v)$. Assume-se que $\mathrm{V} = \{v_0, \ldots, v_n\}$ é o conjunto das variáveis com domínio $\mathrm{dom}(\mathrm{V}) = \mathrm{dom}(v_0) \times \cdots \times \mathrm{dom}(v_n)$. Um elemento de $\mathrm{dom}(\mathrm{V})$ é denotado por $\bar{v} = (\bar{v}_0, \ldots, \bar{v}_n)$, com $\bar{v}_i \in \mathrm{dom}(v_i)$.

Em um AFE, quando uma transição é disparada e uma fórmula implementada sobre essa transição atualiza variáveis, ela, na verdade, altera os valores das variáveis que estão associados ao estado corrente (pré-transição), para novos valores que passam a ser associados ao estado alcançado (pós-transição). Para descrever esse efeito, define-se um segundo conjunto de variáveis, chamado *variáveis de próximo-estado*, o qual é denotado por $\mathrm{V}' = \{v'_0, \ldots, v'_n\}$, com $\mathrm{dom}(\mathrm{V}') = \mathrm{dom}(\mathrm{V})$. Assim, sempre que um estado é alcançado, os valores das variáveis em V' são redefinidos conforme a formulação lógica das próximas transições, e assim sucessivamente.

6.1.1 Construção de um AFE

Em muitos casos práticos, conhecer o valor de uma variável em um determinado estado do modelo de um SED pode facilitar a modelagem de uma especificação de controle. Comparativamente, essa ideia pode ser associada ao benefício que se tem ao refinar a ocorrência de um evento da planta, na abordagem com distinguidores. Mas, para que os valores de variáveis componham adequadamente a semântica de um estado, se faz necessário definir quais variáveis devem ser associadas ao modelo do sistema, qual a estrutura de cada variável, e como deve ser implementada a semântica de atualização dessas variáveis. Ainda que essas sejam atividades tipicamente de engenharia, particulares a cada caso, os passos descritos na sequência são propostos para genericamente apoiar essa construção.

(i) *Identificação do conjunto inicial de variáveis:* O primeiro passo para a construção de um AFE é identificar eventos no modelo de um SED, cujas transições, ao atualizarem apropriadamente um conjunto de variáveis, poderiam simplificar a modelagem de

tado de um SED passa a ser uma *locação* (SKÖLDSTAM; ÅKESSON; FABIAN, 2007; OUEDRAOGO et al., 2011) para estados de um AFE. Isso pode ser informalmente associado à relação entre eventos e máscaras, na abordagem com distinguidores. Na sequência, assume-se que a menção a um estado de um AFE incorpora indistintamente essa particularidade semântica.

uma dada especificação. Identificados tais eventos, criam-se as respectivas variáveis que definem V.

(ii) *Definição do domínio das variáveis:* A partir do passo (i), define-se uma estrutura, ou domínio, de cada variável identificada. Cada valor possível dentro desse domínio visa identificar uma circunstância em que o sistema pode alcançar um determinado estado. A semântica de cada valor de variável depende da estrutura de atualizações (passo (iv)).

(iii) *Variáveis complementares:* Dado um conjunto inicial de variáveis e respectivos domínios (passos (i) e (ii)), a implementação de uma semântica de atualizações para tais variáveis pode depender da definição de outras variáveis. Isso ocorre quando o significado a ser associado ao valor de uma variável é dependente da semântica de outra variável, não identificada no passo (i). Quando esse for o caso, os passos (i) e (ii) devem ser revisitados, a fim de caracterizar corretamente o conjunto V.

(iv) *Construção da estrutura lógica de atualizações:* O passo final para a construção de um AFE consiste da implementação das fórmulas lógicas para atualizar os valores das variáveis em V, considerando suas interdependências e o valor semântico que se deseja expressar. Essa atividade é puramente de modelagem, tal que não é conveniente sistematizá-la, tampouco existe uma única maneira de fazê-la, como exemplifica a seção seguinte.

6.1.2 Formulação lógica

Na prática, implementar uma fórmula sobre uma transição pode se dar pela combinação de variáveis, constantes e operadores (lógicos e relacionais). Por exemplo, seja:

- uma variável: x;
- um domínio: $\mathrm{dom}(x) = \{0, \ldots, 5\}$; e
- um valor inicial: $x^{\circ} = 0$.

Uma transição que implementa a fórmula $x' = x + 1$ adiciona, no estado alcançado, o valor 1 ao valor atual da variável x, desde que tal adição esteja dentro do domínio de x, i.e., se o atual valor de x é menor

do que 5. Do contrário, i.e., se $x = 5$, a transição será desabilitada e nenhuma atualização é efetivada.

Considere, agora, o caso da fórmula $x' = \min(x + 1, 5)$. A única diferença dessa, em relação ao exemplo anterior, é que, nesse caso, a transição permanece habilitada quando $x = 5$, situação em que o valor de x, no próximo estado, é atualizado com o valor máximo de seu domínio. Essa alternativa de implementação pode ser útil, em alguns casos, para evitar que uma fórmula restrinja uma transição quando o valor atribuído a ela ultrapasse o seu domínio.

Em outros casos, restringir o disparo de uma transição pode ser exatamente o que se busca. Se assim for, fórmulas como $x = 3$ podem ser usadas para desabilitar uma transição sempre que $x \neq 3$ no estado corrente. Note que, agora, nada é dito sobre o valor de x no próximo estado, e a única função da fórmula é restringir a transição sob tal circunstância. Normalmente, esse tipo de fórmula é implementado em conjunção com outras fórmulas que atualizam os valores de próximo-estado. Diferentemente, a fórmula $x' = 3$ sempre habilita a transição sobre a qual é implementada, e o valor de x, no próximo estado, é forçado a ser 3.

6.1.3 Apresentação formal de um AFE

Formalmente, um AFE é descrito por uma 6-tupla $A_{\mathrm{v}} = \langle \Sigma, \mathrm{V}, Q, Q^{\circ}, Q^{\omega}, \rightarrow \rangle$, tal que:

- Σ é o alfabeto de eventos;
- $\mathrm{V} = \{v_1, \ldots, v_n\}$ é o conjunto de variáveis;
- Q é o conjunto finito de estados;
- $Q^{\circ} \subseteq Q$ é o conjunto de estados iniciais;
- $Q^{\omega} \subseteq Q$ é o subconjunto de estados marcados;
- $\rightarrow \subseteq Q \times \Sigma \times \Pi_{\mathrm{v}} \times Q$ é a relação de transição de estados, na qual Π_{v} é o conjunto de fórmulas sobre $V \cup V'$, tal que cada elemento $p \in \Pi_{\mathrm{v}}$ implementa uma condição Booleana $p(v, v') = \mathit{true}$ ou false.

Nesta tese, denota-se por $x \overset{\sigma:p}{\rightarrow} y$ a existência de uma transição em A_{v}, que parte do estado x para o estado y, com o evento $\sigma \in \Sigma$ e a fórmula $p \in \Pi_{\mathrm{v}}$.

Um AFE A_{v} pode, também, ser interpretado em uma diferente perspectiva, a *interpretação explícita*, a partir da qual ele é visto como um AF $A = \langle \Sigma, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow \rangle$, definido como segue:

- $Q_A = Q \times \mathrm{dom(V)}$;
- $Q_A^\circ = Q^\circ \times \{(v_1^\circ, \ldots, v_n^\circ)\}$;
- $Q_A^\omega = Q^\omega \times \mathrm{dom(V)}$;
- $\rightarrow$ é tal que $(x, \bar{v}) \xrightarrow{\sigma} (y, \bar{v}')$, para $\bar{v}, \bar{v}' \in \mathrm{dom(V)}$, se existe $x \xrightarrow{\sigma:p} y$ com $p(\bar{v}, \bar{v}') = \mathit{true}$.

Nessa interpretação, os valores das variáveis são descritos explicitamente, como parte de cada estado. A relação de transição é definida com base na relação em A_{v}, porém, agora, leva-se em conta o efeito das fórmulas lógicas implementadas sobre as variáveis, i.e., explicita-se a semântica de atualizações. Sendo assim, a interpretação explícita de um AFE é o que de fato evidencia, em termos de número de estados, a real dimensão dessa estrutura.

A relação de transição pode ser estendida para cadeias em Σ^* por:

- $(x, \bar{v}) \xrightarrow{\varepsilon} (x, \bar{v})$, para todo $(x, \bar{v}) \in Q_A$; e
- $(x, \bar{v}) \xrightarrow{s\sigma} (x'', \bar{v}'')$, se $(x, \bar{v}) \xrightarrow{s} (x', \bar{v}') \xrightarrow{\sigma} (x'', \bar{v}'')$, para algum $(x', \bar{v}') \in Q_A$.

Denota-se por $A_{\mathrm{v}} \xrightarrow{s} (x, \bar{v})$, a existência de um estado $(q^\circ, \bar{v}^\circ) \in Q_A^\circ$, tal que $(q^\circ, \bar{v}^\circ) \xrightarrow{s} (x, \bar{v})$. As linguagens, gerada e marcada por A_{v}, são definidas, respectivamente, por:

$$L(A_{\mathrm{v}}) = \{\, s \in \Sigma^* \mid A_{\mathrm{v}} \xrightarrow{s} (x, \bar{v}) \in Q_A \,\} ;$$
$$L^\omega(A_{\mathrm{v}}) = \{\, s \in \Sigma^* \mid A_{\mathrm{v}} \xrightarrow{s} (x, \bar{v}) \in Q_A^\omega \,\} .$$

6.1.4 Propriedades e Operações

Esta seção descreve como algumas das propriedades e operações, comuns para AFs, podem também ser definidas para AFEs. Primeiramente, dois tipos de determinismos são de interesse no contexto dos AFEs.

Definição 14 (Determinismo) *Um AFE* $A_{\mathrm{v}} = \langle \Sigma, \mathrm{V}, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ *é dito ser*

- *Q-determinístico, se* $|Q^\circ| = 1$ *e*

$$x \xrightarrow{\sigma:p_1} y_1 \quad \wedge \quad x \xrightarrow{\sigma:p_2} y_2$$

$$\Rightarrow$$

$$y_1 = y_2,$$

para todo $x, y_1, y_2 \in Q$, $\sigma \in \Sigma$, *e* $p_1, p_2 \in \Pi_{\mathrm{v}}$; *e*

- *V-determinístico, se*

$$(x, \bar{v}) \xrightarrow{\sigma} (y, \bar{w}) \quad \wedge \quad (x, \bar{v}) \xrightarrow{\sigma} (y, \bar{w}')$$

$$\Rightarrow$$

$$\bar{w} = \bar{w}'.$$

Um AFE também pode ser classificado quanto ao tipo de fórmula que ele implementa.

Definição 15 (Totalidade) *Um AFE é dito ser "total" se todas as suas fórmulas são totais. Uma fórmula* $p \in \Pi_{\mathrm{v}}$ *é total se, para todo* $\bar{v} \in \mathrm{dom}(V)$, *existe* $\bar{v}' \in \mathrm{dom}(V)$ *tal que* $p(\bar{v}, \bar{v}') = true$.

Ou seja, um AFE é total se nenhuma de suas transições é desabilitada por uma fórmula. Vale notar, aqui, que implementar fórmulas totais em um AFE remete a um comportamento que pode ser associado ao de um distinguidor, no sentido de que esse tipo de fórmula "distingue" valores de variáveis sem impor nenhuma restrição ao modelo.

A operação de composição de FAs pode ser estendida para AFEs, conforme se segue.

Definição 16 (Composição de AFEs) *Dados dois AFEs,*

$$A_{\mathrm{v}} = \langle \Sigma_A, \mathrm{V}_A, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$$

e

$$B_{\mathrm{v}} = \langle \Sigma_B, \mathrm{V}_B, Q_B, Q_B^\circ, Q_B^\omega, \rightarrow_B \rangle,$$

a composição síncrona de A_{v} *e* B_{v}, *denotada por* $A_{\mathrm{v}} /\!\!/ B_{\mathrm{v}}$, *é tal que:*

$$A_{\mathrm{v}} /\!\!/ B_{\mathrm{v}} = \langle \Sigma_A \cup \Sigma_B,\ \mathrm{V}_A \cup \mathrm{V}_B,\ Q_A \times Q_B,\ Q_A^\circ \times Q_B^\circ,\ Q_A^\omega \times Q_B^\omega,\ \rightarrow \rangle,$$

onde:

- $(x_A, x_B) \xrightarrow{\sigma\ :\ p_A\ \wedge\ p_B} (y_A, y_B)$ *se:*

  - $\sigma \in \Sigma_A \cap \Sigma_B$,
  - $x_A \xrightarrow{\sigma\ :\ p_A}_A y_A$, *e*
  - $x_B \xrightarrow{\sigma\ :\ p_B}_B y_B$;
- $(x_A, x_B) \xrightarrow{\sigma\ :\ p_A} (y_A, x_B)$ *se:*
  - $\sigma \in \Sigma_A \setminus \Sigma_B$ *e*
  - $x_A \xrightarrow{\sigma\ :\ p_A}_A y_A$;
- $(x_A, x_B) \xrightarrow{\sigma\ :\ p_B} (x_A, y_B)$ *se:*
  - $\sigma \in \Sigma_B \setminus \Sigma_A$ *e*
  - $x_B \xrightarrow{\sigma\ :\ p_B}_B y_B$.

A composição de AFEs é similar à de AFs, no sentido de que os eventos, quando compartilhados entre os autômatos compostos, são sincronizados, enquanto a ocorrência dos demais eventos é assíncrona. Adicionalmente, as fórmulas implementadas sobre transições são combinadas por conjunção, no primeiro caso, ou simplesmente preservadas, no último. A inclusão de AFEs é definida na sequência.

Definição 17 (Subautômato) *Um AFE* $A_{\mathrm{v}} = \langle \Sigma_A, \mathrm{V}_A, Q_A, Q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$ *é um subautômato do AFE* $B_{\mathrm{v}} = \langle \Sigma_B, \mathrm{V}_B, Q_B, Q_B^\circ, Q_B^\omega, \rightarrow_B \rangle$, *denotado por* $A_{\mathrm{v}} \sqsubseteq B_{\mathrm{v}}$, *se:*

- $\Sigma_A = \Sigma_B$;
- $\mathrm{V}_A = \mathrm{V}_B$;
- $Q_A \subseteq Q_B$, $Q_A^\circ \subseteq Q_B^\circ$, *e* $Q_A^\omega \subseteq Q_B^\omega$;
- *Se* $x \xrightarrow{\sigma\ :\ p_1}_A y$, *então existe uma transição* $x \xrightarrow{\sigma\ :\ p_2}_B y$ *tal que a fórmula* p_1 *logicamente implica* p_2.

Ou seja, um subautômato $A_{\mathrm{v}} \sqsubseteq B_{\mathrm{v}}$ resulta da remoção de estados ou transições de B_{v}, ou da expansão (fortalecimento) das fórmulas em B_{v}. Por exemplo, o AFE

$$x \xrightarrow{\sigma\ :\ v'=1} y$$

é um subautômato do AFE

$$x \xrightarrow{\sigma\ :\ v'=1} y \xrightarrow{\gamma\ :\ v'=2} z.$$

Outro exemplo pode ser dado pelo AFE

$$x \xrightarrow{\sigma \,:\, v'=1 \,\wedge\, v=0} y$$

que é um subautômato do AFE

$$x \xrightarrow{\sigma \,:\, v'=1} y,$$

pois, enquanto no primeiro caso a transição com σ só pode ocorrer quando $v = 0$, no segundo, a transição é irrestrita. Ademais, ambos atualizam da mesma forma a variável v, no próximo-estado.

Por definição, $A_{\rm v} \sqsubseteq B_{\rm v}$ implica que $L(A_{\rm v}) \subseteq L(B_{\rm v})$ e $L^\omega(A_{\rm v}) \subseteq L^\omega(B_{\rm v})$.

Uma classe particular de subautômatos pode ser obtida ao aplicar a operação de *restrição* sobre um AFE, cuja definição é apresentada como segue.

Definição 18 (Restrição de AFE) *Dado um AFE* $A_{\rm v} = \langle \Sigma, {\rm V}, Q, Q^\circ, Q^\omega, \rightarrow \rangle$, *seja* $X \subseteq Q \times {\rm dom(V)}$. *A restrição de* $A_{\rm v}$ *a* X, *denotada por* $A_{{\rm v}|X}$, *é o AFE*

$$A_{{\rm v}|X} = \langle \Sigma, V, Q_{|X}, Q^\circ_{|X}, Q^\omega_{|X}, \rightarrow_{|X} \rangle,$$

onde:

- $Q_{|X} = \{\, x \in Q \mid (x, \bar{v}) \in X$, *para algum* $\bar{v} \in {\rm dom}(V) \,\}$;
- $Q^\circ_{|X} = \{\, (x^\circ, \bar{v}^\circ) \mid x^\circ \in Q^\circ \,\}$;
- $Q^\omega_{|X} = \{\, x \in Q^\omega \mid (x, \bar{v}) \in X$, *para algum* $\bar{v} \in {\rm dom}(V) \,\}$; *e*
- $x \xrightarrow{\sigma \,:\, p \wedge q_y}_{|X} y$, *se e somente se:*
  - $x \overset{\sigma:p}{\rightarrow} y$, *em* $A_{\rm v}$; *e*
  - q_y *é uma fórmula sobre* V', *tal que* $q_y(\bar{v}') = true$, *se e somente se,* $(y, \bar{v}') \in X$.

Lema 7 *Dado um AFE* $A_{\rm v} = \langle \Sigma, {\rm V}, Q, Q^\circ, Q^\omega, \rightarrow \rangle$, *seja* $X \subseteq Q \times {\rm dom}(V)$. *Então, é verdade que* $A_{{\rm v}|X} \sqsubseteq A_{\rm v}$.

Prova: Decorre diretamente das definições 17 e 18. ∎

Uma vez definidas as propriedades básicas para AFEs, a próxima seção mostra como esses autômatos podem ser usados na síntese de supervisores para SEDs.

6.1.5 Controle Supervisório com AFE

Considere que, ao invés de serem representadas pelos AFs G e E, como no PCS, a planta e as especificações de um SED passam, agora, a ser modeladas pelos respectivos AFEs G_{v} e E_{v}. Para AFEs, a síntese de controle pode ser definida como segue.

Definição 19 (V-Controlabilidade) *Dados dois AFEs,* $E_{\mathrm{v}} = \langle \Sigma, V, Q_E, Q_E^{\circ}, Q_E^{\omega}, \rightarrow_E \rangle$ *e* $G_{\mathrm{v}} = \langle \Sigma, V, Q_G, Q_G^{\circ}, Q_G^{\omega}, \rightarrow_G \rangle$*, diz-se que* E_{v} *é* V-*controlável e.r.a.* G_{v} *se a seguinte implicação é verdadeira para todo* $s \in \Sigma^*$*,* $\mu \in \Sigma_{\mathrm{u}}$*,* $x_E \in Q_E$*,* $x_G, x'_G \in Q_G$ *e* $\bar{v}, \bar{v}' \in \mathrm{dom(V)}$*:*

$$\begin{array}{c} E_{\mathrm{v}} \xrightarrow{s} (x_E, \bar{v}) \\ \wedge \\ G_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{v}) \xrightarrow{\mu} (x'_G, \bar{v}') \\ \Rightarrow \\ \exists\ x'_E \in Q_E \mid E_{\mathrm{v}} \xrightarrow{s} (x_E, \bar{v}) \xrightarrow{\mu} (x'_E, \bar{v}'). \end{array}$$

Ou seja, uma especificação modelada por E_{v} é V-controlável se não proíbe nenhum evento não-controlável, possível na planta G_{v}. A V-controlabilidade de AFEs se diferencia da controlabilidade convencional, definida para linguagens no Capítulo 3, no sentido de que a especificação não apenas habilita todos os eventos não-controláveis, possíveis na planta, mas também atualiza as variáveis da mesma forma como na planta, quando da ocorrência de um evento não-controlável. Já para eventos controláveis, a especificação pode definir o tipo de atualização a ser efetivado.

A partir do conceito de V-controlabilidade, a síntese para AFEs pode ser definida. Sejam G_{v} e E_{v} dois AFEs modelando respectivamente a planta de um SED e suas especificações, tal que $E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$ modela o comportamento esperado sob controle, define-se o conjunto

$$\mathcal{C}_V = \{K_{\mathrm{v}} \sqsubseteq E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}} \mid K_{\mathrm{v}} \text{ é V-controlável } e.r.a.\ G_{\mathrm{v}}\}.$$

$\mathcal{C}_V$ contém um elemento (AFE) supremo, denotado por $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, representando o comportamento menos restritivo possível de ser implementado sobre G_{v}, de modo a garantir o cumprimento da especificação E_{v}.

Note que a V-controlabilidade é uma generalização, para AFEs, da controlabilidade convencional, definida para AFs. Pode ser mostrado (Proposição 13) que o resultado de síntese é o mesmo, para ambos os métodos, em um caso determinístico.

Proposição 13 *Sejam $G_{\rm v}$ e $E_{\rm v}$ dois AFEs Q-determinísticos, tal que $G_{\rm v}$ também é V-determinístico e seja G a versão explícita de $G_{\rm v}$. Então,*

$$\sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G) = L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})) \ . \tag{6.1}$$

Prova:

- $\sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G) \subseteq L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$:

  Considere o conjunto

$$\begin{aligned}\hat{X} = \{\, &(x_E, x_G, \bar{v}) \in Q_E \times Q_G \times \mathrm{dom(V)} \mid \\ &\exists\ s \in \sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G) \text{ tal que } E_{\rm v} /\!\!/ G_{\rm v} \xrightarrow{s} \\ &(x_E, x_G, \bar{v}) \,\}.\end{aligned} \tag{6.2}$$

Mostra-se que $(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}}$ é V-controlável e.r.a. $G_{\rm v}$. Seja $s \in \Sigma^*$ e $\mu \in \Sigma_{\rm u}$, tal que

$$\begin{aligned}(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}} &\xrightarrow{s} (x_E, x_G, \bar{v}) \\ &\text{e} \\ G_{\rm v} &\xrightarrow{s} (y_G, \bar{v}) \xrightarrow{\mu} (y'_G, \bar{v}') \ .\end{aligned}$$

Então, $s\mu \in L(G_{\rm v})$ e $s \in \sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G)$, pela construção de $\hat{X}$. Além disso, pelo Q-determinismo de $G_{\rm v}$, também é verdade que $x_G = y_G$. Como $\sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G)$ é controlável e.r.a. $L(G_{\rm v}) = L(G)$, então é verdade que $s\mu \in \sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G)$. Uma vez que $G_{\rm v}$ e $E_{\rm v}$ são Q-determinísticos e que, além disso, $G_{\rm v}$ é V-determinístico, então se segue que

$$(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}} \xrightarrow{s} (x_E, x_G, \bar{v}) = (x_E, y_G, \bar{v}) \xrightarrow{\mu} (y'_E, y'_G, \bar{v}'),$$

para algum $y'_E \in Q_E$, i.e., $(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}}$ é V-controlável e.r.a. $G_{\rm v}$. Como também é verdade, por definição, que $(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}} \sqsubseteq E_{\rm v} /\!\!/ G_{\rm v}$, então $(E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}} \sqsubseteq \sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})$. Finalmente, pela construção de $\hat{X}$, se segue que

$$\begin{aligned}\sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G) &= L((E_{\rm v} /\!\!/ G_{\rm v})_{|\hat{\rm X}}) \\ &\subseteq L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})) \ .\end{aligned}$$

- $\sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G) \supseteq L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$:

Para mostrar a inclusão inversa, mostra-se, primeiramente, que $L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$ é controlável e.r.a. $L(G) = L(G_{\rm v})$. Seja $s \in L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$ e $s\mu \in L(G_{\rm v})$, para algum $s \in \Sigma^*$ e $\mu \in \Sigma_{\rm u}$. Assim, pelo V-determinismo de $G_{\rm v}$, existem os domínios $\bar{v}, \bar{v}' \in \mathrm{dom}(V)$, tal que

$$\begin{aligned} G_{\rm v} &\xrightarrow{s} (y_G, \bar{v}) \xrightarrow{\mu} (y'_G, \bar{v}') \text{ e} \\ \sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}) &\xrightarrow{s} (x_E, x_G, \bar{v}) \ . \end{aligned}$$

Como $\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})$ é V-controlável e.r.a. $G_{\rm v}$, então é verdade que

$$\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}) \xrightarrow{s} (x_E, x_G, \bar{v}) \xrightarrow{\mu} (x'_E, x'_G, \bar{v}') \ ,$$

para algum estado $x'_E \in Q_E$ e $x'_G \in Q_G$, e portanto, $s\mu \in L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$. Isso é suficiente para mostrar que $L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$ é controlável e.r.a. $L(G_{\rm v})$. Como, além disso,

$$L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})) \subseteq L(E_{\rm v} /\!\!/ G_{\rm v}),$$

então se segue que

$$L(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v})) \subseteq \sup\mathcal{C}(L(E_{\rm v} /\!\!/ G_{\rm v}), G).$$

■

Além da controlabilidade, também é esperado que o comportamento controlado seja não-bloqueante. Para AFEs, a propriedade de não-bloqueio é definida como segue.

Definição 20 (Não-bloqueio de AFEs) *Um AFE* $A_{\rm v} = \langle \Sigma, \mathrm{V}, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ *é não-bloqueante se*

$$\begin{aligned} & A_{\rm v} \xrightarrow{s} (x, \bar{v}) \\ & \Rightarrow \\ & (x, \bar{v}) \xrightarrow{t} (y, \bar{w}), \ \textit{para algum } (y, \bar{w}) \in Q^\omega. \end{aligned}$$

Note que, por definição, se um AFE $A_{\rm v}$ é não-bloqueante, então a sua linguagem, $L(A_{\rm v})$, também é não-bloqueante. Em seguida, o problema de controle supervisório é formulado para AFEs. Nesse problema, assume-se que a planta e as especificações do sistema são

modeladas por AFEs Q-determinísticos. O V-determinismo não é requerido, uma vez que o uso de abstrações (a ser apresentado na Seção 6), pode levar a modelos nos quais essa propriedade não se observa.

Problema 5 - PCS com Variáveis (PCS-V): *Dados os AFEs E_{v} e G_{v}, tal que E_{v} e G_{v} são Q-determinísticos e modelam respectivamente a especificação e a planta de um SED, encontre um subautômato não-bloqueante $K_{\mathrm{v}} \sqsubseteq E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$ que seja* V*-controlável e.r.a. G_{v}.*

Se o AFE $V_{\mathrm{v}} = \sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ é não-bloqueante, então ele é a solução ótima para o PCS-V e pode ser usada para implementar um supervisor que desabilita todo evento controlável, elegível em G_{v}, que não seja elegível em V_{v}. Essa implementação pode se dar através da malha de controle do PCS, i.e., usando-se a versão explícita do AFE V_{v} para implementar um supervisor S que atua sobre a versão explícita da planta G_{v} atualizando, após cada cadeia $s \in L(G_{\mathrm{v}})$, o conjunto $S(s)$ de eventos habilitados. Nesse contexto, o *comportamento gerado em malha fechada* é dado pela linguagem $L(S/G_{\mathrm{v}})$, enquanto o *comportamento marcado em malha fechada* é dado por $L^{\omega}(S/G_{\mathrm{v}}) = L(S/G_{\mathrm{v}}) \cap L_{\mathrm{S}}$, onde $L_{\mathrm{S}} \subseteq L^{\omega}(G_{\mathrm{v}})$. A ação de controle de S se materializa com a desabilitação de eventos possíveis em $L(G_{\mathrm{v}})$, os quais não são possíveis em $L(V_{\mathrm{v}})$, após uma cadeia $s \in L(S/G_{\mathrm{v}})$. A função de marcação corresponde a marcar cadeias que são marcadas em ambos V_{v} e G_{v}. Pelo não-bloqueio de V_{v}, S é não-bloqueante e $L(S/G_{\mathrm{v}}) = \overline{L^{\omega}(S/G_{\mathrm{v}})}$.

Uma solução para o PCS-V pode ser, também, associada à solução original, para o PCS, desde que seja garantida a equivalência na representação dos problemas. Nesse sentido, considere o seguinte pressuposto:

Pressuposto 1 *Para G e E, como no PCS, seja G_{v} a versão explícita de um AFE modelando a planta de um SED, tal que $L(G_{\mathrm{v}}) = L(G)$, e E_{v} um AFE expressando os mesmos requisitos que E, tal que $L(E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}) = L(E\|G)$ e $L^{\omega}(E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}) = L^{\omega}(E\|G)$.*

O Pressuposto 1 assume a equivalência entre os modelos usados para resolver o PCS e o PCS-V. Note que, enquanto AFs e AFEs definem dois diferentes formalismos de modelagem, o problema de controle permanece exatamente o mesmo. Sendo assim, garantir o Pressuposto 1 passa a ser uma tarefa de modelagem, tal como o é a relação entre o PCS e o PCS-D, estabelecida no Capítulo 4. Se o Pressuposto 1 é verdadeiro, então pode-se enunciar o seguinte teorema.

Teorema 5 *Dados K e G, como no PCS, e E_{v} e G_{v}, como no PCS-V, então, sob as condições do Pressuposto 1, é verdade que*

$$L(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) = \sup\mathcal{C}(K, G) \ . \tag{6.3}$$

Prova:

$$\begin{aligned}
\sup\mathcal{C}(K,G) &= \sup\mathcal{C}(L^{\omega}(E\|G), G) \\
&= \sup\mathcal{C}(L^{\omega}(E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}), G) \qquad (\text{pelo Pressuposto 1}) \\
&= \sup\mathcal{C}(L^{\omega}(E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}), G_{\mathrm{v}}) \\
&= L(\sup\mathcal{C}_V(L^{\omega}(E_{\mathrm{v}}, G_{\mathrm{v}}) \qquad (\text{pela Proposição 13}).
\end{aligned}$$

■

Ou seja, se é garantida a equivalência entre plantas e especificações usadas no PCS e no PCS-V, então a resolução do PCS-V leva a uma solução de controle equivalente à do PCS, e vice-versa. O exemplo, apresentado a seguir, ilustra essa abordagem.

6.1.6 Sistema de manufatura modelado com AFEs

Considere o exemplo do sistema de manufatura com retrabalho, apresentado no Capítulo 4 e reproduzido na Figura 25.

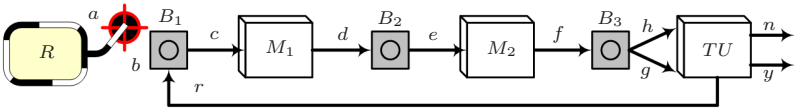


Figura 25 – Pequena fábrica com retrabalho de materiais.

Sobre esse processo foram definidas quatro especificações de controle: E^1, E^2 e E^3, destinadas a evitar o *overflow* e o *underflow* de cada respectivo *buffer*; e E^4, modelada para limitar o número de ciclos de retrabalho. Foi mostrado que a modelagem de E^1, E^2 e E^3 é simples e envolve autômatos com apenas 2 estados cada. Por outro lado, mostrou-se que modelar E^4 é substancialmente mais complexo e essa dificuldade aumenta com a escalabilidade do processo, como, por exemplo, aumentando o número permitido de ciclos.

Nesse sentido, o Capítulo 4 mostra como modelar e resolver esse problema de controle com distinguidores. Nesta seção, é exemplificado

como resolver o problema através de AFEs. Primeiramente, seguem-se os passos, apresentados na Subseção 6.1.1, para estender o modelo do sistema com variáveis, adequadamente identificadas para simplificar o modelo de E^4. Em seguida, mostra-se como tais variáveis podem ser atualizadas e, por fim, a estatística de síntese é apresentada.

6.1.6.1 Construção dos AFEs

Para o exemplo, os passos para a construção dos AFEs são assim caracterizados:

(i) *Identificação do conjunto inicial de variáveis:* Modelar E^4 consiste em desabilitar g ou h após a ocorrência de f. Logo, essa escolha entre g ou h é feita em função de uma peça inserida em B_3, ou seja, após a ocorrência de uma transição etiquetada pelo evento f. Como a decisão de modelagem depende do número de vezes em que a peça em B_3 foi manufaturada, a ocorrência do evento f deve prover informações sobre o *status* da peça. Uma variável, denotada por b_3, é então criada para esse fim, e o passo (i) está concluído.

(ii) *Domínio e semântica de b_3:* Como cada peça pode estar no *buffer* B_3 em três diferentes contextos (0, 1, e 2 retrabalhos), então define-se $\mathrm{dom}(b_3) = \{0, 1, 2\}$, tal que 0 é o valor inicial, e cada um dos valores do domínio visa identificar a respectiva quantidade de retrabalhos. Fazendo-se uso da variável b_3, a modelagem de E^4 torna-se trivial, e pode ser dada pelo AFE mostrado na Figura 26.

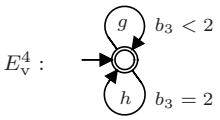


Figura 26 – AFE E_{v}^4 modelando a especificação E^4.

Nesse modelo, uma transição com o evento g (a qual possibilita um retrabalho) só é permitida nos ciclos 0 e 1, enquanto h (que leva à saída definitiva do sistema) só é habilitado no último ciclo. Vale a pena notar que a estrutura de E_{v}^4 pode ser usada para restringir o sistema a qualquer número finito de ciclos de trabalho, simplesmente ajustando as constantes das fórmulas lógicas.

(iii) *Variáveis complementares:* Para identificar quantas vezes uma peça, depositada no *buffer* B_3, foi trabalhada, é necessário rastrear o seu histórico desde que adentrou no ciclo de trabalho. Esse rastreamento pode ser dissociado da enumeração explícita de estados e semanticamente implementado via variáveis, desde que cada subsistema incluso no ciclo de trabalho seja associado a uma variável em particular. Assim, revisitam-se os passos (i) e (ii) a fim de definir o conjunto final de variáveis que, nesse exemplo, é dado por $V = \{b_1, m_1, b_2, m_2, b_3, tu\}$, com domínios $\mathrm{dom}(b_1) = \mathrm{dom}(m_1) = \mathrm{dom}(b_2) = \mathrm{dom}(m_2) = \mathrm{dom}(b_3) = \mathrm{dom}(tu) = \{0, 1, 2\}$ e valores iniciais $b_1^\circ = m_1^\circ = b_2^\circ = m_2^\circ = b_3^\circ = tu^\circ = 0$.

(iv) *Modelagem da semântica das variáveis:* Para o exemplo proposto, a interdependência entre os valores das variáveis em V pode ser representado, no modelo da planta, pelos AFEs mostrados na Figura 27.

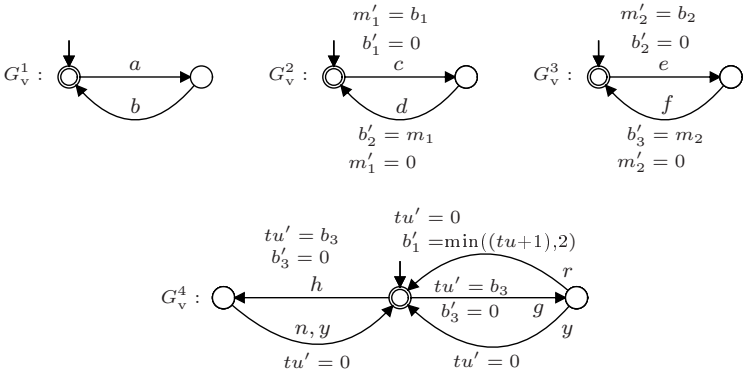


Figura 27 – Modelo da planta construído com variáveis.

Quando uma peça é depositada pelo robô R no *buffer* B_1 (evento b), as variáveis devem ser todas atualizadas com o valor 0 (nenhum retrabalho). Haja visto que todas elas possuem valor inicial 0, então nenhuma fórmula é implementada sobre a transição com b.

Quando M_1 retira a peça de B_1 (evento c), a variável m_1 é atualizada com o valor de b_1 ($m_1' = b_1$) e b_1 é reinicializada ($b_1' = 0$). Em seguida, as transições com os eventos d, e, f, g e h conduzem

o valor atribuído a m_1, individualmente, através das respectivas variáveis b_2, m_2, b_3 e tu, tal que, no primeiro ciclo, todas as variáveis são atualizadas com o valor 0. Após transferirem o valor, cada variável é reinicializada. Em TU, caso ocorra um retrabalho (evento r), a variável b_1 recebe o valor do ciclo anterior (0), trazido por tu, incrementado com 1 ($b'_1 = min((tu + 1), 2)$), identificando assim o novo ciclo que se inicia (1º retrabalho).

Note que, como implementada, a fórmula $b'_1 = min(tu + 1, 2)$ é total e, nesse exemplo, também torna todas as demais fórmulas totais, de modo que o modelo da planta é total. Além disso, para esse caso em particular, o modelo da planta também é Q-determinístico e V-determinístico.

Por razões de clareza, foram removidas da ilustração as fórmulas que atualizam variáveis de próximo-estado com o próprio valor corrente. Na prática, não se exige que esse tipo de fórmula seja explicitamente declarada, uma vez que ferramentas de modelagem, tais como *Supremica* (AKESSON et al., 2013), automaticamente implementam tais atualizações.

6.1.6.2 Síntese de controle com AFEs

A partir dos modelos apresentados acima, uma solução de controle para o exemplo pode ser obtida. Seja a planta do sistema modelada pela composição $G_{\mathrm{v}} = G^1_{\mathrm{v}} /\!\!/ G^2_{\mathrm{v}} /\!\!/ G^3_{\mathrm{v}} /\!\!/ G^4_{\mathrm{v}}$, e a especificação modelada por $E_{\mathrm{v}} = E^1 /\!\!/ E^2 /\!\!/ E^3 /\!\!/ E^4_{\mathrm{v}}$, tal que E^1, E^2 e E^3 são tais como no exemplo do Capítulo 3. Por serem simples, tais especificações não foram modeladas por AFEs, embora possam ser equivalentemente expressas através de fórmulas lógicas.

Os AFEs, usados na síntese do PCS-V, têm número de estados (na interpretação explícita) como mostrado na segunda linha da Tabela 9. A solução para o problema original, PCS, é apresentada na primeira linha, para fins comparativos.

A versão implícita da planta G_{v} é um AFE com 24 estados, que explicitamente pode estender-se para $24 * 3^6 = 17.496$ estados, dos quais, nesse caso em particular, 3.024 são alcançáveis. Note que a especificação global $K_{\mathrm{v}} = E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$ contém o mesmo número de estados do modelo de K, no PCS.

Similarmente, o AFE que representa o supervisor $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ contém os mesmos 212 estados do modelo de $\sup\mathcal{C}(K, G)$. De fato, embora seja útil para fins de modelagem, o uso de AFE não leva di-

Tabela 9 – Número de estados dos autômatos usados na síntese do PCS e do PCS-V.

G	E	K	V
24	1080	**7220**	212
G_{v}	E_{v}	K_{v}	V_{v}
3024	8	**7220**	212

retamente à redução do número de estados dos modelos envolvidos no problema de controle. Isso porque os valores das variáveis precisam ser explicitamente associados aos estados da planta, para a síntese, o que elimina possíveis benefícios computacionais.

Na sequência, mostra-se como explorar AFEs para reduzir o custo de síntese.

6.2 SÍNTESE COM ABSTRAÇÃO DE VARIÁVEIS

Nesta tese, a ideia de abstrair uma variável está associada à ideia de remover de um AFE as fórmulas que atualizam essa variável. Assim, evita-se que seus valores sejam explicitados, ocupando memória computacional ao processar o autômato. Essa técnica vem sendo usada na literatura como forma de simplificar autômatos para *model checking* (BÉRARD et al., 2001), quando a modelagem utiliza-se de AFEs. Na TCS, embora o uso de AFEs não seja inédito enquanto formalismo alternativo de modelagem, a abstração de variáveis é explorada apenas indiretamente, no contexto de outras abordagens de simplificação de síntese.

Na síntese composicional (MOHAJERANI et al., 2011), por exemplo, abstrações de AFs são usadas para facilitar a computação incremental de supervisores modulares. Na síntese com distinguidores, como apresentada no Capítulo 4, o conceito de uma *D-aproximação* remete à ideia de abstração. E, na síntese com AFEs, a abstração por *projeção de transições* (SHOAEI; FENG; LENNARTSON, 2012) e a exploração de técnicas de *computação simbólica* (LEGALL; JEANNET; MARCHAND, 2005), podem ser associadas à ideia da abstração de variáveis.

Salvas as similaridades, não se conhece nenhum resultado que sustente a computação de supervisores ótimos a partir de abstrações de variáveis, lacuna essa que justifica a proposta a ser apresentada nesta seção. Como a síntese com AFEs, particularmente ao envolver abstra-

ções, difere do método convencional e de suas inúmeras implementações e ferramentas, um algoritmo para computar supervisores a partir de AFEs (com ou sem abstrações) também é apresentado. A partir desse algoritmo, mostra-se como determinar se um resultado de síntese é minimamente restritivo. Quando esse não é o caso, i.e., quando a solução é subótima, provê-se um meio para a reconstrução da abstração, otimizada pela incorporação de variáveis automaticamente apontadas pelo algoritmo de síntese.

6.2.1 Construindo uma abstração

Formalmente, a abstração de variáveis pode ser descrita através do conceito de quantificador existencial, definido como segue.

Definição 21 (Abstração Existencial) *Dados um AFE A_{v} e um conjunto de variáveis $W \subseteq V$, a abstração existencial de A_{v} é definida pelo autômato $\exists W A_{\mathrm{v}}$, obtido a partir da substituição de cada transição*

$$x \overset{\sigma:p}{\rightarrow} y, \text{ em } A_{\mathrm{v}}$$

por

$$x \xrightarrow{\sigma:\exists W \exists W' p} y, \quad \text{em } \exists W A_{\mathrm{v}}.$$

A ideia sobre a qual uma *abstração existencial* (ou simplesmente *abstração*) é definida, é bastante simples: a fórmula existencialmente quantificada $\exists W \, \exists W' \, p$ é verdadeira se existe uma combinação de valores de $\bar{w}, \bar{w}' \in \mathrm{dom}(W)$, tal que $p(\bar{w}, \bar{w}') = true$, ou seja, se a partir de um estado, é possível alcançar um próximo estado, com alguma combinação de valores de variáveis. Por exemplo, se $\mathrm{dom}(x) = \{0, \ldots, 9\}$ e $\mathrm{dom}(y) = \{0, 1\}$, então $\exists y \, \exists y' (x' = y)$ é equivalente a $x' = 0 \vee x' = 1$, e $\exists x \, \exists y \, \exists x' \, \exists y' \, (x' = y) = true$.

Note que, por definição, uma abstração $\exists W A_{\mathrm{v}}$ de A_{v} pode ser construída através da remoção das fórmulas implementadas com variáveis em W. Assim, considerar que $W = V$ significa, na verdade, produzir a *máxima* (ou a mais relaxada) abstração de um AFE, correspondendo à remoção de todas as fórmulas do autômato e, consequentemente, à criação de um AF.

Por relaxar as fórmulas implementadas sobre as transições de um AFE, o comportamento modelado por uma abstração tende a ser ampliado em relação ao comportamento original, i.e., um espectro comportamental maior é possível no modelo abstraído. Assim sendo, se

segue, por definição, que $A_{\mathrm{v}} \sqsubseteq \exists W A_{\mathrm{v}}$. Resta saber qual é o efeito da substituição de A_{v} por $\exists W A_{\mathrm{v}}$ na síntese de supervisores, o que é investigado a seguir.

6.2.2 Uma solução para o PCS-V obtida através de abstrações

No contexto do PCS-V, considere abstrair o modelo G_{v} da planta de um SED. A seguir, mostra-se que usar $\exists W G_{\mathrm{v}}$, na síntese, leva a uma solução V-controlável para a planta original.

Teorema 6 *Dados dois AFEs Q-determinísticos, G_{v} e E_{v}, e um conjunto de variáveis $W \subseteq V$, é verdade que*

$$\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \,/\!\!/\, G_{\mathrm{v}} \text{ é V-controlável e.r.a. } G_{\mathrm{v}} \tag{6.4}$$

e

$$L^{\omega}(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \,/\!\!/\, G_{\mathrm{v}}) \subseteq L^{\omega}(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))\ . \tag{6.5}$$

Prova:

- (6.4): $\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \,/\!\!/ G_{\mathrm{v}}$ é V-controlável e.r.a. G_{v}.

  Considere o conjunto

  $$\hat{X} = \{\, (x_E, x_G, \bar{v}) \in Q_E \times Q_G \times \mathrm{dom}(V) \mid \mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \,/\!\!/\, G_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, y_G, \bar{v}), \text{ para algum } s \in \Sigma^* \text{ e } y_G \in Q_G \,\}.$$

  Seja $\hat{K} = (E_{\mathrm{v}} \,/\!\!/\, G_{\mathrm{v}})_{|\hat{\mathrm{X}}}$, $t \in \Sigma^*$ e $\mu \in \Sigma_{\mathrm{u}}$, tal que

  $$G_{\mathrm{v}} \xrightarrow{t} (x_G, \bar{v}) \xrightarrow{\mu} (x'_G, \bar{v}') \tag{6.6}$$

  e

  $$\hat{K} \xrightarrow{t} (y_E, y_G, \bar{v})\ .$$

  Pelo Q-determinismo de G_{v}, pode-se afirmar que $x_G = y_G$. Além disso, pela construção de $\hat{K}$, também é verdade que $(y_E, y_G, \bar{v}) \in \hat{X}$. Assim,

  $$\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \,/\!\!/\, G_{\mathrm{v}} \xrightarrow{t} (y_E, x_G, x_G, \bar{v})$$

  e, portanto,

  $$\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) \xrightarrow{t} (y_E, x_G, \bar{v})\ . \tag{6.7}$$

Como, por (6.6) e pela Definição 21,

$$\exists WG_{\mathrm{v}} \xrightarrow{t} (x_G, \bar{v}) \xrightarrow{\mu} (x'_G, \bar{v}') \ ,$$

então é verdade, por (6.7) e pela V-controlabilidade de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$ e.r.a. $\exists WG_{\mathrm{v}}$, que

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) \xrightarrow{t} (y_E, x_G, \bar{v}) \xrightarrow{\mu} (y'_E, x'_G, \bar{v}') \ ,$$

para algum $y'_E \in Q_E$. Sendo assim, também

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}} \xrightarrow{t} (y_E, x_G, x_G, \bar{v}) \xrightarrow{\mu} (y'_E, x'_G, x'_G, \bar{v}') \ , \quad (6.8)$$

e, portanto, $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}$ é V-controlável e.r.a. G_{v}.

- (6.5): $L^\omega(\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}) \subseteq L^\omega(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$.

Pelas definições de $\hat{X}$ e de $\hat{K}$, note que (6.8) implica

$$\hat{K} \xrightarrow{t} (y_E, x_G, \bar{v}) \xrightarrow{\mu} (y'_E, x'_G, \bar{v}') \ .$$

Logo, $\hat{K}$ também é V-controlável e.r.a. G_{v}. Além disso, como por definição $\hat{K} \sqsubseteq E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$, então $\hat{K} \sqsubseteq \sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$. Assim, seja $s \in L^\omega(\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}})$. Então

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, y_G, \bar{v}) \quad (6.9)$$

e

$$\hat{K} \xrightarrow{s} (x_E, x_G, \bar{v}) \ .$$

Como $\hat{K} \sqsubseteq \sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, então é verdade que $s \in L^\omega(\hat{K}) \subseteq L^\omega(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$. ∎

O Teorema 6 mostra que $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$, quando associado à planta original, G_{v}, compõe a uma solução controlável para essa planta. Ainda assim, esse supervisor é obtido a partir de uma abstração $\exists WG_{\mathrm{v}}$, que não atualiza variáveis em W. A computação de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$ pode ser concretizada através do algoritmo de síntese, a ser apresentado na Subseção 6.2.3. Se além de V-controlável, $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$ também for não-bloqueante, então o seguinte corolário pode ser enunciado.

Corolário 5 *Dados dois AFEs Q-determinísticos, G_{v} e E_{v}, e um conjunto de variáveis $W \subseteq V$, se $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}$ é não-bloqueante,*

então esse autômato é uma solução para o PCS-V.

Prova: Decorre diretamente da V-controlabilidade, mostrada no Teorema 6, e do não-bloqueio da composição $\sup\mathcal{C}_V(E_{\rm v}, \exists WG_{\rm v}) /\!\!/ G_{\rm v}$. ■

O teste de não-bloqueio de $\sup\mathcal{C}_V(E_{\rm v}, \exists WG_{\rm v}) /\!\!/ G_{\rm v}$ pode ser simplificado pelo uso das mesmas técnicas mencionadas no Capítulo 4 (PENA; CURY; LAFORTUNE, 2009; FLORDAL; MALIK, 2009).

Note que, embora a combinação do Teorema 6 e do Corolário 5 aponte para uma solução para o PCS-V, essa solução pode não ser minimamente restritiva, uma vez que $L^\omega(\sup\mathcal{C}_V(E_{\rm v}, \exists WG_{\rm v}) /\!\!/ G_{\rm v}) \subseteq L^\omega(\sup\mathcal{C}_V(E_{\rm v}, G_{\rm v}))$.

Os resultados a serem apresentados a seguir, mostram condições suficientes para determinar a otimidade de $\sup\mathcal{C}_V(E_{\rm v}, \exists WG_{\rm v}) /\!\!/ G_{\rm v}$. Esses resultados são apresentados no contexto de um algoritmo que estrutura a síntese de supervisores com AFEs. Nos casos em que a otimidade não é observada, mostra-se como o algoritmo pode também apoiar a construção de abstrações otimizadas, que sempre garantem o resultado minimamente restritivo.

6.2.3 Síntese de supervisores com AFEs

Nesta seção, apresenta-se um algoritmo para computar supervisores a partir de AFEs. Ainda que o algoritmo seja primariamente implementado para a síntese com abstrações, o uso da planta original (sem abstração) é um caso particular, com $W = \emptyset$, que é naturalmente válido.

Primeiramente, é necessário notar que, embora as fórmulas com variáveis em W sejam removidas da abstração da planta $G_{\rm v}$, elas ainda podem fazer parte da especificação $E_{\rm v}$, a qual não é abstraída. Sendo assim, o AFE $E_{\rm v} /\!\!/ \exists WG_{\rm v}$ é definido, na verdade, sobre o conjunto V de variáveis. Esse aspecto precisa ser levado em conta para mostrar que a síntese com abstrações leva, de fato, a vantagens computacionais.

Nesse sentido, o algoritmo proposto é estruturado de maneira a permitir computar um supervisor sem fazer o uso de variáveis em W. Essa ideia é formalizada através do conceito de *restrição de variáveis*, definido como segue.

Definição 22 (Restrição de variáveis) *A restrição de variáveis a*

$U \subseteq V$, é definida para

$$A_{\mathrm{v}} = \langle \Sigma, \mathrm{V}, Q, Q^{\circ}, Q^{\omega}, \rightarrow \rangle$$

pelo AFE

$$A_{\mathrm{v}|\mathrm{U}} = \langle \Sigma, U, Q, Q^{\circ}, Q^{\omega}, \rightarrow_{|\mathrm{U}} \rangle,$$

onde $\rightarrow_{|\mathrm{U}} \subseteq Q \times \Sigma \times \Pi_{\mathrm{u}} \times Q$, sendo $\Pi_{\mathrm{u}} \subseteq \Pi_{\mathrm{v}}$ o conjunto de fórmulas definidas sobre $U \cup U'$, e $\rightarrow_{|\mathrm{U}}$ é tal que, se $x \stackrel{\sigma:p}{\rightarrow} y$, então $x \stackrel{\sigma:p}{\rightarrow}_{|\mathrm{U}} y$.

Ou seja, uma restrição de variáveis a U leva a um AFE cujas fórmulas são implementadas somente com variáveis em U. Por exemplo, se $V = U \cup W$, então $(\exists W A_{\mathrm{v}})_{|\mathrm{U}}$ é um AFE definido sobre variáveis em U, ao passo que $\exists W A_{\mathrm{v}}$ é definido sobre as variáveis em V.

Note que, por definição, a relação de transição de um AFE se dá em função do conjunto de fórmulas implementadas sobre o conjunto de variáveis de estado e de próximo-estado. Logo, restringir o conjunto de variáveis a $U \subseteq V$ diretamente implica que o conjunto de fórmulas é dado por $\Pi_u \subseteq \Pi_{\mathrm{v}}$ e é implementado sobre $U \cup U'$.

Usando o conceito de restrição de variáveis, pode-se computar

$$\sup\mathcal{C}_V(E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}}) \tag{6.10}$$

sem utilizar variáveis em W, através do algoritmo apresentado a seguir.

Algoritmo 1 - Computação de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$

Entrada: E_{v} e $\exists W G_{\mathrm{v}}$
Saída: $\hat{K}_{\mathrm{V}} = \sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$

Passo 1 - Restrição de variáveis a U

Seja $U = V \setminus W$. Construa o seguinte AFE:

- $(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$.

Os estados de $(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$ tem a seguinte estrutura:

- $(x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U)$;

Passo 2 - Estados Fortemente Controláveis em V

Encontre o seguinte conjunto de estados EF$\mathcal{C}$s:

$$\begin{aligned}\mathbf{EFC_V} = \{\, &(x_E, x_G, \bar{u}, \bar{w}) \in Q_E \times Q_G \times \mathrm{dom}(V) \mid \\ &\text{para todo } \bar{u}' \in \mathrm{dom}(U),\ \bar{w}' \in \mathrm{dom}(W) \text{ e } \mu \in \Sigma_{\mathrm{u}}, \\ &\text{tal que} \\ &\qquad (x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}') \text{ em } (\exists WG_{\mathrm{v}})_{|\mathrm{U}}, \\ &\text{existe } x'_E \in Q_E, \text{ tal que} \\ &\qquad (x_E, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_E, \bar{u}', \bar{w}') \text{ em } E_{\mathrm{v}} \\ &\};\end{aligned} \tag{6.11}$$

Passo 3 - Estados Fortemente Controláveis em U

Refine o conjunto de estados EF$\mathcal{C}$s, tal que:

$$\begin{aligned}\mathbf{EFC}_{\mathrm{U}} = \{\, &(x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid \\ &\text{para todo } \bar{w} \in \mathrm{dom}(W), \text{ é verdade que } (x_E, x_G, \bar{u}, \bar{w}) \in \\ &\mathbf{EFC}_{\mathrm{V}} \ \};\end{aligned} \tag{6.12}$$

Passo 4 - Estados Levemente Controláveis em U

Encontre o conjunto de *Estados Levemente Controláveis* (EL$\mathcal{C}$s), como segue:

$$\begin{aligned}\mathbf{ELC}_{\mathrm{U}} = \{\, &(x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid \\ &\text{para todo } \mu \in \Sigma_{\mathrm{u}}, \text{ se} \\ &\qquad (x_G, \bar{u}) \xrightarrow{\mu} \text{ em } (\exists WG_{\mathrm{v}})_{|\mathrm{U}}, \\ &\text{então, também,} \\ &\qquad (x_E, \bar{u}) \xrightarrow{\mu} \text{ em } (\exists WE_{\mathrm{v}})_{|\mathrm{U}} \\ &\};\end{aligned} \tag{6.13}$$

Passo 5 - Conjunto supremo de Estados Fortemente Controláveis

Um conjunto de estados $X \subseteq Q_E \times Q_G \times \mathrm{dom}(U)$ é fortemente controlável e.r.a. $\exists WG_{\mathrm{v}}$ se, para todo $(x_E, x_G, \bar{u}) \in X$ e toda a transição não-controlável

$$(x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}') \text{ em } (\exists WG_{\mathrm{v}})_{|\mathrm{U}},$$

existe uma transição

$$(x_E, \bar{u}) \xrightarrow{\mu} (x'_E, \bar{u}') \text{ em } (\exists WE_{\mathrm{v}})_{|\mathrm{U}},$$

tal que $(x'_E, x'_G, \bar{u}') \in X \cap \mathbf{EFC}_U$.

Como a união de conjuntos fortemente controláveis permanece fortemente controlável, torna-se possível computar o conjunto supremo

$$\widehat{\mathbf{EFC}_{\mathrm{U}}} = \bigcup \{\, X \subseteq Q_E \times Q_G \times \mathrm{dom}(U) \mid X \text{ é um conjunto fortemente controlável e.r.a. } \exists WG_{\mathrm{v}} \,\};$$

Passo 6 - Restrição a $\widehat{\mathbf{EFC}_{\mathrm{U}}}$

Construa a seguinte restrição:

$$\hat{K}_{\mathrm{U}} = (\exists WE_{\mathrm{v}} \mathbin{/\!\!/} \exists WG_{\mathrm{v}})_{|\mathrm{U}\,|\widehat{\mathbf{EFC}_{\mathrm{U}}}} ; \tag{6.14}$$

Passo 7 - Autômato resultante

Construa o AFE resultante

$$\hat{K}_{\mathrm{V}} = \langle \Sigma, V, Q_E \times Q_G, Q^\circ, Q^\omega_E \times Q^\omega_G, \rightarrow \rangle, \tag{6.15}$$

onde:

$$Q^\circ = \{\, (q^\circ_E, q^\circ_G) \in Q^\circ_E \times Q^\circ_G \mid (q^\circ_E, q^\circ_G, \bar{u}^\circ) \in \widehat{\mathbf{EFC}_{\mathrm{U}}} \text{ e } (q^\circ_E, q^\circ_G, \bar{u}^\circ, \bar{w}^\circ) \in \mathbf{EFC_V} \,\},$$

e

$$(x_E, x_G) \xrightarrow{\sigma \,:\, \hat{p} \,\wedge\, p_E \,\wedge\, p_{\mathbf{EFC}\langle x'_E, x'_G\rangle}} (x'_E, x'_G) \tag{6.16}$$

se

$$(x_E, x_G) \xrightarrow{\sigma \,:\, \hat{p}} (x'_E, x'_G) \text{ em } \hat{K}_{\mathrm{U}};$$

$$x_E \xrightarrow{\sigma\,:\,p_E} x'_E \text{ em } E_{\mathrm{v}};\ \text{e}$$

$$p_{\mathbf{EF}\mathcal{C}}\langle x'_E, x'_G\rangle$$

é uma fórmula sobre V', tal que

$$p_{\mathbf{EF}\mathcal{C}}\langle x'_E, x'_G\rangle(\bar{v}') = true$$

se

$$(x'_E, x'_G, \bar{v}') \in \mathbf{EF}\mathcal{C}_{\mathrm{V}}.$$

Informalmente, a ideia por trás desse algoritmo pode ser descrita como segue. Quando uma variável é abstraída de um AFE e as fórmulas que a atualizam são removidas, essa variável pode assumir qualquer valor dentro de seu domínio. Em termos de síntese, isso significa que, a partir de qualquer estado da planta $\exists W G_{\mathrm{v}}$, quando um evento não-controlável é possível, e a especificação E_{v} proíbe alcançar um próximo estado para algum valor de variável em W, então o estado fonte deve ser removido, por ser *inseguro*. Essa classe de estados pode ser identificada *a priori*, através do conceito de *Estado Fortemente Controlável* (EF$\mathcal{C}$).

Note, porém, que um estado pode ser EF$\mathcal{C}$ para alguns valores de variáveis $\bar{w} \in \mathrm{dom}(W)$, e não para outros. Quando este é o caso, ou seja, quando um estado $(x, \bar{w})$ não for EF$\mathcal{C}$, as seguintes implicações precisam ser consideradas na síntese:

- O estado $(x, \bar{w})$ pode ser *refinado*, de modo a permanecer na síntese somente com a combinação de valores de variáveis que o torna EF$\mathcal{C}$.
- Todos os estados que alcançam $(x, \bar{w})$ com transições não-controláveis são inseguros, pois podem alcançá-lo atualizando uma variável em W com algum dos valores inseguros.
- Transições controláveis alcançando $(x, \bar{w})$ podem ser preservadas, desde que os valores inseguros no próximo estado sejam evitados pela síntese.

Em relação à eficiência computacional do algoritmo proposto, vale a pena notar que o conjunto $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$ é computado, no Passo 5, sobre o espaço de estados de $(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$, ou seja, sem utilizar variáveis em W. Já os estados em $\mathbf{EF}\mathcal{C}_{\mathrm{V}}$ e $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$, embora estejam associados a valores de variáveis em W, podem ser computados *a priori*, sem complexificar a síntese. Alternativamente, os estados, nesses dois

conjuntos, podem ser representados simbolicamente ou avaliados de maneira individual ao computar $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$. Portanto, o conjunto de estados $\mathbf{EF}\mathcal{C}_{\mathrm{V}}$, potencialmente extenso, nunca é, de fato, explorado em síntese.

Dessas observações, pode-se notar que as fórmulas implementadas pela especificação E_{v}, as quais necessitam estar associadas a variáveis em W, não afetam a computação de $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$. Embora as restrições implementadas por essas fórmulas sejam consideradas em síntese, o espaço de estados decorrente dos valores de variáveis em W nunca é computado. Para a composição do resultado final, as fórmulas de E_{v} são simplesmente copiadas para o AFE $\hat{K}_{\mathrm{V}}$, no Passo 7 do algoritmo. O Teorema 7 mostra que $\hat{K}_{\mathrm{V}}$ corresponde, na verdade, a $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$ (6.10).

Teorema 7 *Dados dois AFEs Q-determinísticos, E_{v} e G_{v}, o AFE $\hat{K}_{\mathrm{V}}$, computado pelo Algoritmo 1, é tal que*

$$\hat{K}_{\mathrm{V}} = \sup\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}).$$

Prova: O Teorema 7 é demonstrado através dos passos (i) e (ii), apresentados a seguir.

(i) $\hat{K}_{\mathrm{V}} \sqsubseteq E_{\mathrm{v}} /\!\!/ \exists WG_{\mathrm{v}}$ (a) e $\hat{K}_{\mathrm{V}}$ é V-controlável e.r.a. $\exists WG_{\mathrm{v}}$ (b):

- Note que $\hat{K}_{\mathrm{V}} \sqsubseteq E_{\mathrm{v}} /\!\!/ \exists WG_{\mathrm{v}}$ é verdadeiro por construção, pois qualquer transição em $\hat{K}_{\mathrm{V}}$ é uma transição em E_{v} (6.16) e, também, em $\exists WG_{\mathrm{v}}$ (6.14).
- Agora, mostra-se que $\hat{K}_{\mathrm{V}}$ é V-controlável e.r.a. $\exists WG_{\mathrm{v}}$. Seja $\mu \in \Sigma_{\mathrm{u}}$ e $\bar{v}, \bar{v}' \in \mathrm{dom}(V)$, tal que

  $$\exists WG_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{v}) \xrightarrow{\mu} (x'_G, \bar{v}')$$

  e

  $$\hat{K}_{\mathrm{V}} \xrightarrow{s} (y_E, y_G, \bar{v}).$$

  Pelo Q-determinismo de G_{v}, tem-se que $x_G = y_G$. Agora, seja $\bar{v} = (\bar{u}, \bar{w}) \in \mathrm{dom}(U) \times \mathrm{dom}(W)$ e, equivalentemente, $\bar{v}' = (\bar{u}', \bar{w}')$. Como $\hat{K}_{\mathrm{V}} \xrightarrow{s} (y_E, y_G, \bar{v})$, então, pela construção (6.16), é verdade que

  $$(y_E, y_G, \bar{u}) \in \widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$$

  e

  $$(y_E, y_G, \bar{v}) \in \mathbf{EF}\mathcal{C}_{\mathrm{V}}.$$

Como, também,

$$(y_G, \bar{u}) = (x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}'), \text{ em } (\exists W G_{\mathrm{v}})_{|U},$$

então, pela definição de $\mathbf{EFC}_{\mathbf{V}}$ (6.11), é verdade que

$$(y_E, \bar{v}) \xrightarrow{\mu} (y'_E, \bar{v}') \text{ em } E_{\mathrm{v}},$$

para algum $y'_E \in Q_E$.

Mas, como $\widehat{\mathbf{EFC}_{\mathrm{U}}}$ é fortemente controlável e.r.a. $\exists W G_{\mathrm{v}}$, então existe $y''_E \in Q_E$, tal que

$$(y_E, \bar{u}) \xrightarrow{\mu} (y''_E, \bar{u}'), em (\exists W E_{\mathrm{v}})_{|\mathrm{U}}$$

e

$$(y''_E, x'_G, \bar{u}') \in \widehat{\mathbf{EFC}}_U \cap \mathbf{EFC}_U,$$

onde $y''_E = y'_E$, pelo Q-determinismo de E_{v}. Assim, como $(y'_E, x'_G, \bar{u}') \in \mathbf{EFC}_{\mathrm{U}}$, é verdade que $(y'_E, x'_G, \bar{v}') \in \mathbf{EFC}_{\mathrm{V}}$, por (6.12), e tem-se que

$$(y_E, y_G, \bar{u}) \xrightarrow{\mu}_{|\widehat{\mathbf{EFC}_{\mathrm{U}}}} (y'_E, x'_G, \bar{u}'),$$

$$(y_E, \bar{v}) \xrightarrow{\mu} (y'_E, \bar{v}')$$

e

$$(y'_E, x'_G, \bar{v}') \in \mathbf{EFC}_{\mathrm{V}},$$

o que mostra a existência de uma transição (6.16) e, portanto,

$$\hat{K}_{\mathrm{V}} \xrightarrow{s} (y_E, y_G, \bar{v}) \xrightarrow{\mu} (y'_E, x'_G, \bar{v}').$$

(ii) Qualquer subautômato V-controlável de $E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}}$ é um subautômato de $\hat{K}_{\mathrm{V}}$:

Seja A_{v} um subautômato de $E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}}$, tal que $A_{\mathrm{v}} \sqsubseteq E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}}$ é V-controlável e.r.a. $\exists W G_{\mathrm{v}}$. Pela construção (6.16), qualquer fórmula lógica de A_{v} é satisfeita por E_{v}. Isso é suficiente para mostrar que, se um estado é alcançável em A_{v}, então ele pertence a $\mathbf{EFC}_{\mathrm{V}}$ (por (a)) e a $\widehat{\mathbf{EFC}_{\mathrm{U}}}$ (por (b)).

(a) Seja $A_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w})$, $\bar{u}' \in \mathrm{dom}(U)$, $\bar{w}' \in \mathrm{dom}(W)$, e

$\mu \in \Sigma_{\mathrm{u}}$, tal que

$$(x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}'), \text{ em } (\exists WG_{\mathrm{v}})_{|\mathrm{U}}.$$

Claramente,

$$\exists WG_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_G, \bar{u}', \bar{w}').$$

Como A_{v} é V-controlável e.r.a. $\exists WG_{\mathrm{v}}$, então existe $y'_E \in Q_E$ e $y'_G \in Q_G$, tal que

$$A_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (y'_E, y'_G, \bar{u}', \bar{w}').$$

Como, também, $A_{\mathrm{v}} \sqsubseteq E_{\mathrm{v}} /\!\!/ \exists WG_{\mathrm{v}}$, então

$$(x_E, \bar{u}, \bar{w}) \xrightarrow{\mu} (y'_E, \bar{u}', \bar{w}'), \text{ em } E_{\mathrm{v}}.$$

Isso é suficiente para mostrar que $(x_E, x_G, \bar{u}, \bar{w}) \in \mathbf{EFC}_{\mathbf{V}}$, a partir de (6.11).

(b) Agora, é mostrado que o conjunto de estados

$$X = \{\, (x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid A_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}), \text{para algum } \bar{w} \in \mathrm{dom}(W) \,\} \tag{6.17}$$

é fortemente controlável e.r.a. $\exists WG_{\mathrm{v}}$.

Assuma que $(x_E, x_G, \bar{u}) \in X$ e

$$(x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}'), \text{ em } (\exists WG_{\mathrm{v}})_{|\mathrm{U}}.$$

Então, existe um $\bar{w} \in \mathrm{dom}(W)$, tal que

$$A_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}).$$

Ainda, note que

$$E_{\mathrm{v}} \xrightarrow{s} (x_E, \bar{u}, \bar{w}),$$

e que

$$(x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_G, \bar{u}', \bar{w}'), \text{ em } \exists WG_{\mathrm{v}},$$

para algum $\bar{w}' \in \mathrm{dom}(W)$.

Assim, pela V-controlabilidade de A_{v} e.r.a. $\exists WG_{\mathrm{v}}$, é verdade

que

$$(x_E, x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_E, x'_G, \bar{u}', \bar{w}'), \text{ em } A_{\mathrm{v}}.$$

Logo, $(x'_E, x'_G, \bar{u}') \in X$. Como, além disso, $(x'_E, x'_G, \bar{u}', \bar{w}') \in \mathbf{EF}\mathcal{C}_{\mathbf{V}}$, por (a) e, assim, $(x'_E, x'_G, \bar{u}') \in \mathbf{EF}\mathcal{C}_{\mathbf{U}}$, por (6.12), então é verdade que X é um conjunto fortemente controlável de estados, e.r.a. $\exists W G_{\mathrm{v}}$.

■

O Teorema 7 mostra que o algoritmo proposto computa o supervisor $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$. Assim, em combinação com o Teorema 6, tem-se que esse supervisor pode ser usado para compor uma solução controlável $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}$, para o PCS-V. Mas, como (ainda pelo Teorema 6) $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}} \sqsubseteq \sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, não se pode afirmar se tal solução é a minimamente restritiva. Esse aspecto é investigado na sequência.

6.2.4 Síntese de supervisores ótimos com AFEs

Nesta seção, apresentam-se condições suficientes para atestar a máxima permissividade de soluções de controle obtidas via AFEs. Isso é feito explorando-se o algoritmo de síntese proposto, em particular, o conceito de *Estados Levemente Controláveis*, $\mathbf{EL}\mathcal{C}_U$.

Por definição, um estado pertence a $\mathbf{EL}\mathcal{C}_U$ se a partir dele, toda vez que uma transição não-controlável é possível em $(\exists W G_{\mathrm{v}})_{|U}$, essa transição também é possível em $(\exists W E_{\mathrm{v}})_{|U}$. Estados em $\mathbf{EL}\mathcal{C}_U$ se diferenciam de estados em $\mathbf{EF}\mathcal{C}_U$, no sentido de que, no primeiro, a especificação não precisa habilitar transições não-controláveis para todos os valores de variáveis de próximo-estado, mas para ao menos uma combinação de valores. Então, estados em $\mathbf{EF}\mathcal{C}_U$ são sempre estados em $\mathbf{EL}\mathcal{C}_U$, mas o contrário pode não ser verdadeiro. Um caso particular importante é quando estes dois conjuntos de estados são iguais, o que é formalizado a seguir.

Hipótese 1 *Sejam dois AFEs, E_{v} e G_{v}, tal que G_{v} é total, seja um conjunto de variáveis $W \subseteq V$, e sejam os conjuntos de estados $\mathbf{EF}\mathcal{C}_{\mathbf{U}}$ e $\mathbf{EL}\mathcal{C}_{\mathbf{U}}$, computados como no algoritmo 1. Então, para qualquer $s \in \Sigma^*$,*

$$(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}} \xrightarrow{s} (x_E, x_G, \bar{u}),$$

implica que

$$(x_E, x_G, \bar{u}) \in \mathbf{EF}\mathcal{C}_{\mathbf{U}},$$

$$\vee$$

$$(x_E, x_G, \bar{u}) \notin \mathbf{EL}\mathcal{C}_{\mathbf{U}}.$$

Pela Hipótese 1, se um estado é alcançável na abstração $(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$, então esse estado ou é fortemente controlável, ou não é levemente controlável. Quando tal hipótese é verdadeira, pode-se afirmar que $\mathbf{EF}\mathcal{C}_{\mathbf{U}} = \mathbf{EL}\mathcal{C}_{\mathbf{U}}$. Informalmente, isso significa que cada estado de uma abstração ou é sempre um *bom estado* (nunca removido pela síntese), ou é sempre um *mau estado* (sempre removido em síntese). Sob os termos dessa hipótese, pode ser mostrado que uma solução computada a partir da abstração de um AFE é sempre a minimamente restritiva, como estabelece o teorema a seguir.

Teorema 8 *Sob os termos da Hipótese 1, é verdade que*

$$L^{\omega}(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) \subseteq L^{\omega}(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}).$$

Prova: Considere o conjunto

$$X = \{\, (x_E, x_G, \bar{u}) \in Q_E \times Q_G \times \mathrm{dom}(U) \mid \mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}), \text{ para algum } \bar{w} \in \mathrm{dom}(W) \,\}. \tag{6.18}$$

Inicialmente, mostra-se que

$$X \subseteq \mathbf{EL}\mathcal{C}_U. \tag{6.19}$$

Seja $(x_E, x_G, \bar{u}) \in X$ e assuma $\mu \in \Sigma_{\mathrm{u}}$, tal que

$$(x_G, \bar{u}) \xrightarrow{\mu} \text{ em } (\exists W G_{\mathrm{v}})_{|U}.$$

De $(x_E, x_G, \bar{u}) \in X$ tem-se, pela construção de (6.18), que

$$\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}), \text{ para algum } \bar{w} \in \mathrm{dom}(W),$$

Isso implica que

$$G_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{u}, \bar{w}).$$

Como $(x_G, \bar{u}) \xrightarrow{\mu}$ em $(\exists W G_{\mathrm{v}})_{|U}$ e como G_{v} é total, então existe

$\bar{u}'$ e $\bar{w}'$ tal que

$$(x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_G, \bar{u}', \bar{w}') \text{ em } G_{\mathrm{v}}.$$

Assim, pela V-controlabilidade de $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ e.r.a. G_{v}, é verdade que

$$(x_E, \bar{u}, \bar{w}) \xrightarrow{\mu} \text{ em } E_{\mathrm{v}}.$$

Finalmente, pelas definições 21 e 22 (de $\exists W$ e de $|\mathrm{U}$),

$$(x_E, \bar{u}) \xrightarrow{\mu} \text{ em } (\exists W E_{\mathrm{v}})_{|\mathrm{U}}.$$

Isso é suficiente para mostrar que $(x_E, x_G, \bar{u}) \in \mathbf{EL}\mathcal{C}_{\mathbf{U}}$ e, portanto, $X \subseteq \mathbf{EL}\mathcal{C}_U$.

Mas, se (6.19) é verdadeiro, então também se pode afirmar que

$$X \subseteq \mathbf{EF}\mathcal{C}_U, \tag{6.20}$$

pois, como $(x_E, x_G, \bar{u}) \in X$, então a construção de (6.18) garante que

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}), \text{ para algum } \bar{w} \in \mathrm{dom}(W).$$

Logo,

$$E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}} \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w})$$

e, pelas definições 21 e 22,

$$(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}} \xrightarrow{s} (x_E, x_G, \bar{u}).$$

Portanto, de (6.19) e da Hipótese 1, é verdade que $(x_E, x_G, \bar{u}) \in \mathbf{EF}\mathcal{C}_{\mathbf{U}}$ e, assim, $X \subseteq \mathbf{EF}\mathcal{C}_U$.

O próximo passo da prova consiste em mostrar que $X \subseteq \widehat{\mathbf{EF}\mathcal{C}}_U$, i.e., mostra-se que X é um conjunto de estados fortemente controláveis e.r.a. $\exists W G_{\mathrm{v}}$.

Seja $(x_E, x_G, \bar{u}) \in X$ e seja $\mu \in \Sigma_{\mathrm{u}}$, tal que

$$(x_G, \bar{u}) \xrightarrow{\mu} (x'_G, \bar{u}') \text{ em } (\exists W G_{\mathrm{v}})_{|\mathrm{U}}.$$

De $(x_E, x_G, \bar{u}) \in X$ tem-se, por (6.18), que

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}) \text{ para algum } \bar{w} \in \mathrm{dom}(W).$$

Então,

$$G_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{u}, \bar{w}).$$

Como G_{v} é total e como

$$(x_G, \bar{u}) \xrightarrow{\mu} \text{ em } (\exists W G_{\mathrm{v}})_{|\mathrm{U}},$$

existe $\bar{u}'$ e $\bar{w}'$, tal que

$$G_{\mathrm{v}} \xrightarrow{s} (x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_G, \bar{u}', \bar{w}').$$

Pela V-controlabilidade de $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ e.r.a. G_{v}, tem-se que

$$\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) \xrightarrow{s} (x_E, x_G, \bar{u}, \bar{w}) \xrightarrow{\mu} (x'_E, x'_G, \bar{u}', \bar{w}'),$$

para algum estado x'_E de E_{v}, implicando que $(x'_E, \bar{u}', \bar{w}') \in X$, por (6.18). Como além disso $(x'_E, x'_G, \bar{u}') \in \mathbf{EF}\mathcal{C}_{\mathbf{U}}$, por (6.20), então é verdade que $(x'_E, x'_G, \bar{u}') \in X \cap \mathbf{EF}\mathcal{C}_{\mathbf{U}}$, o que mostra que X é um conjunto de estados fortemente controláveis e, portanto,

$$X \subseteq \widehat{\mathbf{EF}\mathcal{C}_{\mathbf{U}}}. \tag{6.21}$$

Resta então mostrar que

$$L(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) \subseteq L(\hat{K}_{\mathrm{V}}). \tag{6.22}$$

Seja $s = \sigma_1 \ldots \sigma_n$, tal que $s \in L(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$. Então,

$$\begin{aligned}\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}) &\xrightarrow{\varepsilon} (x_E^0, x_G^0, \bar{u}^0, \bar{w}^0) \xrightarrow{\sigma_1} (x_E^1, x_G^1, \bar{u}^1, \bar{w}^1) \\ &\xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_E^n, x_G^n, \bar{u}^n, \bar{w}^n).\end{aligned} \tag{6.23}$$

Será mostrado que $s \in L(\hat{K}_{\mathrm{V}})$. De (6.23), tem-se que

$$G_{\mathrm{v}} \xrightarrow{\varepsilon} (x_G^0, \bar{u}^0, \bar{w}^0) \xrightarrow{\sigma_1} (x_G^1, \bar{u}^1, \bar{w}^1) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_G^n, \bar{u}^n, \bar{w}^n) \tag{6.24}$$

e que

$$E_{\mathrm{v}} \xrightarrow{\varepsilon} (x_E^0, \bar{u}^0, \bar{w}^0) \xrightarrow{\sigma_1} (x_E^1, \bar{u}^1, \bar{w}^1) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_E^n, \bar{u}^n, \bar{w}^n). \tag{6.25}$$

Logo, pelas definições 21 e 22, também é verdade que

$$\begin{aligned}(\exists W E_{\mathrm{v}} \| \exists W G_{\mathrm{v}})_{|U} &\xrightarrow{\varepsilon} (x_E^0, x_G^0, \bar{u}^0) \xrightarrow{\sigma_1} (x_E^1, x_G^1, \bar{u}^1) \\ &\xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_E^n, x_G^n, \bar{u}^n).\end{aligned} \tag{6.26}$$

De (6.18) e (6.23), pode-se afirmar que, para todo $k = 0, \cdots n$,

$$(x_E^k, x_G^k, \bar{u}^k) \in X \subseteq \widehat{\mathbf{EFC_U}}, \text{ por (6.21)}.$$

Assim,

$$\hat{K}_{\mathrm{U}} = (\exists W G_{\mathrm{v}} \mathbin{/\!\!/} \exists W E_{\mathrm{v}})_{|\mathrm{U}|\widehat{\mathbf{EFC_U}}} \xrightarrow{\varepsilon} (x_E^0, x_G^0, \bar{u}^0) \xrightarrow{\sigma_1} (x_E^1, x_G^1, \bar{u}^1) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_E^n, x_G^n, \bar{u}^n). \tag{6.27}$$

Resta ser provado que s também é uma cadeia possível em $\hat{K}_{\mathrm{V}}$. Primeiro, note que

$$(x_E^0, x_G^0, \bar{u}^0, \bar{w}^0) \in Q^\circ,$$

pois $(x_E^0, x_G^0, \bar{u}^0) \in X \subseteq \widehat{\mathbf{EFC_U}}$ e, então, $(x_E^0, x_G^0, \bar{u}^0, \bar{w}^0) \in \mathbf{EFC_V}$, por (6.11). Agora, considere a transição

$$(x_E^k, x_G^k, \bar{u}^k, \bar{w}^k) \xrightarrow{\sigma_{k+1}} (x_E^{k+1}, x_G^{k+1}, \bar{u}^{k+1}, \bar{w}^{k+1}), \text{ em (6.23)}.$$

Então:

- De (6.27), tem-se que

  $$(x_E^k, x_G^k, \bar{u}^k) \xrightarrow{\sigma_{k+1}} (x_E^{k+1}, x_G^{k+1}, \bar{u}^{k+1}), \text{ em } \hat{K}_{\mathrm{U}},$$

  o que implica que existe uma transição

  $$(x_E^k, x_G^k) \xrightarrow{\sigma_{k+1}:\hat{p}} (x_E^{k+1}, x_G^{k+1}), \text{ tal que } \hat{p}(\bar{u}^k, \bar{u}^{k+1}) = \mathit{true};$$

- De (6.25), tem-se que

  $$(x_E^k, \bar{u}^k, \bar{w}^k) \xrightarrow{\sigma_{k+1}} (x_E^{k+1}, \bar{u}^{k+1}, \bar{w}^{k+1}) \text{ em } E_{\mathrm{v}},$$

  o que implica que existe uma transição

  $$x_E^k \xrightarrow{\sigma_{k+1}:p_E} x_E^{k+1}, \text{ tal que } p_E(\bar{u}^k, \bar{w}^k, \bar{u}^{k+1}, \bar{w}^{k+1}) = \mathit{true};$$

- Finalmente,

  $$(x_E^{k+1}, x_G^{k+1}, \bar{u}^{k+1}) \in X \cap \widehat{\mathbf{EFC_U}}$$

implica que

$$(x_E^{k+1}, x_G^{k+1}, \bar{u}^{k+1}, \bar{w}^{k+1}) \in \mathbf{EFC_V}, \text{ por (6.11)}$$

o que, por sua vez, implica que

$$p_{\mathbf{EFC}}\langle x_E^{k+1}, x_G^{k+1}\rangle(\bar{u}^{k+1}, \bar{w}^{k+1}) = true.$$

Dos três itens acima, tem-se que $\hat{K}_{\mathrm{V}}$ possui uma transição

$$(x_E^k, x_G^k) \xrightarrow{\sigma_{k+1}:\hat{p}\wedge p_E \wedge p_{\mathbf{EFC}}\langle x_E^{k+1}, x_G^{k+1}\rangle} (x_E^{k+1}, x_G^{k+1}).$$

Como isso vale para todo k, então isso é suficiente para provar que

$$\hat{K}_V \xrightarrow{\varepsilon} (x_E^0, x_G^0, \bar{u}^0, \bar{w}^0) \xrightarrow{\sigma_1} (x_E^1, x_G^1, \bar{u}^1, \bar{w}^1) \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} (x_E^n, x_G^n, \bar{u}^n, \bar{w}^n) \tag{6.28}$$

e, assim, $s \in L(\hat{K}_{\mathrm{V}}) \supseteq L(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$, provando 6.22.

Para, por fim, provar o Teorema 8, seja $s \in L(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$. Então, claramente, $s \in L(G_{\mathrm{v}})$ e, por (6.22), $s \in L(\hat{K}_{\mathrm{V}})$. Como, pelo Teorema 7, $\hat{K}_{\mathrm{V}} = \mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}})$, então $s \in L(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}})$. Portanto,

$$L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) \subseteq L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}).$$

■

O Teorema 8 provê condições sob as quais uma solução de controle, obtida via abstração de variáveis, sempre contém a solução minimamente restritiva, obtida sem abstrações, i.e., $L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) \subseteq L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}})$. Em particular, isso ocorre sempre que a Hipótese 1 é satisfeita. Como, pelo Teorema 6, o contrário também é verdadeiro, i.e., $L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}) \subseteq L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}))$, então, sob tais condições,

$$L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}) = L^\omega(\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})).$$

A partir do Teorema 8, o corolário seguinte pode ser enunciado.

Corolário 6 *Sob as mesmas condições do Teorema 8, se também* $\hat{K}_{\mathrm{V}} /\!\!/ G_{\mathrm{v}}$ *é não-bloqueante, então* $\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists WG_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}}$ *é a solução ótima para o PCS-V.*

Prova: Decorre diretamente dos teoremas 6 e 8. ■

Para os casos em que a Hipótese 1 não é verdadeira e, assim, a otimidade não pode ser determinada, torna-se difícil construir uma abstração melhorada, que possa levar o resultado de síntese à máxima permissividade.

No entanto, de posse do algoritmo proposto, pode-se determinar se a Hipótese 1 é, ou não, verdadeira. Do mesmo modo, quando não for, pode-se apontar quais são as variáveis necessárias à síntese. Ao incorporar tais variáveis à abstração, cria-se, na verdade, uma classe de abstrações que sempre garante o resultado minimamente restritivo. O procedimento de identificação de variáveis "necessárias" à síntese, é ilustrado a seguir, no contexto de um exemplo.

6.2.5 Um exemplo de síntese com abstrações de AFEs

Considere o exemplo de uma pequena fábrica, como mostra a Figura 28.

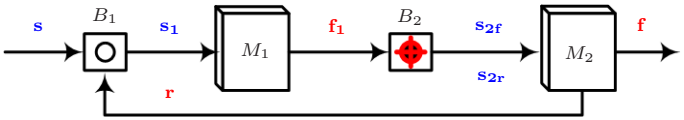


Figura 28 – Pequena fábrica com retroalimentação de materiais.

O sistema é composto por duas máquinas, M_1 e M_2, interligadas pelos respectivos *buffers* unitários, B_1 e B_2. O *buffer* B_1 é alimentado com peças recebidas de fora do sistema (evento s), ou do retrabalho (evento r).

A máquina M_1 remove as peças de B_1 (evento s_1), as manufatura, e as coloca em B_2 (evento f_1). Em B_2, um sensor testa a qualidade das peças, determinando o tipo de operação a ser efetuada em M_2 (s_{2f} ou s_{2r}), o que leva respectivamente as peças a deixarem o sistema (evento f) ou a serem redirecionadas a B_1, para retrabalho (evento r). Os eventos f_1, f e r são não-controláveis (em vermelho), enquanto os demais são controláveis (em azul).

O objetivo de controle é evitar o *overflow* e o *underflow* dos *buffers* e limitar em 1 o número de ciclos de trabalho a serem efetivados em cada peça. As variáveis b_1, m_1, b_2 e m_2, com domínios $\mathrm{dom}(b_1) = \mathrm{dom}(m_1) = \mathrm{dom}(b_2) = \mathrm{dom}(m_2) = \{0, 1, 2\}$ e valores inici-

ais $b_1^\circ = m_1^\circ = b_2^\circ = m_2^\circ = 0$, são definidas para facilitar a modelagem das especificações. Assim, ao modelar a planta do sistema, essas variáveis são atualizadas conforme uma semântica tal que a atribuição do valor 0 representa a ausência de uma peça, enquanto a atribuição dos valores 1 e 2 representa as peças durante o primeiro e o segundo ciclo, respectivamente. Os modelos da planta são apresentados na Figura 29.

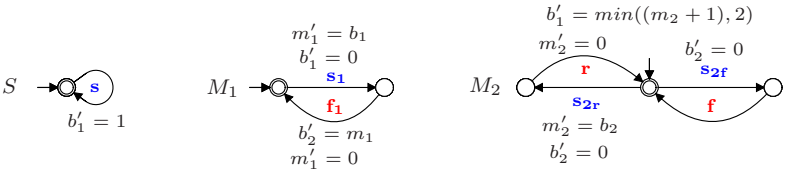


Figura 29 – Modelagem da planta do sistema.

Basicamente, quando uma peça é descarregada pela primeira vez em B_1 (evento s), o valor corrente da variável b_1 (0) é atualizado para 1, indicando o início do primeiro ciclo de trabalho. Em seguida, quando M_1 inicia (evento s_1), o valor de b_1 é transferido para a variável m_1 ($m'_1 = b_1$), e o valor de b_1 é reinicializado ($b'_1 = 0$), já que o *buffer* B_1 está novamente vazio. Posteriormente, quando M_1 finaliza a manufatura (evento f_1), b_2 recebe o valor de m_1 ($b'_2 = m_1$) e m_1 é reinicializada. Se um retrabalho acontece, (M_2 inicia com o evento s_{2r}), m_2 recebe o valor de b_2 ($m'_2 = b_2$) e b_2 é reinicializada. Nesse caso, a transição com r atualiza b_1 com $b'_1 = min((m_2 + 1), 2)$, identificando o novo ciclo que se inicia. Por clareza, a Figura 29 oculta as fórmulas sobre variáveis cujos valores não mudam no próximo estado.

A partir das variáveis atualizadas pela planta, as especificações de controle para o exemplo podem ser facilmente modeladas pelos AFEs apresentados na Figura 30. O *overflow* e o *underflow* em B_1 e B_2 são evitados, respectivamente, pelas especificações O_1, U_1, O_2, e U_2, enquanto o AFE Max especifica o limite de ciclos de trabalho.

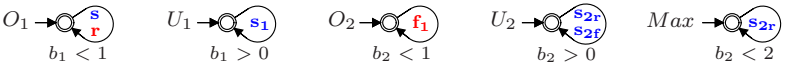


Figura 30 – Modelos para as especificações de controle.

O AFE $G_v = S \,/\!/\, M_1 \,/\!/\, M_2$ modela a composição da planta e possui 6 estados. Já a composição das especificações, $E_v = O_1 \,/\!/\, U_1 \,/\!/$

$O_2 /\!/ U_2 /\!/ Max$, possui apenas 1 estado. A interpretação explícita da composição $E_{\mathrm{v}} /\!/ G_{\mathrm{v}}$ e o resultado de síntese, $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, possuem respectivamente 156 e 18 estados.

Na sequência, o exemplo é tratado com o uso de abstrações e a síntese é processada através do algoritmo proposto.

6.2.5.1 Síntese com a abstração em W = V

A primeira abstração a ser considerada é construída através da remoção de todas as variáveis da síntese, i.e., $W = V = \{b_1, m_1, b_2, m_2\}$. O AFE modelando $(\exists W E_{\mathrm{v}} /\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$ é mostrado na Figura 31.

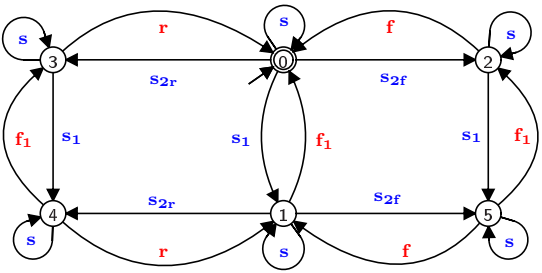


Figura 31 – Abstração $\exists b_1\, \exists m_1\, \exists b_2\, \exists m_2\, E_{\mathrm{v}} /\!/ \exists b_1\, \exists m_1\, \exists b_2\, \exists m_2\, G_{\mathrm{v}}$.

A seguir, é obtido um supervisor a partir dessa abstração. Assume-se que $(\mathsf{x}, b_1, m_1, b_2, m_2)$ denota a associação do estado x aos valores das variáveis b_1, m_1, b_2 e m_2, e * denota uma variável que pode assumir todos os seus valores. A síntese de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$, para $W = \{b_1, m_1, b_2, m_2\}$, é como segue.

Passo 1 - Restrição de variáveis a U

Note que, nesse caso,

$$U = \emptyset.$$

Ou seja, b_1, m_1, b_2 e m_2 são abstraídas e podem assumir valor $*$ em cada estado;

Passo 2 - Estados Fortemente Controláveis em V

$\mathbf{EF}\mathcal{C}_{\mathrm{V}} = \{(0,*,*,*,*),(1,*,*,0,*),(2,*,*,*,*),(3,0,*,*,*),(4,0,*,0,*),(5,*,*,0,*)\}$.

O conjunto $\mathbf{EF}\mathcal{C}_{\mathrm{V}}$ filtra os estados da abstração (Figura 31), preservando apenas os valores das variáveis com os quais nenhuma transição não-controlável é desabilitada.

Passo 3 - Estados Fortemente Controláveis em U

O conjunto $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$ é tal que

$$\mathbf{EF}\mathcal{C}_{\mathrm{U}} = \{0,2\}.$$

Apenas os estados 0 e 2 definem $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$, pois apenas $(0,*,*,*,*)$ e $(2,*,*,*,*)$, em $\mathbf{EF}\mathcal{C}_{\mathrm{V}}$, não desabilitam transições não-controláveis para todos os valores das variáveis.

O estado $(1,*,*,0,*)$, por exemplo, falha para habilitar uma transição não-controlável quando $b_2 > 0$ e, portanto, $1 \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$. O mesmo se aplica aos estados 3, 4 e 5;

Passo 4 - Estados Levemente Controláveis em U

Agora, define-se $\mathbf{EL}\mathcal{C}_{\mathrm{U}}$, tal que

$$\mathbf{EL}\mathcal{C}_{\mathrm{U}} = \{0,1,2,3,4,5\}.$$

$\mathbf{EL}\mathcal{C}_{\mathrm{U}}$ contém todos os estados da abstração pois, em cada estado, toda transição não-controlável que parte é habilitada por algum valor de variável.

Passo 5 - Conjunto supremo de Estados Fortemente Controláveis

Para determinar o conjunto $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$, considere o conjunto de estados

$$X = \{0,1,2,3,4,5\}.$$

A computação de $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$, a partir de X, pode ser informalmente descrita como segue:

(i) Cheque se, a partir de cada estado $x \in X$, parte alguma transição não-controlável;

(ii) Se não parte, então x é seguro e permanece em X;

(iii) Se parte, então x permanece em X se, e somente se:

- o estado alcançado x' ainda está em X, i.e., não foi removido pela síntese; e
- x' é fortemente controlável, i.e., $x' \in \mathbf{EF}\mathcal{C}_{\mathrm{U}}$.

Seguindo-se esse raciocínio, pode-se notar que o estado $\mathsf{0}$ é seguro, pois as transições partindo de $\mathsf{0}$ são todas controláveis. Os estados $\mathsf{1}$, $\mathsf{2}$ e $\mathsf{3}$ possuem as transições não-controláveis $\mathsf{1} \xrightarrow{f_1} \mathsf{0}$, $\mathsf{2} \xrightarrow{f} \mathsf{0}$, e $\mathsf{3} \xrightarrow{r} \mathsf{0}$. Porém, como o estado alcançado por essas transições é $\mathsf{0} \in X \cap \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, então todos eles são seguros e permanecem em X.

O estado $\mathsf{5}$, embora seja seguro para a transição $\mathsf{5} \xrightarrow{f_1} \mathsf{2} \in X \cap \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, falha para a transição $\mathsf{5} \xrightarrow{f} \mathsf{1} \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$ e, portanto, é removido de X. Por razão equivalente, o estado $\mathsf{4}$ também é removido.

Como nenhum outro estado precisa ser removido,

$$X = \widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}} = \{\mathsf{0}, \mathsf{1}, \mathsf{2}, \mathsf{3}\}$$

é o conjunto final de estados fortemente controláveis.

Análise de otimidade

Resta saber se os estados removidos, $\mathsf{4}$ e $\mathsf{5}$, são *sempre* inseguros, ou apenas na síntese com abstrações. Essa checagem pode ser concretizada ao analisar os estados alcançados a partir de $\mathsf{4}$ e $\mathsf{5}$ com transições não-controláveis. Na prática, tais estados representam a causa pela qual $\mathsf{4}$ e $\mathsf{5}$ são maus-estados.

Note que, a partir de $\mathsf{4}$ e $\mathsf{5}$, alcança-se $\mathsf{1}$ com transições não-controláveis. Como $\mathsf{1} \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, pois a transição $\mathsf{1} \xrightarrow{f_1} \mathsf{0}$ não pode ocorrer quando $b_2 > 0$ (pela especificação O_2), então $\mathsf{4}$ e $\mathsf{5}$ são maus-estados porque podem alcançar $\mathsf{1}$ atualizando b_2 com qualquer valor $b_2 > 0$, com uma transição não-controlável. Por outro lado, $\mathsf{1}$ é levemente controlável, pois quando $b_2 = 0$, a transição $\mathsf{1} \xrightarrow{f_1} \mathsf{0}$ é habilitada e, nesse caso, $\mathsf{4}$ e $\mathsf{5}$ seriam seguros.

Assim, embora o resultado de síntese seja controlável e possa ser implementado como solução, não há garantias de que ele seja o menos restritivo, pois $\mathsf{1} \in \mathbf{EL}\mathcal{C}_{\mathbf{U}}$ e $\mathsf{1} \notin \mathbf{EF}\mathcal{C}_{\mathbf{U}}$ não satisfaz a Hipótese 1. Alternativamente, pode-se sintetizar um novo supervisor, dessa vez usando uma abstração que preserva as variáveis necessárias à síntese. Para isso, é necessário informar quais são as variáveis necessárias, o que é investigado a seguir.

Identificação de variáveis necessárias à síntese

Uma forma de apontar quais variáveis são necessárias para melhorar uma abstração se dá através da análise do estado causador de maus-estados, nesse caso 1, no contexto do conjunto $\mathbf{EF}\mathcal{C}_{\mathrm{V}}$. Note que, enquanto $(1,*,*,0,*) \in \mathbf{EF}\mathcal{C}_{\mathrm{V}}$, $1 \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, porque a variável b_2 somente habilita a transição $1 \xrightarrow{f_1} 0$ quando $b_2 = 0$ (pela especificação O_2).

Contudo, se a variável b_2 for mantida em síntese, então ocorre que, para os estados 1i, tal que $\mathrm{i} = 0, 1, 2$ é o valor assumido por b_2, $10 \in \mathbf{EF}\mathcal{C}_{\mathrm{U}}$ e $11,\ 12 \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$.

Ou seja, nesse caso, tornar-se-ia possível incluir o estado 10 no conjunto $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$, o que, por sua vez, poderia permitir preservar em síntese estados que antes eram inseguros. A síntese, apresentada a seguir, considera essa nova abstração.

6.2.5.2 Síntese com a abstração em $W = \{b_1, m_1, m_2\}$

A segunda abstração considerada neste exemplo é construída através da remoção de todas as variáveis em V, exceto b_2, i.e., $W = \{b_1, m_1, m_2\}$. O AFE modelando $(\exists W E_{\mathrm{v}} \mathbin{/\!\!/} \exists W G_{\mathrm{v}})_{|\mathrm{U}}$ possui, fundamentalmente, a estrutura mostrada na Figura 32. Porém, agora, os valores da variável $b_2 = 0, 1, 2$ são associados a cada estado do autômato, o que significa que, explicitamente, a abstração passa a envolver $6 * 3 = 18$ estados.

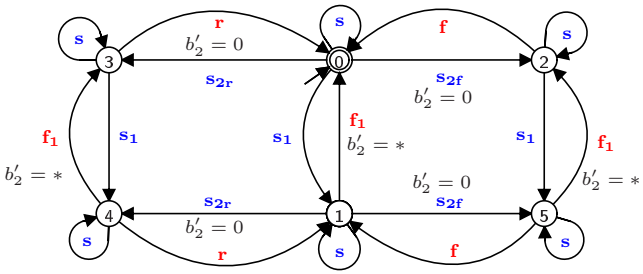


Figura 32 – Abstração $\exists b_1\, \exists m_1\, \exists m_2\, E_{\mathrm{v}} \mathbin{/\!\!/} \exists b_1\, \exists m_1\, \exists m_2\, G_{\mathrm{v}}$.

Assume-se, na figura, que as transições que não atualizam b_2, implementam a fórmula $b_2' = b_2$. Nesse caso, cada estado x possui

a estrutura $(\mathsf{x}b_2, b_1, m_1, m_2)$, e a síntese de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$, para $W = \{b_1, m_1, m_2\}$, é como segue.

Passo 1 - Restrição de variáveis a U

$$U = \{b_2\};$$

Passo 2 - Estados Fortemente Controláveis em V

$\mathbf{EF}\mathcal{C}_{\mathrm{V}} = \{(0*, *, *, *), (10, *, *, *), (2*, *, *, *), (3*, 0, *, *), (40, 0, *, *), (50, *, *, *)\}$;

Passo 3 - Estados Fortemente Controláveis em U

$$\mathbf{EF}\mathcal{C}_{\mathrm{U}} = \{0*, 10, 2*, 50\};$$

Note que, agora, os estados 10 e 50 foram adicionados a $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$ já que, nesses estados, $b_2 = 0$ satisfaz a especificação O_2. As demais combinações (estados 11, 12, 51 e 52) não são nem fortemente controláveis, nem levemente controláveis, pois os valores assumidos por b_2 nunca satisfazem O_2. Portanto, tais estados não compõem $\mathbf{EF}\mathcal{C}_{\mathrm{U}}$.

O estado 40 também se torna controlável para a transição $40 \xrightarrow{f_1} 3*$. Contudo, ele ainda falha para a transição não-controlável $40 \xrightarrow{r} 10$, a qual depende adicionalmente de outra variável que será identificada nesta síntese;

Passo 4 - Estados Levemente Controláveis em U

$$\mathbf{EL}\mathcal{C}_{\mathrm{U}} = \{0*, 10, 2*, 3*, 40, 50\};$$

Passo 5 - Conjunto supremo de Estados Fortemente Controláveis

Agora, para encontrar o conjunto fortemente controlável de

estados, $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$, seja

$$X = \{0*, 1*, 2*, 3*, 4*, 5*\}.$$

O estado $0*$ é novamente seguro; já os estados 11 e 12 são inseguros, pois $b_2 > 0$ implica (por O_2) desabilitar a transição não-controlável $11 \xrightarrow{f_1} 0*$ e $12 \xrightarrow{f_1} 0*$. Portanto, esses estados são removidos de X. Contudo, agora o estado 10 se torna seguro pois, nesse estado, $b_2 = 0$ e, assim, $10 \xrightarrow{f_1} 0*$ alcança $0* \in X \cap \mathbf{EF}\mathcal{C}_{\mathrm{U}}$. Os estados $2*$ e $3*$ também alcançam $0*$ e, portanto, são seguros. O estado 50 também se torna seguro, uma vez que as transições $50 \xrightarrow{f} 10$ e $50 \xrightarrow{f_1} 2*$ alcançam 10 e $2*$, ambos em $X \cap \mathbf{EF}\mathcal{C}_{\mathrm{U}}$. As demais combinações, 51 e 52, são removidas.

Note que o estado $4*$ permanece inseguro. Na verdade, após manter b_2 em síntese, $4*$ se torna seguro para a transição $4* \xrightarrow{r} 10$, mas ainda é inseguro para $4* \xrightarrow{f_1} 3*$, uma vez que $3* \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$. Assim, $4*$ é, novamente, removido de X.

Como nenhum outro estado precisa ser eliminado, o conjunto supremo de estados fortemente controláveis é dado por

$$X = \widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}} = \{0*, 10, 2*, 3*, 50\}.$$

Note que $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$ foi melhorado em relação à síntese anterior, já que passou a conter um novo estado, 50. Resta checar se esse conjunto leva à solução ótima.

Análise de otimidade

O estado $4*$ é removido de X porque o estado alcançado por ele com uma transição não-controlável, $3*$, não é fortemente controlável. De fato, a partir de $3*$, a transição não-controlável $3* \xrightarrow{r} 0*$ é desabilitada quando $b_1 > 0$ (por O_1). Então, $4*$ é inseguro porque pode alcançar $3*$ atualizando b_1 a qualquer valor $b_1 > 0$.

Mas observe que $3*$ é levemente controlável, pois quando $b_1 = 0$ em $3*$, a transição $3* \xrightarrow{r} 0*$ é habilitada e, nesse caso, $4*$ é seguro. Portanto, como $3* \in \mathbf{EL}\mathcal{C}_{\mathrm{U}}$ e $3* \notin \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, não se pode garantir que a síntese é ótima, pela Hipótese 1.

Identificação das variáveis necessárias à síntese

Analisando-se o estado $(3*, 0, *, *) \in \mathbf{EF}\mathcal{C}_{\mathrm{V}}$, pode-se observar a necessidade de manter a variável b_1 em síntese, além de b_2. Nesse caso, ocorreria que $(30*, *, *) \in \mathbf{EF}\mathcal{C}_{\mathrm{V}}$ e, assim, $30* \in \mathbf{EF}\mathcal{C}_{\mathrm{U}}$, para 0 e $*$ modelando, respectivamente, os valores de b_1 e b_2, associados ao estado 3. Na próxima etapa de síntese, constrói-se uma abstração que preserva b_1 e b_2.

6.2.5.3 Síntese com a abstração em $W = \{m_1, m_2\}$

Considere abstrair apenas as variáveis m_1 e m_2 na planta, i.e., $W = \{m_1, m_2\}$. O AFE, modelando $(\exists W E_{\mathrm{v}} /\!\!/ \exists W G_{\mathrm{v}})_{|\mathrm{U}}$, possui a estrutura mostrada na Figura 33. Porém, aqueles 6 estados são associados aos valores das variáveis $b_1 = 0, 1, 2$ e $b_2 = 0, 1, 2$, o que resulta numa composição com 50 estados alcançáveis.

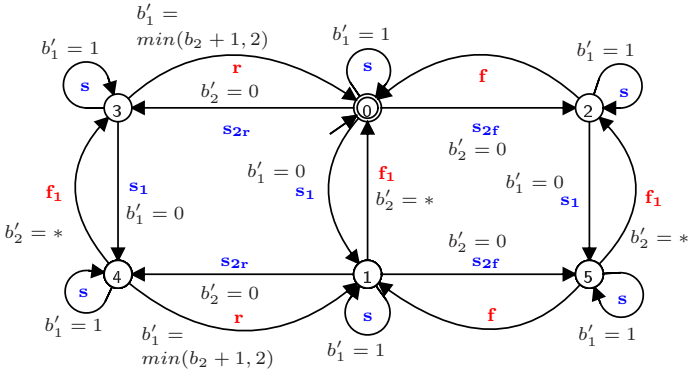


Figura 33 – Abstração $\exists m_1\, \exists m_2\, E_{\mathrm{v}} /\!\!/ \exists m_1\, \exists m_2\, G_{\mathrm{v}}$.

Agora, cada estado x possui a estrutura $(\mathsf{x} b_1 b_2, m_1, m_2)$, e a síntese de $\sup\mathcal{C}_V(E_{\mathrm{v}}, \exists W G_{\mathrm{v}})$, para $W = \{m_1, m_2\}$, é como segue.

Passo 1 - Restrição de variáveis a U

$$U = \{b_1, b_2\};$$

Passo 2 - Estados Fortemente Controláveis em V

$\mathbf{EF}\mathcal{C}_{\mathrm{V}} = \{(0**,*,*),(1*0,*,*),(2**,*,*),(30*,*,*),(400,*,*),(5*0,*,*)\}$;

Passo 3 - Estados Fortemente Controláveis em U

$$\mathbf{EF}\mathcal{C}_{\mathrm{U}} = \{0**, 1*0, 2**, 30*, 400, 5*0\};$$

Agora, os estados $30*$ e 400 também são fortemente controláveis, pois os valores de b_1 e b_2, retidos em cada estado, habilitam todas as transições não-controláveis que partem.

Passo 4 - Estados Levemente Controláveis em U

$$\mathbf{EL}\mathcal{C}_{\mathrm{U}} = \{0**, 1*0, 2**, 30*, 400, 5*0\};$$

Passo 5 - Conjunto supremo de Estados Fortemente Controláveis

Determina-se, a seguir, o conjunto $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}}$. Seja

$$X = \{0**, 1**, 2**, 3**, 4**, 5**\}.$$

O estado $0**$ é novamente seguro. $1*0$, $2**$, $30*$ e $5*0$ também são seguros, pois alcançam $0**$ e $1*0$, ambos em $X \cap \mathbf{EF}\mathcal{C}_{\mathrm{U}}$. As demais combinações correspondentes são eliminadas de X.

Dessa vez, também o estado 400 passa a ser seguro, pois as transições $400 \xrightarrow{r} 1*0$ e $400 \xrightarrow{f_1} 30*$ alcançam estados fortemente controláveis, também em X. Além disso, as demais combinações de estados $4**$ não são fortemente, nem levemente controláveis, i.e., tais estados são sempre maus-estados, e removidos de X.

Assim, como nenhum outro estado precisa ser eliminado de X,

$$X = \widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}} = \{0**, 1*0, 2**, 30*, 400, 5*0\}.$$

Análise de otimidade

Observe que, agora, $\mathbf{EF}\mathcal{C}_{\mathrm{U}} = \mathbf{EL}\mathcal{C}_{\mathrm{U}}$ satisfaz a Hipótese 1. Assim, pelos teoremas 6 e 8, tem-se que o resultado minimamente

restritivo, i.e., $\widehat{\mathbf{EF}\mathcal{C}_{\mathrm{U}}} = X$ leva a um AFE, representando o supervisor $\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists m_1\, \exists m_2\, G_{\mathrm{v}})$ e modelado com 18 estados, tal que

$$\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists m_1\, \exists m_2\, G_{\mathrm{v}}) /\!\!/ G_{\mathrm{v}} = \mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}).$$

Ressalta-se que $\mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, \exists m_1\, \exists m_2\, G_{\mathrm{v}})$ é computado a partir de um AFE com 50 estados, ao invés de 156, como na síntese sem abstrações.

6.2.6 Sistema de manufatura modelado com abstrações

Retoma-se, aqui, o exemplo do sistema de manufatura apresentado na Seção 6.1.6, o qual é tratado agora com o uso de abstrações.

A primeira abstração a ser usada é construída considerando-se que $W = V$, i.e., todas as variáveis são removidas da síntese. Esse efeito é reproduzido na planta modelada com AFEs (Figura 27, Seção 6.1.6) pela abstração $G_{\mathrm{v}}^1 = \exists b_1\, \exists m_1\, \exists b_2\, \exists m_2\, \exists b_3\, \exists tu\, G_{\mathrm{v}}$.

Então, a planta G_{v}^1 e a especificação E_{v} são usados para obter um supervisor $V_{\mathrm{v}}^1 = \mathrm{sup}\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}^1)$. A segunda linha da Tabela 10 apresenta o número de estados desses autômatos. Para fins comparativos, a primeira linha da tabela reproduz a síntese sem abstrações.

Tabela 10 – Síntese de controle usando *abstração de variáveis*.

G_{v}	E_{v}	$E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$	V_{v}	$V_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$
3024	8	7220	212	**212**
G_{v}^1	E_{v}	$E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}^1$	V_{v}^1	$V_{\mathrm{v}}^1 /\!\!/ G_{\mathrm{v}}$
24	8	192	59	***276***
G_{v}^2	E_{v}	$E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}^2$	V_{a}^2	$V_{\mathrm{v}}^2 /\!\!/ G_{\mathrm{v}}$
144	8	768	103	**212**

Note que o esforço computacional para obter V_{v}^1 é muito menor do que para obter V_{v}, pois a síntese é conduzida sobre o AFE $E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}^1$, com 192 estados, enquanto que $E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}$ possui 7.220 estados. Além do mais, é possível mostrar que $V_{\mathrm{v}}^1 /\!\!/ G_{\mathrm{v}}$ é não-bloqueante e, portanto, pode ser usado para implementar uma solução para o PCS-V.

Porém, pode também ser mostrado que, para essa abstração, $\mathbf{EF}\mathcal{C}_{\mathrm{U}} \neq \mathbf{EL}\mathcal{C}_{\mathrm{U}}$. Como essa diferença não satisfaz a Hipótese 1, então nada pode ser dito a respeito da otimidade da solução de controle, i.e., $V_{\mathrm{a}}^1 /\!\!/ G_{\mathrm{v}}$ pode ser mais restritivo do que V_{v}.

A razão pela qual a solução $V_{\mathrm{a}}^1 /\!\!/ G_{\mathrm{v}}$ pode ser subótima segue um raciocínio similar àquele explicado na síntese com distinguidores. Ou seja, como os valores das variáveis foram omitidos da síntese, então existe um ciclo de trabalho (o último ciclo de uma peça), que necessita ser completado para que, só então, o ingresso de uma nova peça seja admitido no sistema, caso esse esteja saturado. Por outro lado, conhecendo-se os valores das variáveis, sabe-se quando o ciclo é o último e, nesse caso, uma nova peça pode ser recebida tão logo o *buffer* B_1 esteja vazio. Logo, o resultado computado via abstrações é, de fato, mais restritivo.

No sentido de melhorar esse resultado, as variáveis m_2 e b_3 são incorporadas à síntese. A escolha dessas variáveis é automaticamente suportada pelo algoritmo proposto, ou, alternativamente, pode ser feita com base na interpretação informal do processo, tal como no exemplo com distinguidores. Além do mais, variáveis podem ser gradualmente incorporadas à síntese, até que se observem as melhorias esperadas no resultado.

Nesse sentido, seja G_{v}^2 uma nova abstração a ser usada neste exemplo, construída sobre o conjunto $W = \{b_1, m_1, b_2, tu\}$, tal que $G_{\mathrm{v}}^2 = \exists b_1\, \exists m_1\, \exists b_2\, \exists tu\, G_{\mathrm{v}}$. A planta G_{v}^2 e a especificação E_{v} são usadas para obter um supervisor $V_{\mathrm{v}}^2 = \sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}}^2)$. O número de estados desses modelos é mostrado na última linha da Tabela 10.

Agora, a síntese de V_{v}^2 se dá sobre uma composição $E_{\mathrm{v}} /\!\!/ G_{\mathrm{v}}^2$ com 768 estados, a qual, portanto, passa a ser mais complexa do que a síntese de V_{v}^1, efetivada sobre um modelo com 192 estados. Contudo, essa complexidade ainda é, substancialmente, menor do que a observada para computar V_{v}, envolvendo 7.220 estados.

Como $V_{\mathrm{v}}^2 /\!\!/ G_{\mathrm{v}}$ é não-bloqueante, então esse AFE pode ser usado para implementar uma solução para o PCS-V. Além disso, para essa nova abstração, a síntese é tal que $\mathbf{EF}\mathcal{C}_{\mathrm{U}} = \mathbf{EL}\mathcal{C}_{\mathrm{U}}$. Como essa igualdade satisfaz a Hipótese 1, então, pelos teoremas 6 e 8, $V_{\mathrm{a}}^2 /\!\!/ G_{\mathrm{v}} = V_{\mathrm{v}}$ é uma solução ótima para o exemplo.

6.3 SUMÁRIO

Este capítulo apresentou a síntese de supervisores para SEDs modelados por AFEs. A abordagem permite simplificar a modelagem de especificações de controle e, ao mesmo tempo, leva a uma solução de controle equivalente à síntese original, conduzida sobre AFs.

O preço a ser pago pela simplificação na modelagem das especi-

ficações é que o modelo da planta se torna mais complexo. Isso se dá pela associação de valores de variáveis aos estados do modelo, de modo que, computacionalmente, o esforço para obter uma solução de controle é equivalente, usando ou não AFEs.

Nesse sentido, introduz-se o conceito de abstrações de variáveis. As abstrações podem ser usadas para substituir a planta original (complexa) na síntese, proporcionando importantes ganhos computacionais, ao passo que o modelo das especificações permanecem simples e a controlabilidade do resultado é preservada.

Ainda, apresenta-se um algoritmo para computar supervisores a partir de abstrações, juntamente com as condições sob as quais a solução computada, através de abstrações, é ótima.

As vantagens da abordagem proposta neste capítulo foram ilustradas no contexto de dois exemplos. O primeiro exemplo ilustra a síntese de supervisores a partir de abstrações e, o segundo, contextualiza genericamente a proposta e permite compará-la aos demais resultados apresentados na tese.

7 RELACIONANDO DISTINGUIDORES E AUTÔMATOS ESTENDIDOS

Nos capítulos 4 e 6, o termo *complexidade* foi abordado, no contexto da TCS, sob duas perspectivas distintas: (i) para se referir ao esforço de modelagem de especificações e (ii) para denotar o esforço computacional da síntese de supervisores. Duas alternativas foram propostas para lidar com esses aspectos: o uso de distinguidores e o uso de AFEs.

Neste capítulo, essas abordagens voltam a ser exploradas, porém, de maneira associativa e comparativa. Embora distinguidores e AFEs tratem da mesma natureza de problema, na prática não se observa uma relação explícita entre eles. A análise a ser apresentada na sequência, compara, informalmente, os passos descritos nos capítulos 4 e 6 como base para a respectiva construção de distinguidores e de AFEs, apresentando exemplos de como se dá a relação entre cada passo, em cada abordagem. A Tabela 11 reproduz paralelamente esses passos.

Tabela 11 – Etapas da construção de distinguidores e de AFEs.

Distinguidores	**AFEs**
Identificação	
de eventos	*de variáveis*
eventos no sistema que, quando refinados em novos eventos, podem facilitar a tarefa de modelagem	variáveis cujos valores, quando adequadamente atualizados, podem facilitar a tarefa de modelagem
Estruturação	
instâncias de refinamentos	*domínios de variáveis*
quantidade de eventos refinados associados a determinada máscara	domínio a ser assumido por determinada variável
Complementação	
refinamentos complementares	*variáveis complementares*
eventos adicionais cujos refinamentos são necessários para a distinção de eventos elementares	variáveis adicionais cujos valores são necessários para a atualização adequada das variáveis elementares
Modelagem	
do distinguidor	*da lógica de atualização de variáveis*
construção de um modelo para distinguir, na planta, a ocorrência de cada instância de refinamento, conforme o comportamento do sistema	implementação de fórmulas lógicas para atualizar as variáveis de modo a associar cada valor a aspectos do comportamento do sistema

A seguir, cada associação apresentada na Tabela 11 é discutida, comparada e exemplificada.

- Etapa de *identificação*:

  A etapa de identificação, seja ela de eventos a serem refinados ou de variáveis a serem criadas, compartilha de dois aspectos em particular:

  (i) é feita com base na ocorrência de eventos no sistema;

  (ii) é justificada por um problema de modelagem.

  Informalmente, eventos e variáveis são identificados dentro do seguinte contexto do sistema, respectivamente:

  – Certo evento, se refinado em novos eventos, facilita modelar tal problema.

  – Certo evento, se ao ocorrer atribui a uma variável determinado valor, facilita modelar tal problema.

  Em suma, parte-se do modelo de um SED, que naturalmente provê um conjunto de eventos, e exploram-se novas perspectivas de ocorrência de alguns desses eventos. Em geral, isso se justifica perante a insuficiência de informações sobre o evento original, para que um problema possa ser modelado.

  Nesse sentido, a exploração de um problema em particular facilita o detalhamento dos passos da Tabela 11. Por isso, considere a estrutura parcial de um SED, mostrada na Figura 34, composta por uma máquina, M, que entra em operação após um evento α.

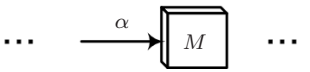


Figura 34 – Modelo parcial de um SED.

  Independentemente de relevância prática, suponha que, após 3 inícios de operações, M precisa ser parada para manutenção. Agora, tendo em mãos um sistema e um problema de controle, pode-se projetar uma forma de resolvê-lo. Comecemos, então, por modelar o sistema e a especificação, denotados respectivamente por G e E, da maneira convencional, como mostra a Figura 35.

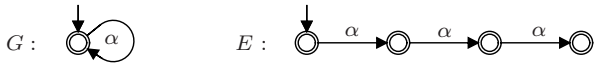


Figura 35 – Modelo da planta G e da especificação E, construídos convencionalmente.

O modelo da planta, G, consiste apenas de um auto-laço com o evento α, enquanto a especificação E, proibindo a 4ª operação de M, requer enumerar as 3 primeiras ocorrências de α. Claramente, tem-se uma condição de modelagem que justifica a exploração da ocorrência de α. Conforme a proposta desta tese, isso pode ser feito de duas maneiras:

(i) Refinando o evento após o qual se controla α (nesse caso o próprio α):
   - aqui, cada ocorrência de α deve ser rotulada por um novo evento α_i, que renomeia α conforme o contexto i em que ele ocorre, e cujo valor semântico é atribuído por um distinguidor.

(ii) Atualizando uma variável x junto ao evento após o qual se controla α (o próprio α):
   - já nesse caso, cada transição com α enriquece o estado alcançado (q, x), ao atribuir a x um valor que identifica o contexto em que α alcançou aquele estado. Esse valor facilita decidir sobre a próxima ocorrência de α e é atualizado conforme a estrutura lógica implementada sobre a transição.

Note que, dentro desse cenário, após uma transição etiquetada com o evento α existe, claramente, uma relação entre o contexto identificado por i, em α_i, e a semântica do valor atribuído a x. Esta tese não objetiva explorar formalmente tal relação, mas analisá-la, para propósitos práticos.

Resta, então, definir as estruturas dos refinamentos de α e da variável x, para que, por fim, se possa modelar o distinguidor e a formulação lógica para a atualização de x. Essas construções são acompanhadas passo-a-passo, na sequência.

- Etapa de *estruturação*:

  Neste passo, dimensionam-se as estruturas, dos refinamentos e das variáveis, identificadas no passo anterior. Definir quantas

instâncias de refinamentos serão associadas ao evento α, requer identificar sob quantas circunstâncias distintas α pode ocorrer no sistema. No mesmo sentido, um domínio para a variável x é definido com base nos valores com os quais x pode ser atualizado ao longo do sistema. Em ambos os casos, a decisão é tomada com base no problema de controle (especificação E).

Para o exemplo proposto, α pode ocorrer em 3 diferentes contextos (1ª, 2ª e 3ª operação da máquina M). Assim, o conjunto de refinamentos de α é definido por

$$\Delta^{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\}$$

e o domínio da variável x é definido por

$$\mathrm{dom}(x) = \{0, 1, 2, 3\}, \text{ com } x^{\circ} = 0.$$

A partir dos conjuntos Δ^{α} e $\mathrm{dom}(x)$, duas diferentes (mas equivalentes) versões para a especificação E, E_{d} e E_{v}, podem ser respectivamente modeladas, como na Figura 36.

Figura 36 – Versões refinadas do modelo da especificação E.

Ambas as especificações proíbem a máquina M de operar após a 3ª ocorrência de α. A diferença é que E_{d} implementa essa restrição através da desabilitação de $\alpha_i \in \Delta^{\alpha}$ após a ocorrência de α_3, enquanto E_{v} restringe α quando $x \geq 3$. Como as especificações E_{d} e E_{v} passam a ser modeladas por estruturas que independem do número de vezes que M pode operar, a complexidade de modelagem está mitigada.

- Etapa de *complementação*:

  Em particular, um complemento (evento ou variável) se justifica perante a dependência semântica entre os elementos identificados para simplificar um problema de modelagem (etapa de identificação), em relação a outros elementos do sistema, os quais ainda não tenham sido identificados por não influenciar diretamente no problema de modelagem.

Assim, essa etapa consiste em buscar por eventos cujos refinamentos sejam necessários para a distinção dos eventos apontados na etapa de identificação, e por variáveis cujos valores são usados para atualizar as variáveis previamente identificadas.

Note que, para o exemplo proposto, a etapa de complementação não se aplica, já que não há interdependências em relação a outros refinamentos ou variáveis.

- Etapa de *modelagem*:

  O objetivo geral dessa etapa é modelar uma estrutura capaz de distinguir os refinamentos e de atualizar as variáveis, ambos apontados nos passos de identificação e complementação, conforme o contexto em que eles ocorrem no modelo do sistema.

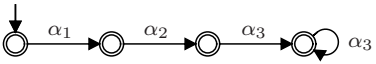
  No exemplo proposto, note que o efeito de associar, diretamente, um novo alfabeto $\Delta^{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\}$ e uma variável x, com $\mathrm{dom}(x) = \{0, 1, 2, 3\}$, ao modelo do sistema, leva a uma situação que pode ser informalmente associada à ideia de um não-determinismo. De fato, a ocorrência na planta de uma transição com o evento α passa agora a corresponder a uma diferente transição para cada evento refinado, e a variável x passa a ser atualizada, no estado alcançado, com qualquer valor dentro de seu domínio.

  É necessário, então, atribuir um significado (como foi conjecturado na etapa de estruturação) para cada elemento de Δ^{α} e de $\mathrm{dom}(x)$, conforme o contexto em que se dá sua ocorrência na planta do sistema. A carga semântica desses elementos é atribuída por um distinguidor e por fórmulas lógicas, respectivamente.

  No caso do distinguidor, o objetivo é ordenar a habilitação de $\alpha_i \in \Delta^{\alpha}$, conforme a ocorrência de α na planta. Equivalentemente, no caso do AFE, objetiva-se atribuir um valor a x de maneira a identificar, no estado alcançado pela transição com α, a real instância em que α ocorreu. Duas estruturas que implementam esses papéis são apresentadas na Figura 37.

  Em (a), o autômato H_{D} modela um distinguidor preditível que evita que mais de um refinamento, com a mesma máscara, seja simultaneamente habilitado na planta. Já a fórmula lógica em (b), a ser associada à transição em auto-laço com α, na planta, é claramente total, Q-determinística e V-determinística. Assim, a variável x é atualizada pela planta de maneira exata e sem impor nenhuma restrição ao sistema modelado.

(a) Distinguidor H_{D}:



(b) Fórmula lógica:

$x' = \min(x+1, 3)$

Figura 37 – Modelos para (a) a distinção dos eventos em Δ^{α} e para (b) a atualização de x.

A Figura 38 mostra as plantas G_{d} e G_{v}, resultantes da associação das estruturas (a) e (b), da Figura 37, às respectivas versões refinadas do modelo G, mostrado na Figura 35.

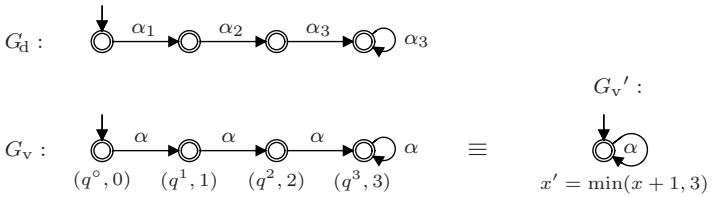


Figura 38 – Modelos da planta: com distinguidores (G_{d}) e com variáveis (G_{v} e $G_{\mathrm{v}}{}'$).

Os autômatos G_{v} e $G_{\mathrm{v}}{}'$ correspondem, respectivamente, às versões explícita e implícita da planta modelada por AFEs. Já G_{d} é uma planta refinada que distingue as ocorrências de α, conforme o modelo H_{D}.

Note que G_{d} e G_{v} diferem apenas na forma como carregam as informações acerca da ocorrência de α. O primeiro modelo carrega tais informações na forma de eventos refinados, enquanto o segundo, na forma de elementos de estados. Em termos de distinções, tanto o efeito do distinguidor quanto da formulação lógica, resulta em uma dinâmica similar na planta.

Finalmente, podem ser obtidas as soluções ótimas para o PCS, para o PCS-D e para o PCS-V, implementadas pelos modelos dos supervisores $\sup\mathcal{C}(K, G)$, $\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})$ e $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, respectivamente. A partir dos resultados apresentados nos capítulos 4 e 6, tem-se que

$$\sup\mathcal{C}(K, G) = \Pi(\sup\mathcal{C}(K_{\mathrm{d}}, G_{\mathrm{d}})) = L(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})).$$

7.1 UM EXEMPLO DE UM SISTEMA DE ESTOCAGEM

Dada a estrutura parcial do SED apresentado na Figura 34, considere complementá-la com novos subsistemas, de modo a compor um sistema de manufatura com estocagem, como mostrado na Figura 39. A modelagem apresentada a seguir ilustra, de maneira comparativa, as vantagens do uso de distinguidores e AFEs, combinados às respectivas técnicas de aproximações e de abstrações, na síntese de supervisores para SEDs.

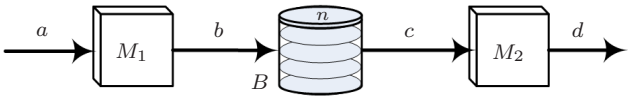


Figura 39 – Pequena fábrica com estocagem intermediária de materiais.

O sistema passa agora a ser composto por duas máquinas, M_1 e M_2, que manufaturam peças sequencialmente e são intermediadas por um *buffer*, B, de capacidade n. A Figura 40 apresenta os autômatos G^1 e G^2, modelando respectivamente M_1 e M_2.

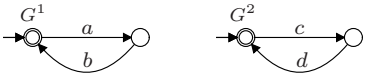


Figura 40 – Modelos para os subsistemas M_1 e M_2.

Os eventos a e c modelam o início de operação em M_1 e M_2, respectivamente, enquanto b e d modelam o final. O comportamento do sistema em malha aberta é dado pela composição $G = G^1 \| G^2$, com $\Sigma = \{a, b, c, d\}$, onde é assumido que $\Sigma_c = \{a, c\}$ e $\Sigma_u = \{b, d\}$.

O objetivo de controle, para esse exemplo, é impedir o *overflow* (E^O) e o *underflow* (E^U) em B. Modelar essas especificações requer explicitar a estrutura de estados que contempla a combinação completa de eventos de inserção e de remoção no *buffer*, desde o momento em que o *buffer* está vazio até que esteja cheio. Isso significa lidar com autômatos de $n + 1$ estados, como mostrados na Figura 41.

Ainda que a estruturação de tais $n+1$ estados possa ser possível, em termos de esforço de modelagem, esses autômatos são dependentes do fator n, ou seja, do tamanho do *buffer*. Adicionalmente, E^O e E^U não são diretamente modularizáveis. Isso implica que uma solução de

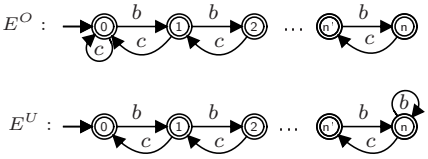


Figura 41 – Modelos para as especificações E^O e E^U.

controle, envolvendo esses modelos, pode ser complexa para *buffers* com alta capacidade de armazenamento. Na sequência, os distinguidores e os AFEs são introduzidos para simplificar esses modelos.

7.1.1 Construção do distinguidor e da lógica de variáveis

Aplicam-se, aqui, os passos definidos para a construção de um distinguidor e da estrutura lógica de um AFE, conforme apresentados nos capítulos 4 e 6 e retomados neste capítulo.

(i) modelar E^O e E^U requer saber, respectivamente, quando o *buffer* B está cheio e quando está vazio. Como B pode saturar após um evento b, e esvaziar após um evento c, então é em relação a esses dois eventos que devem ser providas as informações adicionais. Nesse sentido, são definidos:

- os refinamentos Δ^b e Δ^c, para identificar quando o evento b ocorre na última posição de B, e quando c ocorre na primeira posição;
- uma variável x, para identificar cada posição de B.

(ii) As estruturas dos conjuntos Δ^b, Δ^c e da variável x, são definidas como segue:

- $\Delta^b = \{b_1, \cdots, b_{n+1}\}$ e $\Delta^c = \{c_0, \cdots, c_n\}$;
- $\mathrm{dom}(x) = \{0, \cdots, n\}$, onde $x^\circ = 0$.

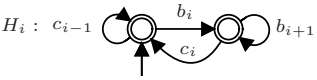
Semanticamente, i, para $i = 1, \cdots, n$, identifica a respectiva posição no *buffer*. Assim, o valor $x = i$ identifica o estágio corrente de ocupação de B, o evento b_i modela que *uma peça foi depositada na posição i de B*, e o evento c_i modela que *uma peça foi removida da posição i de B*.

Em decorrência dessas estruturas, o *overflow* e o *underflow* em B podem ser caracterizados, em cada uma das abordagens de modelagem, respectivamente por c_0 e $x = 0$, e por b_{n+1} e $x = n$.

(iii) As construções definidas acima são complementadas como segue:

- como os eventos a e d não precisam ser instanciados, então define-se $\Delta^a = \{a\}$ e $\Delta^d = \{d\}$ e, finalmente, $\Delta = \Delta^a \cup \Delta^b \cup \Delta^c \cup \Delta^d$;
- como uma única variável é suficiente para representar o *buffer*, então $V = \{x\}$. Alternativamente, poderia ser criada uma variável para cada posição ou, ainda, para estágios do *buffer*, com a vantagem de atribuir modularidade à abordagem. Nesse caso, a estrutura de atualização de variáveis (próximo item) seria particular.

(iv) Finalmente, a estrutura do distinguidor (a) e das fórmulas lógicas para a atualização de x (b) podem ser modeladas como na Figura 42.

(a) 

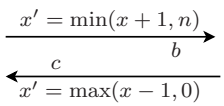
(b) 

Figura 42 – Distinguidor H_i, tal que $H_{\mathrm{D}} = \|_{i=1}^{n} H_i \| H_{\Delta}$, e fórmulas sobre x.

O autômato H_{D}, tal que $L(H_{\mathrm{D}}) = L_{\mathrm{D}}$, resulta da composição dos módulos distinguidores H_i. Esses são individualmente modelados usando-se apenas dois estados, independentemente do tamanho de n. Esse aspecto pode ser empregado para dimensionar o esforço de modelagem requerido na construção do distinguidor. Já o esforço para implementar as fórmulas em (b), está associado à percepção de que são as transições com os eventos b e c que representam as inserções e retiradas, respectivamente, de peças do *buffer*, e que, portanto, são elas que devem reger a atualização de x. A complexidade inerente à estruturação da fórmula lógica é praticamente irrelevante.

7.1.2 Modelagem da planta

A partir do alfabeto refinado e da variável x, pode-se reintroduzir os modelos G^1 e G^2 no contexto da resolução dos problemas PCS-D e

PCS-V, como mostrado na Figura 43.

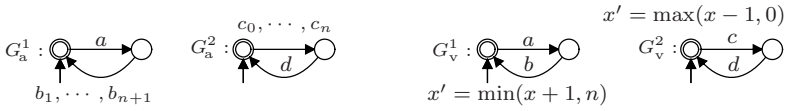


Figura 43 – Plantas G^1_{a} e G^2_{a}, modeladas em Δ, e AFEs G^1_{v} e G^2_{v}, atualizando x.

Os autômatos G^1_{a} e G^2_{a} são obtidos através da aplicação do mapa Π^{-1} sobre os modelos da planta em Σ, i.e., através da substituição de cada evento original pelo conjunto dos respectivos eventos refinados. Já os AFEs G^1_{v} e G^2_{v} são simplesmente as versões dos AFs G^1 e G^2, aderindo às estruturas lógicas em cada respectiva transição.

7.1.3 Modelagem das especificações

Também os modelos das especificações E^O e E^U podem ser reintroduzidos, agora nas versões simplificadas, como mostram os autômatos E^O_{d}, E^U_{d}, E^O_{v} e E^U_{v}, na Figura 44.

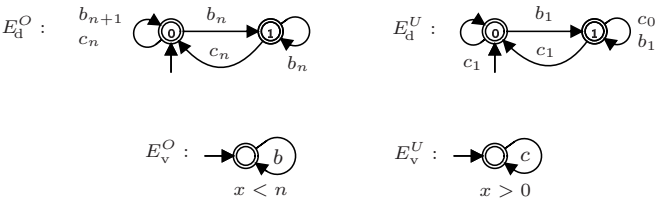


Figura 44 – Especificações modeladas em Δ e através de AFEs.

A especificação E^O_{d} evita um evento de inserção quando o *buffer* está cheio (evento b_{n+1} no estado 1), enquanto E^U_{d} previne uma remoção com o *buffer* vazio (evento c_0 no estado 0). Analogamente, E^O_{v} evita um evento de inserção no *buffer*, b, quando a variável x aponta para o topo do *buffer* (valor n), enquanto E^U_{v} previne uma remoção, evento c, quando $x = 0$, situação em que o *buffer* está vazio. Em quaisquer dos casos, a decisão de controle é tomada com base nos eventos refinados b_{n+1} e c_0, ou nos valores de x, sem, portanto, estruturar extensas sequências de eventos.

7.1.4 Síntese de controle

Obtêm-se, a seguir, as soluções para o PCS, o PCS-D e o PCS-V. Em cada contexto, a planta, a especificação e a especificação global, são dadas, respectivamente, por

$$\begin{array}{rcll} G = G^1 \parallel G^2 & , & E = E^O \parallel E^U & e \quad K = E \cap L^\omega(G); \\ G_\mathrm{a} = G^1_\mathrm{a} \parallel G^2_\mathrm{a} & , & E_\mathrm{d} = E^O_\mathrm{d} \parallel E^U_\mathrm{d} & e \quad K_\mathrm{a} = E_\mathrm{d} \cap L^\omega(G_\mathrm{a}); \ e \\ \exists V G_\mathrm{v} & , & E_\mathrm{v} = E^O_\mathrm{v} \mathbin{/\!\!/} E^U_\mathrm{v} & e \quad K_\mathrm{v} = E_\mathrm{v} \mathbin{/\!\!/} \exists V G_\mathrm{v}, \end{array}$$

tal que $L(G_\mathrm{a}) = \Pi^{-1}(L(G))$, $G_\mathrm{v} = G^1_\mathrm{v} \mathbin{/\!\!/} G^2_\mathrm{v}$ e E e E_d representam as linguagens dos respectivos autômatos $E^O \| E^U$ e $E^O_\mathrm{d} \| E^U_\mathrm{d}$.

A Tabela 12 apresenta o resultado da síntese dos supervisores V, V_a e V_v, modelando $\sup\mathcal{C}(K, G)$, $\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a})$ e $\sup\mathcal{C}_V(E_\mathrm{v}, \exists V G_\mathrm{v})$, respectivamente.

Tabela 12 – Número de estados dos autômatos usados no PCS, no PCS-D e no PCS-V.

G	E	K	V
4	$n+1$	$4n+4$	4n+2
G_a	E_d	K_a	V_a
4	4	16	12
$\exists V G_\mathrm{v}$	E_v	K_v	V_v
4	1	4	4

Considere um caso particular em que o *buffer* possui capacidade para estocar até $n = 1000$ itens. Nesse caso, resolver o PCS consiste em obter um autômato V que implementa $\sup\mathcal{C}(K, G)$, o qual é computado a partir de K, modelado com 4004 estados. Já no PCS-D, um supervisor V_a, que implementa $\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a})$, pode ser obtido a partir do modelo de K_a, com apenas 16 estados. Ainda assim, pode ser mostrado que, para esse caso particular, $\Pi(\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a}) \| L_\mathrm{D}) = \sup\mathcal{C}(K, G)$. Finalmente, na resolução do PCS-V, o supervisor $V_\mathrm{v} = \sup\mathcal{C}_V(E_\mathrm{v}, \exists V G_\mathrm{v})$ é computado a partir de apenas 4 estados e, equivalentemente, leva a uma solução tal que

$$L(\sup\mathcal{C}_V(E_\mathrm{v}, \exists V G_\mathrm{v})) = \Pi(\sup\mathcal{C}(K_\mathrm{a}, G_\mathrm{a}) \| L_\mathrm{D}) = \sup\mathcal{C}(K, G).$$

Nesse exemplo, os supervisores V_a e V_v foram computados a par-

tir de plantas construídas com as respectivas aproximação e abstração máximas, ou seja, sem envolver o modelo distinguidor H_{D} e sem atualizar a variável x. Ainda assim, tais plantas levaram diretamente ao resultado ótimo de síntese, de modo que o exemplo foi tratado com a máxima eficiência computacional.

7.2 SUMÁRIO

Neste capítulo, foram identificadas e comparadas as principais características dos distinguidores e dos AFEs. Inicialmente, foram relacionados os passos usados para a construção dessas duas estruturas; em seguida, cada um desses passos foi discutido no contexto de um exemplo parcial de um SED; após, o exemplo foi complementado e uma solução de controle foi obtida, o que permitiu evidenciar os principais benefícios de cada uma das abordagens, entre si e em relação à solução original.

8 CONCLUSÕES

Nesta tese foram propostos dois métodos para modificar o modelo de um SED, refinando-o conforme uma abordagem *top-down* em relação ao nível de abstração com que o sistema é visto: o uso de distinguidores e o uso de Autômatos Finitos Estendidos (AFEs). Também foi investigada a associação de distinguidores ao Controle Modular Local (CML). Comentários que sumarizam esses métodos são introduzidos a seguir.

O primeiro método proposto, fundamentou-se no conceito de distinguidores para prover um meio de simplificar a síntese de supervisores para SEDs. Foi mostrado que um distinguidor permite que complexas especificações de controle possam ser representadas de maneira mais simples preservando-se, ao mesmo tempo, a otimidade do supervisor resultante. O preço a se pagar por essa simplificação, é que o modelo da planta do sistema se torna mais complexo ao incorporar o modelo do distinguidor, de maneira que, em geral, não se observam vantagens computacionais a partir do uso direto desse método.

Nesse sentido, foi apresentado o conceito de D-Aproximação (ou simplesmente aproximação) para o modelo distinguido da planta de um SED. Esse conceito se baseia no fato de que, embora a distinção de eventos refinados seja necessária ao implementar o sistema de supervisão sobre um SED, sob certas condições, pode ser possível desconsiderar parcialmente essa distinção durante a síntese. Como resultado, o modelo aproximado de uma planta tende a ser mais simples, em termos de número de estados, do que a planta distinguida. Logo, também o esforço computacional, requerido para obter uma solução de controle, tende a ser reduzido, ao passo que as especificações compactas podem ainda ser utilizadas. Foram introduzidas condições sob as quais o uso de aproximações leva a resolver o problema de controle supervisório. No mesmo sentido, foi provida uma condição suficiente para testar a otimidade de tal solução. Para os casos em que ela não corresponde à minimamente restritiva, mostrou-se como o modelo do distinguidor pode ser gradualmente utilizado na construção de aproximações que levam ao resultado ótimo. A abordagem foi ilustrada através dos exemplos de um sistema de manufatura e de um sistema de estocagem.

O segundo método apresentado propôs o uso de AFEs na síntese de supervisores para SEDs. Os AFEs definem um formalismo de modelagem que, em geral, facilita a representação de especificações de controle e vem sendo usado na TCS para a síntese de supervisores óti-

mos. Nesta tese, o uso de AFEs foi explorado, também, do ponto de vista da redução do esforço computacional de síntese.

Foi introduzido o conceito de abstração de variáveis, um método estendido das técnicas de verificação formal, para a TCS. Esse método utiliza-se de quantificadores existenciais para abstrair um conjunto de variáveis em um AFE, durante a síntese, devolvendo essas variáveis na etapa de composição de uma solução implementável de controle. Foi mostrado que a abstração de variáveis permite obter importantes ganhos computacionais, mas, ao mesmo tempo, pode levar a uma solução subótima de controle. Por isso, foi apresentado um meio de informar se um resultado de síntese é o minimamente restritivo. Essa informação é provida por um algoritmo de síntese, também proposto na tese, que, ao computar um supervisor, reúne informações sobre certas variáveis que foram abstraídas do modelo do sistema. Isso lhe confere o poder de apontar para variáveis que devem compor a síntese em face à solução ótima. Portanto, esse algoritmo também foi introduzido como um mecanismo através do qual uma classe de abstrações pode ser construída, a qual sempre leva à síntese ótima.

A abordagem com AFEs foi ilustrada no contexto dos exemplos de um sistema de manufatura e de um sistema de estocagem. Já o algoritmo de síntese foi ilustrado através de um exemplo adicional, que permitiu ilustrar o procedimento de síntese em detalhes.

A associação de distinguidores ao CML foi motivada pelo fato de que, embora a modularização de um problema de controle, em geral, simplifique o procedimento de síntese, ela não altera a forma como as especificações são modeladas.

Nesse sentido, a incorporação de distinguidores ao CML mostrou-se útil por possibilitar que os aspectos de modelagem sejam tratados de forma mais simples. Porém, também foi mostrado que essa abordagem é computacionalmente mais custosa, se comparada ao CML sem distinguidores.

Então, foi apresentado um método de síntese que combina a resolução local de módulos do problema de controle, com e sem distinguidores, sempre que conveniente. Essa abordagem equilibra a relação entre custo computacional e benefícios de modelagem, porém, ainda sem vantagens na síntese. Nesse sentido, a abordagem combinada foi estendida para envolver, também, o uso de aproximações locais, o que permitiu obter importantes ganhos computacionais, sem alterar o comportamento global controlado.

8.1 APLICABILIDADE E LIMITAÇÕES DA TESE

Os dois métodos de síntese propostos nesta tese, com distinguidores e com AFEs, são aplicáveis a problemas de controle envolvendo SEDs. Em particular, os métodos se justificam perante uma classe de problemas que naturalmente impõem um elevado grau de complexidade de modelagem. Em certos casos, essa complexidade pode ser reduzida ao refinar o modelo do sistema de forma a prover mais informações acerca da ocorrência de certos eventos. Quando assim for, pode-se considerar que os métodos propostos são aplicáveis à resolução do problema.

Em relação ao CML, uma das limitações desta tese se observa ao não explicitar como se dá sua relação com os AFEs, embora essa associação possa ser pensada de maneira análoga ao CML com distinguidores.

8.2 DESCRIÇÃO DOS RESULTADOS ALCANÇADOS

A produção técnica, vinculada a esta tese, iniciou-se no ano de 2011 com a publicação de um artigo científico em conferência internacional (TEIXEIRA; CURY; QUEIROZ, 2011) descrevendo os primeiros resultados sobre a associação de distinguidores ao CML e introduzindo a abordagem combinada de síntese. Paralelamente, foi produzido um segundo artigo (CURY et al., 2012), contemplando a generalização e importantes melhorias na fundamentação teórica da abordagem de síntese com distinguidores. Esse artigo foi submetido no ano de 2012 ao periódico *IFAC Automatica*.

No ano de 2012, a pesquisa de tese foi vinculada ao contexto de uma cooperação externa, decorrente do período de *Doutorado Sanduíche*, cumprido por 9 meses junto à *The University of Waikato*, na Nova Zelândia. Durante esse período, iniciou-se a investigação do uso de AFEs na TCS. Dois artigos científicos configuram, essencialmente, os resultados obtidos nesses 9 meses de pesquisa. O primeiro foi submetido em 2012 e publicado em 2013 (TEIXEIRA et al., 2013), em conferência internacional. Um segundo artigo, contemplando resultados complementares sobre a abordagem com AFEs, foi submetido em 2013 ao periódico *IEEE Transactions on Automatic Control*.

Ainda em 2013, um artigo científico complementar, que estende os resultados sobre CML e distinguidores para envolver também aproximações, foi publicado em conferência internacional (TEIXEIRA; CURY;

QUEIROZ, 2013).

8.3 FERRAMENTAS UTILIZADAS

As atividades de modelagem e síntese de supervisores, conduzidas nesta tese, foram suportadas pela ferramenta *Supremica* (AKESSON et al., 2013). A corretude dos resultados foram checados em outras ferramentas de síntese, como o *Ides* (RUDIE et al., 2013). A editoração da tese foi conduzida através do processador de textos *LaTeX* (LATEX, 2013).

O padrão gráfico da Supremica foi adotado para ilustrar os modelos na tese. Um dos diferenciais da ferramenta, considerados para a escolha, foi a possibilidade de integrar as operações de síntese, a editoração gráfica dos modelos e o processamento textual.

8.4 PROJETO DE TRABALHOS FUTUROS

A continuidade desta tese está vinculada a diversas vertentes, dentre as quais, citam-se:

- *Implementação*: a ferramenta Supremica suporta apenas parcialmente as operações envolvendo AFEs e a síntese com abstração de variáveis, de modo que tais implementações são planejadas como trabalhos futuros. Ademais, a computação de supervisores Λ-*Preservantes*, bem como outras operações sobre modelos refinados (como os mapas Π e Π^{-1} (TEIXEIRA, 2013)), devem ser integradas ao mesmo ambiente.

- *Modularização da síntese com AFEs*: o objetivo, aqui, é preparar a estrutura e mostrar resultados que permitam associar os AFEs e o CML, tal como feito com distinguidores; adicionalmente, o uso de AFEs pode ser estendido ao contexto de outras arquiteturas modulares, como a síntese composicional (MOHAJERANI et al., 2011).

- *Síntese com garantia de não-conflito*: Supervisores obtidos a partir de aproximações / abstrações, requerem ser associados à linguagens exatas, a fim de comporem soluções implementáveis de controle. Essa composição requer um teste de conflito, o que, em geral, é computacionalmente caro. Sob certas condições (possivelmente derivadas das propriedades das aproximações / abstrações)

pode ser possível garantir, por construção, que uma solução de controle seja não-conflitante, descaracterizando a necessidade do teste.

- *Relação entre distinguidores e AFEs*: o objetivo é formalizar os passos da construção dessas estruturas, para que se possa, então, explorar a síntese desses modelos.

- *Síntese de distinguidores e AFEs*: A construção de distinguidores e de AFEs está vinculada, sobretudo, à habilidade técnica de um projetista. Sob certas condições, contudo, pode ser possível estruturar a síntese automática desses modelos.

- *Exploração de outros tipos de variáveis*: No contexto dos AFEs, a modelagem de especificações se dá através de fórmulas lógicas implementadas sobre variáveis. Na prática, uma variável pode estar vinculada ao contexto de diversas estruturas computacionais, muitas delas não exploradas atualmente, como é o caso dos *objetos*. A ideia de definir um objeto é justamente a de aumentar o poder de expressão semântica em modelos. Logo, é intuitivo pensar que objetos podem ser úteis na TCS de SEDs, em certos casos.

REFERÊNCIAS

AGUIAR, R. S. S. *Busca Heurística na obtenção de supervisores aproximados por distinguidores: estudo de caso na coordenação de sistemas multiveiculares*. Dissertação (Mestrado) — Instituto Militar de Engenharia (IME), 2013.

AKESSON, K. et al. *Supremica*. [S.l.], 2013. <http://www.supremica.org/>.

BÉRARD, B. et al. *Systems and Software Verification*. 2001. ed. Berlin: Springer Verlag, 2001.

BOLCH, G. et al. *Queueing Networks and Markov Chains. Modeling and Performance Evaluation with Computer Science Applications*. 2. ed. New Jersey: John Wiley & Sons, Inc., 2006.

BOUZON, G.; QUEIROZ, M. H. de; CURY, J. E. R. Supervisory control of des with distinguishing sensors. In: IFAC. *International Workshop on Discrete Event Systems (WODES'08)*. Gothenburg, Sweden, 2008. p. 22–27.

BOUZON, G.; QUEIROZ, M. H. de; CURY, J. E. R. Exploiting distinguishing sensors in supervisory control of des. In: IEEE. *International Conference on Control and Automation (ICCA'09)*. Christchurch, New Zealand, 2009. p. 442–447.

BRANDIN, B. A.; WONHAM, W. M. Modular supervisory control of timed discrete-event systems. In: IEEE. *Annual Conference on Decision and Control (CDC'93)*. San Antonio, USA, 1993. p. 2230–2235.

CAI, K.; WONHAM, W. M. Supervisor localization: A top-down approach to distributed control of discrete-event systems. *IEEE Transactions Automatic Control*, v. 55, n. 3, p. 605–618, 2010.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. 2. ed. New York: Springer Science, 2008.

CHEN, Y.; LIN, F. Modeling of discrete event systems using finite state machines with parameters. In: IEEE. *International Conference on Control Applications (ICCA'00)*. Anchorage, USA, 2000. p. 941–946.

CHEN, Y. L.; LAFORTUNE, S.; LIN, F. Modular supervisory control with priorities for discrete event systems. In: IEEE. *34th Annual Conference on Decision and Control (CDC'95)*. New Orleans, USA, 1995. p. 409–415.

CIESLAK, R. et al. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions Automatic Control*, v. 33, n. 3, p. 249–260, 1988.

CUNHA, A. E. C. da; CURY, J. E. R. Hierarchical supervisory control based on discrete event systems with flexible marking. *IEEE Transactions Automatic Control*, v. 52, p. 2242–2253, Dec. 2007.

CURY, J. E. R. *Teoria de Controle Supervisório de Sistemas a Eventos Discretos*. Canela, RS, 2001. Apostila.

CURY, J. E. R. et al. *Supervisory Control of Discrete Event Systems with Distinguishers*. 2012. Submetido ao periódico IFAC Automatica.

DARONDEAU, P. Distributed implementations of ramadge-wonham supervisory control with petri nets. In: IEEE. *Conference on Decision and Control and European Control Conference (CDC'05-ECC'05)*. Seville, Spain, 2005. p. 2107–2112.

DESROCHERS, A. A.; AL-JAAR, R. Y. *Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis*. [S.l.]: IEEE, 1994.

FENG, L.; WONHAM, W. M. Supervisory control architecture for discrete-event systems. *IEEE Transactions Automatic Control*, v. 53, n. 6, p. 1449–1461, 2008.

FLORDAL, H.; MALIK, R. Compositional verification in supervisory control. *SIAM Journal on Control and Optimization*, v. 48, n. 3, p. 1914–1938, 2009.

GIUA, A.; DICESARE, F. Blocking and controllability of petri nets in supervisory control. *IEEE Transactions on Automatic Control*, v. 39, n. 4, p. 818–823, 1994.

HAJI-VALIZADEH, A.; LOPARO, K. A. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions Automatic Control*, v. 41, n. 11, p. 1579–1593, 1996.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. *Introduction to Automata Theory, Languages and Computation*. 2. ed. [S.l.]: Addison Wesley, 2001.

HUANG, Y.; RUDIE, K.; LIN, F. Decentralized control of discrete-event systems when supervisors observe particular event occurrences. *IEEE Transactions Automation Control*, v. 53, n. 1, p. 384–388, 2008.

JIANG, S.; KUMAR, R. Supervisory control of discrete event systems with ctl* temporal logic specifications. *SIAM Journal of Control and Optimization*, v. 33, p. 419–439, 2006.

KHULLER, S.; KORTSARZ, G.; ROHLOFF, K. Approximating the minimal sensor selection for supervisory control. In: IFAC. *7th International Workshop on Discrete Event Systems (WODES'04)*. Reims, France, 2004. p. 143–170.

KUMAR, R.; GARG, V. K. On computation of state avoidance control for infinite state systems in assignment program framework. *IEEE Transactions Automation Science and Engineering*, v. 2, n. 1, p. 87–91, 2005.

LATEX. *Latex - A document preparation system*. [S.l.], 2013. <http://www.latex-project.org/>.

LEGALL, T.; JEANNET, B.; MARCHAND, H. Supervisory control of infinite symbolic systems using abstract interpretation. In: IEEE. *Conference on Decision and Control and European Control Conference (CDC'05-ECC'05)*. Seville, Spain, 2005. p. 30–35.

LIN, F.; WONHAM, W. M. On observability of discrete-event systems. *Information Sciences*, v. 44, n. 3, p. 173–198, 1988.

LIN, F.; WONHAM, W. M. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions Automatic Control*, v. 35, n. 12, p. 1330–1337, 1990.

MALIK, R.; FABIAN, M.; ÅKESSON, K. Modelling large-scale discrete-event systems using modules, aliases, and extended finite-state automata. In: *18th IFAC World Congress (IFAC'11)*. Milan, Italy: [s.n.], 2011. p. 7000–7005.

MOHAJERANI, S. et al. Compositional synthesis of discrete event systems using synthesis abstraction. In: IEEE. *Chinese Control*

and Decision Conference (CCDC'11). Mianyang, China, 2011. p. 1549–1554.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, 1989.

OLIVEIRA, C.; CURY, J.; KAESTNER, C. Discrete event systems with guards. In: IFAC. *11th Symposium on Information Control Problems in Manufacturing (INCOM'04)*. Salvador, Brazil, 2004. p. 90–95.

OUEDRAOGO, L. et al. Nonblocking and safe control of discrete-event systems modeled as extended finite automata. *IEEE Transactions on Automation Science and Engineering*, v. 8, n. 3, p. 560–569, 2011.

PENA, P. N.; CURY, J. E. R.; LAFORTUNE, S. Verification of nonconflict of supervisors using abstractions. *IEEE Transactions on Automatic Control*, v. 54, n. 12, p. 2803–2815, 2009.

PNUELI, A. The temporal logic of programs. In: IEEE. *18th Annual Symposium on Foundations of Computer Science (FOCS'97)*. Miami Beach, USA, 1977. p. 46–57.

QUEIROZ, M. H. D.; CURY, J. E. R. Modular control of composed systems. In: *American Control Conference (ACC'00)*. Illinois, USA: [s.n.], 2000. p. 4051–4055.

QUEIROZ, M. H. D.; CURY, J. E. R. Modular supervisory control of large scale discrete event systems. In: IFAC. *International Workshop on Discrete Event Systems (Wodes'00)*. Ghent, Belgium, 2000. p. 103–110.

RAMADGE, P. J. Some tractable supervisory control problems for discrete-event systems modeled by buchi automata. *IEEE Transactions on Automatic Control*, v. 34, n. 1, p. 10–19, 1989.

RAMADGE, P. J.; WONHAM, W. M. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, v. 25, n. 5, p. 1202–1218, 1987.

RAMADGE, P. J.; WONHAM, W. M. Supervisory control of a class of discrete event process. *SIAM Journal of Control and Optimization*, v. 25, n. 1, p. 206–230, 1987.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. *IEEE Special Issue on Discrete Event Dynamic Systems*, v. 77, n. 1, p. 81–98, 1989.

ROSEN, K. H. *Discrete Mathematics and Its Applications*. 6. ed. [S.l.]: McGraw-Hill, 2007.

RUDIE, K. et al. *Ides - Integrated Discrete-Event Systems*. [S.l.], 2013. <https://qshare.queensu.ca/Users01/rudie/www/software.html>.

RUDIE, K.; WONHAM, W. M. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions Automatic Control*, v. 37, n. 11, p. 1692–1708, 1992.

SCHMIDT, K.; MOOR, T.; PERK, S. Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Transactions Automatic Control*, v. 53, n. 10, p. 2252–2265, 2008.

SCHUTTER, B. D.; BOOM, T. van den. Max-plus algebra and max-plus linear discrete event systems: An introduction. In: IFAC. *9th International Workshop on Discrete Event Systems (WODES'08)*. Göteborg, Sweden, 2008. p. 36–42.

SHOAEI, M. R.; FENG, L.; LENNARTSON, B. Abstractions for nonblocking supervisory control of extended finite automata. In: IEEE. *International Conference on Automation Science and Engineering (CASE'12)*. Seoul, Korea, 2012. p. 364 – 370.

SKÖLDSTAM, M.; ÅKESSON, K.; FABIAN, M. Modeling of discrete event systems using finite automata with variables. In: IEEE. *46th Annual Conference on Decision and Control (CDC'07)*. New Orleans, USA, 2007. p. 3387–3392.

SPACEK, P. et al. Control of nonautonomous discrete event systems using dioid algebra. In: IEEE. *International Conference on Robotics and Automation (ICRA'96)*. St. Paul, USA, 1996. p. 609–615.

TEIXEIRA, M. *Refiner - an IDES plugin*. [S.l.], 2013. <http://www.das.ufsc.br/ mt/html/Downloads.html>.

TEIXEIRA, M.; CURY, J. E. R.; QUEIROZ, M. H. de. Local modular supervisory control of des with distinguishers. In: IEEE. *International Conference on Emerging Technologies and Factory Automation (ETFA'11)*. Toulouse, France, 2011. p. 1–8.

TEIXEIRA, M.; CURY, J. E. R.; QUEIROZ, M. H. de. Local modular control with distinguishers applied to a manufacturing system. In: IFAC. *2013 International Conference on Manufacturing Modelling, Management and Control (MIM'13)*. Saint Petersburg, Russia, 2013.

TEIXEIRA, M. et al. Variable abstraction and approximations in supervisory control synthesis. In: *2013 American Control Conference (ACC'13)*. Washington, USA: [s.n.], 2013. p. 120–125.

TRIVEDI, K. S. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. 2. ed. New York: John Wiley and Sons, 2002.

WONG, K. C.; WONHAM, W. M. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, v. 6, n. 3, p. 241–273, 1996.

WONG, K. C.; WONHAM, W. M. Modular control and coordination of discrete event systems. *Discrete Event Dynamic Systems*, v. 8, n. 3, p. 241–273, 1998.

WONHAM, W. *Notes on Discrete Event Systems*. 2002. University of Toronto.

WONHAM, W.; RAMADGE, P. Modular supervisory control of discrete-event systems. *Mathematics of Control, Signals, and Systems (MCSS)*, Springer London, v. 1, p. 13–30, 1988.

YOO, T.; LAFORTUNE, S. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, v. 12, n. 3, p. 335–377, 2002.

YOO, T.; LAFORTUNE, S. NP-completeness of sensor selection problems arising in partially-observed discrete-event systems. *IEEE Transactions Automatic Control*, v. 47, n. 9, p. 1495–1499, 2002.

ZHONG, H.; WONHAM, W. M. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions Automatic Control*, v. 35, n. 10, p. 1125–1134, 1990.