DISSERTAÇÃO DE MESTRADO Nº 1039

ABSTRAÇÕES DE SUPERVISORES LOCALMENTE MODULARES PARA APLICAÇÃO NA SOLUÇÃO DE PROBLEMAS DE PLANEJAMENTO

Michel Rodrigo das Chagas Alves

DATA DA DEFESA: 23/02/2018

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

ABSTRAÇÕES DE SUPERVISORES LOCALMENTE MODULARES PARA APLICAÇÃO NA SOLUÇÃO DE PROBLEMAS DE PLANEJAMENTO

Michel Rodrigo das Chagas Alves

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientadora: Profa. Patrícia Nascimento Pena

Belo Horizonte - MG

Fevereiro de 2018

Alves, Michel Rodrigo das Chagas.
A474a Abstrações de supervisores localmente modulares para aplicação na solução de problemas de planejamento [manuscrito] / Michel Rodrigo das Chagas Alves. - 2018.
97 f., enc.: il.

Orientadora: Patrícia Nascimento Pena.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 93-97.

Bibliografia: f. 87-91.

1. Engenharia elétrica - Teses. 2. Planejamento - Teses. 3. Abstração - Teses. I. Pena, Patrícia Nascimento. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Dedico esse trabalho a meus pais

Agradecimentos

Aos meus pais, Celeste e João, e minha irmã Aline pelo amor, incentivo e apoio incondicional. À minha orientadora, Professora Patrícia, pela orientação, apoio e confiança. Aos amigos, companheiros das horas difíceis e de alegria, que fizeram parte dessa etapa. À CAPES, que investiu em minha educação e à todos que direta ou indiretamente fizeram parte dessa trajetória, a minha sincera gratidão.

“La liberté commence où l’ignorance finit.”
(Victor Hugo)

Resumo

Uma abordagem para solução do problema de planejamento em sistemas de manufatura é a integração de métodos de otimização com a Teoria de Controle Supervisório. Essa teoria permite restringir o espaço de busca somente às soluções que respeitam a segurança e o não bloqueio do sistema. Se, no lugar deste comportamento, usa-se uma abstração do comportamento do sistema sob controle, é possível reduzir ainda mais o espaço de busca da otimização. Este trabalho estende resultado anterior, pela utilização de abstrações de supervisores obtidos pela síntese modular local, no lugar do supervisor monolítico. As abstrações consistem na projeção natural, que têm a propriedade do observador, dos supervisores modulares locais para o alfabeto de eventos controláveis. O resultado principal garante que qualquer cadeia obtida a partir da abstração será executável na sistema físico, com a presença dos eventos não controláveis. Como a síntese monolítica sofre com o problema de explosão de estados, a extensão torna viável o uso das abstrações mesmo quando não é possível tratar o problema monolítico. Além disso, dados experimentais mostram que a abstração reduz o espaço de busca em no mínimo 10 vezes, se comparado ao supervisor original.

Palavras-chaves: Sistemas a Eventos Discretos. Planejamento. Controle Supervisório. Propriedade do Observador. Abstrações.

Abstract

An approach to solving a scheduling problem in manufacturing systems is the integration of optimization methods with Supervisory Control Theory. This theory allows to restrict the search space to include solutions that respect the safety and nonblockingness of the system. If, instead of this behavior, an abstraction of the behavior of the system under control is used, it is possible to further reduce the optimization search space. This work extends a previous result, by the use of abstractions on supervisors obtained by the local modular synthesis, rather than the monolithic supervisor. The abstractions consist in the natural projection, which has the observer property, of the local modular supervisors to the set of controllable events. The main result guarantees that any string obtained from the abstractions will execute in the phisical system, in the presence of the uncontrollable events. As the monolithic synthesis suffers from the problem of computational explosion, the extension makes feasible the use of abstractions even when it is not possible to deal with the monolithic problem. Besides, experimental data show that the abstraction reduces the search space by a factor of 10, at least, when comparing to the original supervisor.

Key-words: Discrete Event Systems. Planning. Supervisory Control. Observer Property. Abstractions.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TCS	Teoria de Controle Supervisório
VNS	*Variable Neighborhood Search*
SED	Sistema a Eventos Discretos
RSP	Representação por Sistema Produto

Lista de símbolos

$\in$	pertence à
!	fatorial
$\geq$	maior ou igual (os operandos são números reais)
Σ	conjunto de eventos ou alfabeto
ε	cadeia vazia
$\|s\|$	comprimento da cadeia s
$=$	igual
$:=$	definição
uv	concatenação de cadeias
$s \leq t$	s é prefixo de t (os operandos são cadeias)
Σ^*	conjunto de todas as cadeias possíveis de serem formadas com elementos de Σ
$\overline{L}$	prefixo fechamento da linguagem L
$\subseteq$	contido em ou igual a
L^*	fechamento Kleene da linguagem L
$\subset$	contido em
P	projeção natural
$2^{\Sigma_M^*}$	conjunto de todos os subconjuntos de Σ_M^*
P^{-1}	projeção inversa
e, α, β, σ	eventos
$\setminus$	diferença de conjuntos
$\exists$	existe
$\cup$	união
$\cap$	interseção
$\forall$	para todo
$\Longrightarrow$	implica
$\|\|$	operador de composição paralela

G	autômato
Q	conjunto de estados do autômato
δ	função de transição do autômato
q_0	estado inicial do autômato
Q_m	conjunto de estados marcados do autômato
$\emptyset$	conjunto vazio
$\mathcal{L}(G)$	linguagem gerada pelo autômato G
$\mathcal{L}_m(G)$	linguagem marcada pelo autômato G
$\neq$	diferente
$Ac(G)$	parte acessível do autômato G
$CoAc(G)$	parte coacessível do autômato G
$Trim(G)$	parte acessível e coacessível do autômato G
$\supset$	contém
$\times$	produto de autômatos (os operandos são autômatos)
$\times$	produto cartesiano de conjuntos (os operandos são conjuntos)
$\wedge$	operador lógico E
$\notin$	não pertence
Σ_c	alfabeto de eventos controláveis
Σ_u	alfabeto de eventos não controláveis
S	autômato que implementa o supervisor
K	linguagem desejada
$Sup\mathcal{C}(K, G)$	operação que resulta na máxima sublinguagem controlável de K em relação à G
A	linguagem recuperada
$\Pi(s)$	operador de permutação dos eventos de uma cadeia s
S_{red}	supervisor reduzido

Sumário

1 Introdução

A produção em massa de bens possibilitou a redução do custo, aumento de qualidade, quantidade e variedade dos produtos oferecidos aos consumidores. Isso é possível graças ao emprego de máquinas e sistemas de produção automatizados para a elaboração de produtos e componentes, que geralmente são padrões e intercambiáveis (HU, 2013).

Nos últimos anos, no entanto, nota-se uma nova tendência no mercado: os consumidores procuram cada vez mais por produtos e serviços customizados, que atendam exatamente suas necessidades. O desafio das empresas é o de combinar as características da produção em massa com a necessidade de processar cada pedido customizado de forma eficiente, de modo a não reduzir a lucratividade em favor da customização (HU, 2013).

Com esse objetivo, os processos industriais têm se tornado cada vez mais flexíveis e, consequentemente, mais complexos, podendo se adaptar às novas demandas. Para manter a competitividade as empresas devem usar seus recursos de maneira eficiente. O problema de planejamento surge nesse contexto: como utilizar os recursos de maneira mais eficiente possível para a produção de uma determinada quantidade de produtos? Se antes uma sequência de produção era adequada para a produção de um determinado produto, ela também será para a produção de diferentes produtos? As respostas para essas perguntas não são simples de serem encontradas. Diversos trabalhos já abordaram o assunto mas nenhum deles trouxe as respostas definitivas (ARISHA *et al.*, 2001; BAKER; TRIETSCH, 2009; HILL; LAFORTUNE, 2016).

Portanto, é válido o interesse pela busca de técnicas para a solução do problema de planejamento que sejam mais genéricas, mais eficientes em obter uma solução ótima e que necessitem cada vez menos de recursos computacionais. Essa última característica é importante porque os processos têm se tornado cada vez mais complexos e flexíveis, o que torna e necessidade de planejamento mais frequente.

Dentre as técnicas que atendem esses requisitos, destacam-se aquelas que utilizam a modelagem dos sistemas como Sistemas a Eventos Discretos aliado à utilização da Teoria de Controle Supervisório. Essa última, com a figura do supervisor monolítico, traz uma redução significativa do espaço de busca pela solução ótima. Uma redução ainda maior deste espaço busca pode ser obtido com abstrações do supervisor monolítico, que já foram propostas em trabalhos anteriores. No entanto, a aplicação dessa abstração pode não ser possível devido à

dificuldade em se obter o supervisor monolítico.

Apoiando-se na ideia de dividir o problema maior em diversos subproblemas menores, este trabalho estende um resultado anterior e apresenta uma abstração e as condições para sua aplicação, que pode ser utilizada sobre supervisores obtidos pela síntese modular local. A abstração consiste na projeção natural, com a propriedade do observador, de cada um dos supervisores para o respectivo alfabeto de eventos controláveis.

Este trabalho é organizado da seguinte maneira: o Capítulo 2 apresenta uma revisão da bibliográfica sobre o problema de planejamento e de algumas técnicas aplicadas para a solução do mesmo, que incluem também a utilização da Teoria de Controle Supervisório. Adicionalmente, são apresentados algumas utilizações de abstrações em diferentes contextos. O Capítulo 3 traz os conceitos que fundamentam a modelagem por linguagens e autômatos, além de apresentar sucintamente a Teoria de Controle Supervisório e o Controle Modular Local. O Capítulo 4 reúne extratos de outros trabalhos que foram utilizados como base para a elaboração da contribuição deste trabalho, que é apresentada no Capítulo 5. Além disso, no Capítulo 5 é apresentado um exemplo de aplicação da abstração proposta. Finalmente, o Capítulo 6 traz as conclusões deste trabalho.

2 Revisão de Literatura

Este capítulo apresenta uma revisão de literatura acerca dos temas abordados neste trabalho, como o problema de planejamento, controle modular local e controle hierárquico.

2.1 O problema de planejamento

O planejamento é um termo que descreve um processo que acontece regularmente na indústria e na vida quotidiana das pessoas. É um processo de deliberação explícita que escolhe e organiza ações pela antecipação de seus resultados esperados. As escolhas têm o propósito de alcançar da melhor maneira possível determinados objetivos (GHALLAB *et al.*, 2004). Uma pessoa, por exemplo, tendo que visitar vários lugares em um dia, pode planejar, ou decidir, a ordem em que as visitas serão feitas, de maneira a minimizar a distância percorrida. Num ambiente industrial, muitas vezes é necessário escolher a ordem de produção de determinados produtos de modo a reduzir o tempo ou o custo de produção.

Embora as pessoas possam realizar o processo de planejamento de forma intuitiva e muitas vezes inconscientemente, em ambientes industriais essa tarefa torna-se mais complexa, em razão do grande número de variáveis envolvidas. Isso justifica o uso de ferramentas e métodos formais para realização do planejamento, bem como termos específicos para caracterizar e descrever os sistemas. Tipicamente, aplica-se o processo de planejamento em sistemas de manufatura compostos de M máquinas diferentes, onde N tarefas devem ser realizadas. Cada tarefa J_n, com $n \in \{1, ..., N\}$ pode ser considerada a transformação de um insumo bruto em um produto acabado. A completude de uma tarefa pode depender do processamento em diferentes máquinas M_m, com $m \in \{1, .., M\}$, numa ordem específica, determinada por características tecnológicas. O processamento da tarefa J_n na máquina M_m é chamado de operação O_{nm}. A operação O_{nm} requer o uso exclusivo da máquina M_m por um tempo de processamento ininterrupto p_{nm} (YAMADA; NAKANO, 1997). O tempo decorrido para a realização de todas as tarefas é chamado de *makespan*.

Formalmente, o problema de planejamento, também conhecido como escalonamento, consiste em determinar a sequência das operações O_{nm} de forma a minimizar uma determinada função dos tempos de finalização (ARISHA *et al.*, 2001). Habitualmente, deseja-se minimizar o *makespan*. A solução para o problema de planejamento é o conjunto de tempos de finalização de cada operação O_{nm} que atende às restrições. A Tabela 1 apresenta um

exemplo de um problema com 3 máquinas e 3 tarefas.

Tabela 1 – Exemplo de um problema 3×3

Tarefa	Sequência de operação (Tempo de processamento)		
J_1	M_1 (3)	M_2 (3)	M_3 (3)
J_2	M_1 (2)	M_3 (3)	M_2 (4)
J_3	M_2 (3)	M_1 (2)	M_3 (1)

Uma forma conveniente de visualizar o planejamento de um sistema de manufatura é através de um diagrama de Gantt, como o mostrado na Figura 1, que ilustra uma possível solução para o problema apresentado na Tabela 1.

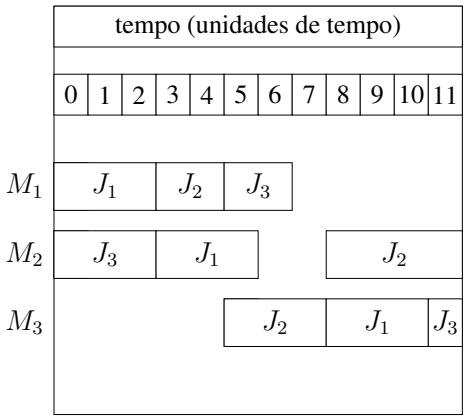


Figura 1 – Gráfico de Gantt de uma solução para um problema 3x3.

Adaptado de Yamada e Nakano (1997)

O problema de planejamento pode ser visto como um problema de otimização. A forma mais simples de efetuar a busca pela solução é através da realização de permutações na ordem de operações das máquinas. Uma solução é válida, ou factível, se nenhuma das restrições foi violada. No entanto, o número de soluções possíveis, contabilizando as que são factíveis ou não, é dado por $(N!)^M$ (PINEDO, 2008). O crescimento mais que exponencial do número de soluções é a principal dificuldade na resolução dos problemas de planejamento, sendo o mesmo classificado como *NP-hard*. Desde o seu surgimento diversos trabalhos abordaram o desenvolvimento de técnicas para a solução do problema, mas nenhuma trouxe uma solução definitiva.

Em Ghallab *et al.* (2004) é apresentado um modelo conceitual para a solução do problema de planejamento. O modelo é formado pela interação de três componentes: o Planejador, o Controlador e o Sistema Físico, como pode ser visto na Figura 2. O Planejador é responsável por encontrar um plano ótimo em relação a um determinado objetivo. Para

encontrar essa solução, o Planejador deve contar internamente com um otimizador e um avaliador. O primeiro é responsável pela geração de sequências candidatas ao plano ótimo, enquanto o segundo avalia essas sequências e atribui à elas uma nota. A sequência de melhor nota obtida pelo otimizador é considerada então o plano ótimo. Esse é então passado ao Controlador, que por sua vez, acompanha a evolução do Sistema Físico observando os eventos e atuando sobre o mesmo com ações que estão de acordo com o plano.

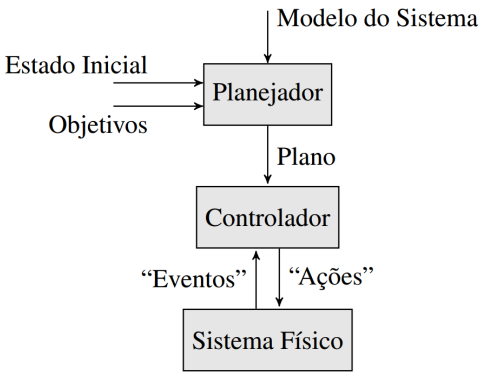


Figura 2 – Modelo conceitual para o processo de planejamento

Adaptado de Ghallab *et al.* (2004)

Diversos métodos podem ser empregados para que o Planejador obtenha o plano ótimo. Dentre as técnicas consideradas como tradicionais (ARISHA *et al.*, 2001), há dois grandes grupos: as técnicas analíticas e as heurísticas.

As técnicas analíticas têm a abordagem de considerar o problema como um todo, o que muitas vezes é inviável. A enumeração explícita é uma técnica em que todas as possíveis soluções são consideradas. A limitação é o tamanho do espaço de busca, que cresce com o exponencial do fatorial do tamanho do problema. A enumeração implícita, ou programação dinâmica, considera determinadas soluções sem explicitamente avaliá-las, o que torna a técnica mais eficiente. No entanto, o esforço computacional ainda cresce exponencialmente com o tamanho do problema, o que é um fator limitante para sua aplicação (BAKER; TRIETSCH, 2009).

A técnica conhecida como *branch and bound* consiste em dois procedimentos básicos. O primeiro é o particionamento (ou *branching*) de um problema maior em dois ou mais subproblemas. O segundo é o processo de calcular um limite inferior (*bound*) para a solução ótima de cada subproblema. O procedimento de *branching* substitui o problema original por um conjunto de novos problemas que são i) mutualmente exclusivos, ii) versões do original que são parcialmente resolvidas e iii) menores problemas que o original (BAKER; TRIETSCH, 2009).

Métodos de programação inteira mista também já foram aplicados para a minimização de *makespan* (KU; BECK, 2016) e para minimização de tarefas que terminam antes ou depois de determinado tempo, termos conhecidos como *earliness* e *tardiness*, respectivamente (RONCONI; BIRGIN, 2012). O problema é modelado por equações em que algumas variáveis só podem assumir valores inteiros. Os resultados, no entanto, indicam que esses métodos não são eficientes para problemas muito grandes.

Dentre as técnicas heurísticas, a de escalonamento incremental é uma das mais simples. Dadas as operações a serem realizadas, essa técnica consiste em tomar aquela que termina primeiro, respeitando as restrições, e adicioná-la à sequência. Esse processo se repete até que todas as operações foram agendadas ou até quando não é mais possível alocá-las sem a violação de restrições. Isso é uma grande desvantagem dessa técnica, pois pode não apresentar nenhuma solução, ou tornar o planejamento em uma busca de tentativa e erro (ARISHA *et al.*, 2001).

O método de pesquisa em vizinhança parte de uma solução factível e faz pequenas alterações na mesma, checando em seguida se as alterações trouxeram melhorias. O processo se repete até que uma condição de parada seja atingida. Esse procedimento tem como solução final uma solução que é um ótimo local e não há como saber se também é um ótimo global. Esse método é aplicado em trabalhos como os de Roshanaei *et al.* (2009), Bürgy (2017) e Pena *et al.* (2016).

A busca tabu é uma técnica que busca contornar as deficiências do método anterior. Buscando entre os vizinhos de uma solução, um ótimo local é obtido e é chamado de tabu. Esse ótimo é armazenado e busca-se encontrar um novo ótimo local. Se esse novo ótimo encontrado for melhor que o anterior, então ele passa ser o novo tabu (PEZZELLA; MERELLI, 2000), (AHANI; ASYABANI, 2014).

Quanto às técnicas avançadas, existem diversas abordagens (ARISHA *et al.*, 2001). Redes neurais (MAHMOOD; BASHIR, 2011), (SACHE, 2014), e lógica *fuzzy* (RAMKUMAR *et al.*, 2011), (BILKAY *et al.*, 2004), foram aplicadas para reduzir a complexidade computacional. Algoritmos genéticos também são aplicados em diversos trabalhos (LI; CHEN, 2010), (BHATT; CHAUHAN, 2015). Observa-se porém, uma tendência de combinação de diversas técnicas, o que dá origem aos métodos chamados de híbridos (NING *et al.*, 2016), (TAVAKKOLI-MOGHADDAM *et al.*, 2005), (TANG *et al.*, 2011), (PENA *et al.*, 2016).

2.2 Teoria de Controle Supervisório

Nesta seção será feita uma breve introdução à Teoria de Controle Supervisório - TCS - proposta por Ramadge e Wonham (1989). Maiores detalhes serão dados na Seção 3.3.

A TCS tem o objetivo de sintetizar uma estrutura chamada de supervisor, que restringe o comportamento em malha fechada de um sistema a eventos discretos a um comportamento desejado. O supervisor e o modelo do sistema são representados por autômatos, que são grafos orientados. Cada vértice é chamado de estado enquanto cada aresta é uma transição, cujos rótulos são os eventos. A Figura 3 mostra a estrutura básica da TCS. O comportamento do sistema, ou planta, é guiado pela ocorrência de eventos, que podem ser classificados como controláveis, sobre os quais o supervisor pode agir, e não controláveis, que não sofrem a ação do supervisor. Para evitar que a planta entre em um estado indesejado, ou em uma situação de bloqueio, o supervisor atua desabilitando eventos controláveis. O supervisor é minimamente restritivo e possui um comportamento permissivo, pois não força a planta a um determinado comportamento, mas evita que ela entre em estados indesejáveis.

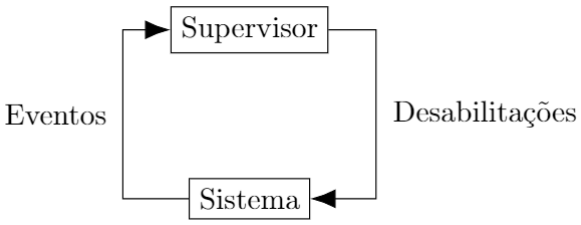


Figura 3 – Estrutura da Teoria de Controle Supervisório

A TCS descreve o processo para síntese de um supervisor único, abordagem que é conhecida como Controle Monolítico. Para o caso de sistemas muito complexos, a síntese do supervisor monolítico torna-se inviável, já que os autômatos envolvidos crescem exponencialmente com o tamanho do sistema. As extensões da TCS, conhecidas como Controle Modular (RAMADGE; WONHAM, 1989) e Controle Modular Local (QUEIROZ; CURY, 2000) visam contornar esse problema da explosão de estados, com a síntese de vários supervisores de tamanhos menores. A ideia básica dessas extensões é a divisão do sistema completo em subsistemas menores. Para cada especificação desejada obtém-se um supervisor. A diferença entre as abordagens reside na planta considerada para a obtenção do supervisor. Enquanto o controle modular considera a planta como um todo, o controle modular local considera como planta apenas os subsistemas que são diretamente afetados pelas especificações.

No entanto, ao dividir o problema, surge o conceito de conflito, que ocorre quando um supervisor interfere no comportamento de outro, podendo levar o sistema como um todo

a uma situação de bloqueio. O conflito é uma característica não desejável e busca-se sempre a obtenção de supervisores não conflitantes. Identificar o conflito e sua causa, em geral, não é trivial. Pena *et al.* (2009) propõem um método eficiente para a verificação do conflito de supervisores, enquanto trabalhos como os de Hill *et al.* (2008) e Wong *et al.* (1998) tratam da resolução de conflito. Outra solução para o conflito é a utilização de um supervisor monolítico no lugar dos supervisores conflitantes.

2.3 O Controle Hierárquico e abstrações

Para lidar com o crescimento exponencial do tamanho dos modelos com o aumento da complexidade dos sistemas físicos, fenômeno também conhecido como explosão de estados, é comum decompor, horizontalmente e verticalmente, o problema maior em subproblemas que são mais facilmente tratáveis. Exemplos de decomposição horizontal são as abordagens de controle modular e controle modular local, já citadas. A decomposição vertical é tratada por Wong e Wonham (1996) e é conhecida como controle hierárquico.

O controle hierárquico é uma estrutura composta de dois níveis, no qual o nível superior é um modelo do processo do nível inferior e é guiado por ele através de um canal de informações. Para garantir que as ações impostas pelo controlador do nível superior sejam realizáveis pelo nível inferior, é introduzido o conceito de consistência hierárquica. Esse conceito se resume na garantia de que o não bloqueio seja preservado entre os dois níveis. Isso é garantido se o canal de informação for um observador, conceito também proposto no trabalho.

A aplicação do controle hierárquico se dá na forma de abstrações. Diversos trabalhos aplicam abstrações em diferentes etapas da síntese de supervisores. Cunha e Cury (2007) aplicam a TCS no nível inferior do controle hierárquico, enquanto no nível superior os sistemas são tratados como tendo marcação flexível, conceito apresentado por Cury *et al.* (2004). Assim é possível obter uma representação compacta do comportamento marcado em malha fechada do nível superior como uma abstração do comportamento em malha fechada do nível inferior.

Abstrações são utilizadas por Moor *et al.* (2013) com o objetivo de reduzir a complexidade na síntese de supervisores descentralizados. Ao invés de considerar o sistema completo, o trabalho utiliza a projeção natural para obter abstrações compostas de eventos de alto nível que são relevantes para a síntese dos supervisores. Como vários supervisores são envolvidos, é importante garantir que a operação conjunta dos mesmos não leve ao bloqueio do sistema. O trabalho apresenta então condições sobre a escolha dos eventos de alto nível para

que tenham uma consistência hierárquica com o sistema original. Em continuação a esse trabalho, Moor (2014) propõe condições necessárias e suficientes para que as abstrações deem origem a supervisores não conflitantes.

Em (TEIXEIRA *et al.*, 2015), a TCS é aplicada em uma modelagem com autômatos estendidos. Isso permite uma simplificação dos modelos das especificações. Além disso, é apresentada uma forma de realizar abstrações das variáveis dos autômatos estendidos, bem como um algoritmo para a síntese de supervisores a partir das abstrações, reduzindo o esforço computacional se comparado à síntese de supervisores da TCS. Por fim, é apresentada uma maneira de verificar se o supervisor obtido é minimamente restritivo. Caso não seja, é apresentado um método para identificar variáveis dos autômatos estendidos que, se usadas na síntese dos supervisores, levam eventualmente a uma solução minimamente restritiva.

Abstrações também são utilizadas na resolução de conflitos. Pena *et al.* (2009) propõem métodos eficientes para a verificação de não conflito em supervisores modulares locais. Isso é realizado através da aplicação do teste de não conflito sobre abstrações dos supervisores, que consistem na projeção natural dos mesmos com a propriedade do observador. No entanto, para garantir que o teste aplicado sobre as abstrações seja equivalente ao teste aplicado nos supervisores originais, são apresentadas condições sobre o alfabeto que é mantido na projeção, chamado de alfabeto de eventos relevantes.

2.4 Aplicação da TCS na solução de problemas de planejamento

Retomando o modelo conceitual da Figura 2, no otimizador interno do Planejador, as sequências geradas são obtidas a partir de informação sobre o modelo do sistema, que determina o espaço de busca. Pode-se reduzir significativamente o espaço de busca com a utilização de um supervisor obtido pela TCS como modelo do sistema, já que ele descreve apenas as sequências que respeitam as restrições. No entanto, o supervisor obtido pela versão clássica da TCS não traz informações que permitem o avaliador quantificar o quão boa uma solução é. Faz-se necessário então acrescentar ao modelo do sistema, informações sobre o custo de cada solução de forma coerente ao objetivo a ser alcançado.

Diversos trabalhos utilizam a abordagem de sistemas a eventos discretos e a TCS para a solução do problema de planejamento, como em (ABDEDDAÏM; MALER, 2001). Nesse trabalho, modela-se um sistema de manufatura através de um autômato temporizado. A solução do problema de planejamento é obtida pela busca do menor caminho com relação

ao tempo, baseando-se em ideias provenientes da área de verificação, otimização e planejamento. O trabalho não apresenta ganhos ou perdas em relação ao tempo de computação para obtenção da solução, se comparado com outras técnicas. A vantagem consiste na utilização de autômatos temporizados a modelagem do problema, o que a torna mais simples e direta.

Em (MARCHAND *et al.*, 2002) o sistema, que é parcialmente observável, é modelado por um autômato. Aos eventos são relacionadas informações quantitativas de custo. O autômato é então transformado em um observador, que é um novo autômato completamente observável. Em seguida, um controlador ótimo correspondente ao comportamento ótimo desejado é sintetizado. Por fim, utilizando a técnica de *back propagation*, o supervisor é construído, baseado nos custos dos eventos. Esse supervisor representa o comportamento ótimo em relação aos eventos observáveis, no sentido em que ele tem as melhores aproximações das trajetórias não observáveis entre dois eventos observáveis.

A modelagem por autômatos temporizados também é aplicada por Wakatake *et al.* (2009). Cada componente de um sistema é modelado por um autômato temporizado. Para evitar a explosão de estados resultante da composição paralela, que representa o comportamento global do sistema, uma técnica de decomposição e coordenação é proposta. O algoritmo de busca da solução do problema de planejamento é do tipo *branch and bound*, combinando análise de alcançabilidade com programação linear.

Panek *et al.* (2004) também se utilizam de autômatos temporizados para a modelagem de sistemas de manufatura, com objetivo de construir uma árvore de alcançabilidade. Aplicando técnicas de programação inteira, propõe-se um algoritmo para identificar e podar limites inferiores da árvore. Isso se traduz na redução do espaço de busca. Novas heurísticas baseadas em programação inteira são propostas, com o objetivo de encontrar a solução ótima no espaço de busca reduzido. Os autores destacam que, embora o tempo computacional para obtenção da solução é maior se comparado com os métodos tradicionais de programação inteira mista, o tempo necessário para a modelagem do problema como autômatos temporizados é muito menor do que para a modelagem algébrica do problema.

A Teoria de Controle Supervisório é utilizada por Kobetski e Fabian (2006) para a obtenção de um supervisor que atua em um sistema de manufatura, composto de diversos robôs. O supervisor obtido é maximamente permissivo, não bloqueante e garante que não há colisões entre robôs. Através de técnicas de programação linear, deseja-se obter as sequências ótimas para a operação dos robôs. O trabalho propõe um método que converte um modelo de simulação 3D em um supervisor temporizado, além da formulação algébrica do problema, que é resolvido com a técnica de programação linear.

Em (KOBETSKI *et al.*, 2007) os formalismos de autômatos determinísticos, probabilísticos e estendidos são combinados com objetivo de modelar sistemas de manufatura, nos quais eventos não controláveis como a quebra de equipamento são considerados. Deseja-se então minimizar a expectativa do *makespan*, através de um método baseado no algoritmo A*.

Na busca da solução do problema de planejamento de sistemas complexos, Fabre e Jezequel (2009) exploram a modularidade das ações, ou seja, o fato de que uma ação afeta apenas uma pequena parte do sistema. Essa abordagem é conhecida como planejamento fatorado, e cada subsistema é chamado de agente, sendo considerado um novo problema de planejamento, porém menor. O plano ótimo é obtido para cada agente, mas deve-se garantir que esses planos locais são compatíveis entre si, podendo dar origem a um plano global válido. Em continuação a esse trabalho, Jezequel e Fabre (2012), propõem a aplicação de algoritmos turbo para a solução de planejamento fatorado. Apesar de terem sido aplicados em problemas aleatórios, os resultados mostram que o algoritmo converge com poucas iterações e obtém uma solução ótima ou próxima do ótimo, em um autômato com 10^{50} estados.

Su *et al.* (2012) utilizam a modelagem por autômatos, os quais têm suas transições associadas com pesos positivos, que representam os tempos de disparo, para a solução do problema de planejamento. Uma técnica derivada da teoria de *heaps of pieces* é utilizada para a obtenção de uma cadeia com mínimo *makespan*. Com o objetivo de aplicar esse algoritmo centralizado em problemas maiores, Su (2012) propõe, como continuação do trabalho anterior, um método de síntese ótima baseada numa abstração do modelo. Utilizando matrizes *max-plus* para agregar a informação do tempo, os modelos de autômatos temporizados são reduzidos em modelos mais simples quando as transições locais são apagadas. Essa abstração pode ser aplicada em cada componente do sistema, permitindo assim sua aplicação em problemas de grande porte. A aplicação dessa abstração é estendida para o caso de supervisores cíclicos por Ware e Su (2015).

Em (ALVES *et al.*, 2016) é proposta uma solução para problema de planejamento, em que deseja-se maximizar o número de operações ocorrendo de forma paralela. A sequência de máximo paralelismo não necessariamente é ótima em relação ao *makespan*, mas apresenta boas soluções em tempo e que podem ser usadas como ponto de partida para outros métodos de busca. O trabalho apresenta um algoritmo para a obtenção da sequência de máximo paralelismo que tem complexidade linear no tamanho do problema.

O problema de planejamento para grandes sistemas é solucionado com a aplicação de um método de otimização heurística por Pena *et al.* (2016). O problema é modelado atra-

vés de supervisores obtidos pela síntese modular local e um algoritmo baseado na técnica *Variable Neighborhood Search*, ou VNS, é empregado. Os resultados obtidos indicam que a metodologia proposta alcança significativa redução de *makespan*, quando comparado com algoritmos gulosos. Além disso, a metodologia proposta é robusta contra distúrbios nos parâmetros dos modelos, que é um obstáculo na aplicação do planejamento em sistemas reais.

Hill e Lafortune (2016) reduzem a complexidade da busca pela solução do problema de planejamento através do emprego de um abstração sobre o supervisor obtido pela TCS. Para cada transição do supervisor, associa-se um número que indica o custo da ocorrência de um evento. Dois estados são equivalentes em custo se os caminhos que os ligam até um estado final têm o mesmo custo. Escolhe-se então eventos que quando apagados pela operação de projeção natural, a equivalência de custo entre os estados é mantida. Os estados equivalentes são agrupados, dando origem a um autômato menor, que é a abstração do supervisor. Há também uma preocupação em garantir que qualquer solução obtida na abstração seja realizável no autômato original, mesmo com a presença de eventos não controláveis.

Novamente considerando o modelo conceitual da Figura 2, observa-se que há uma correspondência entre as ações e eventos do modelo conceitual com os eventos controláveis e não controláveis da TCS. Assim como os eventos controláveis, as ações podem ser forçadas ou impedidas de acontecer. Os eventos do modelo conceitual correspondem aos eventos não controláveis da TCS, que são a resposta do sistema e nunca podem ser impedidos de acontecer. Como o Controlador só pode atuar ativamente sobre os eventos controláveis, considera-se válido o argumento de que o plano ótimo obtido pelo Planejador seja composto apenas de eventos controláveis. Baseando-se nisso, Vilela e Pena (2016) propõem uma abstração que consiste no autômato obtido após a projeção do supervisor monolítico no alfabeto de eventos controláveis. A condição suficiente para que qualquer sequência obtida na abstração seja possível de ser implementada no sistema físico, com a presença dos eventos não controláveis, é que a projeção natural para o alfabeto de eventos controláveis tenha a propriedade do observador. Todavia, o resultado apresentado por Vilela e Pena (2016) por si só não é a solução do problema de planejamento, mas uma ferramenta que pode ser usada com outros métodos de otimização.

Uma continuação do trabalho de Vilela e Pena (2016) é proposta por Alves e Pena (2017), onde propõe-se a aplicação de abstrações em dois supervisores obtidos pela síntese modular local. O presente trabalho estende o resultado de Vilela e Pena (2016) para lidar com supervisores obtidos pela síntese modular local, com o objetivo de reduzir a complexidade envolvida no cálculo dos supervisores e abstrações, e estabelece novas condições para que o resultado de Alves e Pena (2017) possa ser aplicado.

3 Preliminares

Neste capítulo serão apresentados os conceitos básicos aplicados ao estudo de Sistemas a Eventos Discretos. Esses conceitos serão essenciais para a compreensão do restante do trabalho.

3.1 Sistemas a Eventos Discretos

Sistemas físicos são uma interconexão de componentes, dispositivos ou subsistemas. O efeito de um agente externo sobre um sistema é o de mudar o estado do mesmo, o que pode ser expresso em função de seus parâmetros internos, não necessariamente mensuráveis ou observáveis, e que, possivelmente causa uma mudança nos parâmetros de saída (OPPENHEIM; WILLLSKY, 2010). Com o objetivo de estudá-los, modelos físicos e matemáticos, que representam certos aspectos do sistema, podem ser utilizados. Os mesmos são classificados de acordo com o tipo de modelo que melhor descreve os aspectos de interesse.

Em geral os modelos apresentam variáveis de entrada, de saída e de estados. Por definição, o estado de um sistema em determinado instante de tempo t_0 é a informação necessária, em conjunto com a entrada, para a determinação da saída de maneira única para todo $t \geq t_0$. O processo de modelagem consiste em determinar relações matemáticas entre essas variáveis, definindo assim a dinâmica do sistema (CASSANDRAS; LAFORTUNE, 2007).

O conjunto de todos os valores possíveis que as variáveis de estado podem assumir, é chamado de espaço de estados. Quando o mesmo é um conjunto de vetores reais, diz-se que o espaço de estados é contínuo. Por outro lado, o espaço de estados é discreto quando é formado por um conjunto discreto. Nesse caso, as variáveis de estado mudam de valor apenas em momentos discretos no tempo. A essa transição instantânea dá-se o nome de evento.

Os eventos podem representar ações individuais, como uma pessoa apertando um botão, ou podem representar que um conjunto de condições foram atendidas simultaneamente. Diferentes ações são representadas por diferentes eventos, que juntos formam o conjunto de eventos, que também é discreto. Quando em um sistema de espaço de estados discreto, as transições de um estado para outro, ou eventos, acontecem de maneira independente do tempo, diz-se que o sistema é orientado a eventos.

Um sistema que apresenta um espaço de estados discretos bem como é orientado a eventos, é por definição um Sistema a Eventos Discretos, ou SED (CASSANDRAS; LAFORTUNE, 2007). Muitos sistemas, em particular os sistemas criados pelo homem, nos domínios de manufatura, robótica, organização e serviços de entrega, tráfico de veículos, computação, redes de informação, entre outros, são de fato Sistemas a Eventos Discretos. Até mesmo processos que tradicionalmente são classificados como contínuos, podem ser vistos como discretos. Um exemplo é um reservatório em que seu nível é classificado como $\{BAIXO, NORMAL, ALTO\}$.

Existem várias técnicas para se modelar os Sistemas a Eventos Discretos. Dependendo do aspecto que se deseja analisar, uma pode apresentar vantagem sobre a outra. A modelagem na qual se baseia esse trabalho é a de autômatos não temporizados. Nas próximas sessões serão apresentados os conceitos desenvolvidos por Ramadge e Wonham (1989) e também expostos em Cassandras e Lafortune (2007), que possibilitarão a compreensão do trabalho.

3.2 Linguagens e Autômatos

O conjunto Σ de eventos de um SED é também chamado de alfabeto e assume-se que ele é sempre finito. Uma sequência de eventos de um alfabeto forma uma palavra ou cadeia. Uma cadeia formada de zero eventos é uma cadeia vazia e é denotada por ε. O comprimento de uma cadeia é o número de eventos que a forma, incluindo também as repetições. O comprimento de uma cadeia s é denotado por $|s|$. Convenciona-se que $|\varepsilon| = 0$.

Uma linguagem definida sobre um alfabeto Σ é qualquer conjunto de cadeias finitas formadas com elementos de Σ. Na próxima sessão serão apresentadas as principais operações que podem ser aplicadas às linguagens e que foram extensivamente utilizadas nesse trabalho.

3.2.1 Operações sobre linguagens

Uma das operações básicas sobre linguagens é a concatenação. A concatenação de duas cadeias u e v é uma nova cadeia uv, formada pelos eventos de u seguidos pelos eventos de v. A cadeia vazia ε é o elemento neutro da concatenação, ou seja, $\varepsilon u = u\varepsilon = u$ para qualquer cadeia u. Seja uma cadeia $tuv = s$. Então t é chamado de prefixo de s, u uma subcadeia de s e v é um sufixo de s. Usa-se a notação $t \leq s$ para indicar que t é um prefixo de s. Tanto ε quanto s são prefixos, subcadeias e sufixos de s.

O operador fechamento Kleene, denotado por Σ^*, resulta no conjunto de todas as

cadeias finitas formadas com elementos de Σ, incluindo a cadeia vazia ε. Uma linguagem é então, qualquer subconjunto de Σ^*.

As operações usuais de conjuntos, como união, interseção, diferença e complemento, são aplicáveis às linguagens, já que as mesmas são conjuntos. A operação de prefixo fechamento de uma linguagem L é denotada por $\overline{L}$ e consiste de todos os prefixos de todas as cadeias de L. Em geral, $L \subseteq \overline{L}$. No entanto, se $L = \overline{L}$, então L é chamada de linguagem prefixo fechada ou L-fechada. O fechamento Kleene também pode ser aplicado sobre linguagens. Um elemento de L^* é formado pela concatenação de um número finito de elementos de L, incluído também a concatenação de zero elementos.

Uma operação que será amplamente usada nesse trabalho é a projeção natural ou simplesmente projeção, realizada de um alfabeto Σ_M para outro Σ_m, onde $\Sigma_m \subseteq \Sigma_M$. A operação é denotada pela letra P e índices podem ser adicionados para facilitar a compreensão no caso de vários conjuntos estarem envolvidos. A projeção é definida a seguir:

$$P : \Sigma_M^* \rightarrow \Sigma_m^*$$

onde

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

Em palavras, a projeção toma uma cadeia formada no alfabeto Σ_M e apaga aqueles eventos que não pertencem ao alfabeto Σ_m. A operação inversa da projeção é a função

$$P^{-1} : \Sigma_m^* \rightarrow 2^{\Sigma_M^*}$$

definida como

$$P^{-1}(t) \coloneqq \{s \in \Sigma_M^* : P(s) = t\}$$

em que, dado um conjunto A, a notação 2^A é o conjunto de todos os subconjuntos de A. Em palavras, a projeção inversa P^{-1} de uma cadeia t retorna todas as cadeias de Σ_M^* que projetam, com P, para a referida cadeia. A projeção e a projeção inversa podem ser estendidas para linguagens simplesmente aplicando-as para todas as cadeias da linguagem. Para $L \subseteq \Sigma_M^*$,

$$P(L) \coloneqq \{t \in \Sigma_m^* : (\exists s \in L)[P(s) = t]\}$$

e para $L_m \subseteq \Sigma_m^*$,

$$P^{-1}(L_m) := \{s \in \Sigma_M^* : (\exists t \in L_m)[P(s) = t]\}.$$

A noção de projeção inversa pode ser utilizada para prover a definição formal de operação de composição paralela ou composição síncrona, denotada por $||$, de linguagens $L_i \subseteq \Sigma_i^*$, com $i \in I$ e $\Sigma = \bigcup_{i \in I} \Sigma_i$:

$$\Big|\Big|_{i \in I} L_i = \bigcap_{i \in I} P_i^{-1}(L_i).$$

As projeções tem um papel muito importante nesse trabalho, já que é a operação aplicada para se obter as abstrações. Considere que A, B e L são linguagens definidas no alfabeto Σ. Considere também a projeção $P : \Sigma^* \to \Sigma_i^*$, com $\Sigma_i \subseteq \Sigma$. A seguir são apresentadas algumas propriedades, extraídas de (CASSANDRAS; LAFORTUNE, 2007), que são repetidamente utilizadas:

1) $P[P^{-1}(L)] = L$
   $L \subseteq P^{-1}[P(L)]$

2) Se $A \subseteq B$, então $P(A) \subseteq P(B)$ e $P^{-1}(A) \subseteq P^{-1}(B)$

3) $P(A \cup B) = P(A) \cup P(B)$
   $P(A \cap B) \subseteq P(A) \cap P(B)$

4) $P^{-1}(A \cup B) = P^{-1}(A) \cup P^{-1}(B)$
   $P^{-1}(A \cap B) = P^{-1}(A) \cap P^{-1}(B)$

5) $P(AB) = P(A)P(B)$
   $P^{-1}(AB) = P^{-1}(A)P^{-1}(B)$

A projeção natural pode ter uma propriedade conhecida como propriedade do observador, apresentada na Definição 1.

Definição 1 ((WONG *et al.*, 1998)). *Seja uma linguagem $L \subseteq \Sigma^*$, um alfabeto $\Sigma_i \subseteq \Sigma$ e $P_{\Sigma \to \Sigma_i} : \Sigma^* \to \Sigma_i^*$ a projeção natural de cadeias em Σ^* para cadeias em Σ_i^*. Se $(\forall a \in \overline{L})$ $(\forall b \in \Sigma_i^*)$, $P_{\Sigma \to \Sigma_i}(a)b \in P_{\Sigma \to \Sigma_i}(L) \implies (\exists c \in \Sigma^*)P_{\Sigma \to \Sigma_i}(ac) = P_{\Sigma \to \Sigma_i}(a)b$ e $ac \in L$, então a projeção natural $P_{\Sigma \to \Sigma_i}(L)$ tem a propriedade do observador.* □

Para facilitar o entendimento da propriedade do observador, considere a Figura 4. A elipse mais interna, à esquerda, delimita uma linguagem L, definida sobre o alfabeto Σ, enquanto a elipse mais interna, à direita, representa a projeção natural dessa linguagem para o alfabeto Σ_i. Escolhe-se uma cadeia a que é um prefixo de uma cadeia em L. Em seguida faz-se a projeção de a para o alfabeto Σ_i. Completa-se então a projeção de a, $P_{\Sigma\to\Sigma_i}(a)$, com um sufixo b, formado apenas de elementos de Σ_i, levando à obtenção de uma cadeia que está na projeção de L, ou seja $P_{\Sigma\to\Sigma_i}(a)b \in P_{\Sigma\to\Sigma_i}(L)$. Essas operações são indicadas pelas setas contínuas. Se a projeção possui a propriedade do observador, então, para todo a e b tal que $P_{\Sigma\to\Sigma_i}(a)b \in P_{\Sigma\to\Sigma_i}(L)$, existirá c tal que $P_{\Sigma\to\Sigma_i}(ac) = P_{\Sigma\to\Sigma_i}(a)b$ e $ac \in L$, que é o caminho representado pelas setas tracejadas. Como consequência, tem-se que $P_{\Sigma\to\Sigma_i}(c) = b$.

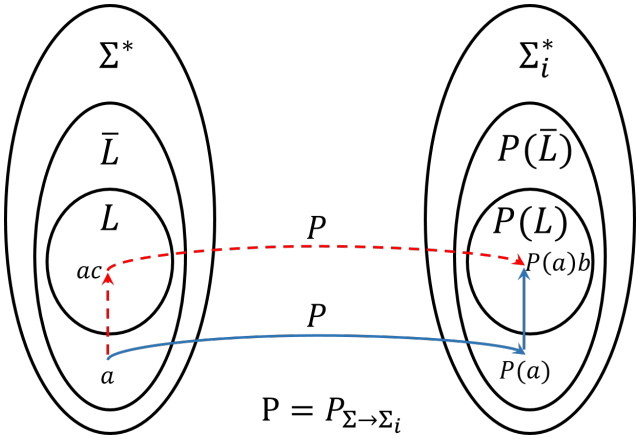


Figura 4 – Representação gráfica da propriedade do observador

A linguagem obtida após a operação de projeção natural pode ser chamada de abstração. Uma abstração que possua a propriedade do observador é chamada de OP-abstração. A propriedade do observador pode ser verificada utilizando o algoritmo apresentado em (PENA *et al.*, 2014).

3.2.2 Autômatos

As linguagens são uma forma de descrever o comportamento de SEDs. No entanto, é conveniente a representação das linguagens por um meio gráfico. Uma dessas formas é por meio de autômatos. Esses podem ser representados como grafos orientados capazes de representar linguagens através de regras bem definidas. Um autômato determinístico de estados finitos é uma quintúpla

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

onde:

Q é o conjunto finito de estados.

Σ é o conjunto finito de eventos associados à G;

$\delta : Q \times \Sigma \to Q$ é a função de transição: $\delta(x, e) = y$ significa que existe uma transição rotulada pelo evento e do estado x ao estado y. A função de transição é dita parcial quando não há transições com todos os eventos em todos os estados. O autômato é determinístico se $\delta(x, e) = y_1$ e $\delta(x, e) = y_2$ sempre implica que $y_1 = y_2$, para todo x e e;

q_0 é o estado inicial;

$Q_m \subseteq Q$ é o conjunto de estados marcados. A determinação de qual ou quais estados são marcados é um problema de modelagem. Geralmente os estados marcados representam a completude de uma operação ou tarefa.

Uma forma bastante comum de se representar os autômatos é através do diagrama de transição. Esse é um grafo orientado em que os nós, representados por círculos, são os estados e os arcos, representados por setas, são as transições definidas pela função de transição. O rótulo de cada arco é o evento associado à transição. Estados marcados são representados por círculos duplos, enquanto o estado inicial é identificado por uma seta sem rótulo. A Figura 5 mostra um exemplo de autômato determinístico de estados finitos, em que $Q = \{0, 1\}$, $\Sigma = \{\alpha, \beta\}$, $q_0 = 0$, $Q_m = \{0\}$, e $\delta(0, \alpha) = \{1\}$, $\delta(1, \beta) = \{0\}$, $\delta(0, \beta) = \emptyset, \delta(1, \alpha) = \emptyset$.

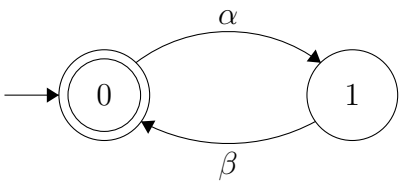


Figura 5 – Autômato determinístico

A função de transição pode ser estendida para lidar com cadeias, $\delta : Q \times \Sigma^* \to Q$. Nesse caso, $\delta(q, \varepsilon) = \varepsilon$ e $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$, com $q \in Q, s \in \Sigma^*$ e $\sigma \in \Sigma$. Um autômato G representa dois tipos de linguagens, a linguagem gerada

$$\mathcal{L}(G) \coloneqq \{s \in \Sigma^* : \delta(q_0, s) \neq \emptyset\}$$

e a linguagem marcada

$$\mathcal{L}_m(G) \coloneqq \{s \in \mathcal{L}(G) : \delta(q_0, s) \in Q_m\}.$$

A linguagem $\mathcal{L}(G)$ representa todos os caminhos que podem ser seguidos através do autômato, a partir do estado inicial, enquanto a linguagem marcada $\mathcal{L}_m(G)$ representa todos os

caminhos a partir do estado inicial que podem ser seguidos através do autômato que levam a um dos estados marcados. A cadeia correspondente a um caminho é a concatenação dos eventos dos rótulos das transições.

Se um autômato G possui algum estado que não pode ser alcançado por nenhuma sequência a partir do estado inicial, então o estado é chamado de não acessível. A operação $Ac(G)$ remove todos os estados não acessíveis e as transições associadas à eles. Essa operação não afeta a linguagem gerada e marcada de G. Se $G = Ac(G)$, então o autômato é dito acessível.

Por outro lado, se G possui um estado a partir do qual não é possível alcançar nenhum estado marcado, então esse estado é dito não coacessível. A operação $CoAc(G)$ remove todos os estados não coacessíveis de G. Essa operação não afeta a linguagem marcada, mas pode reduzir a linguagem gerada. Se $G = CoAc(G)$, o autômato é chamado de coacessível e nesse caso, $\mathcal{L}(G) = \overline{\mathcal{L}_m(G)}$.

Quando um autômato é não coacessível, ele é chamado de bloqueante. O bloqueio pode ocorrer de duas formas. A primeira é o *deadlock*, que ocorre quando para algum $q \in (Q \setminus Q_m)$, $\delta(q, e) = \emptyset$, $\forall e \in \Sigma$. Ou seja, se em um estado q, não marcado, nenhum evento pode ocorrer, então o autômato fica bloqueado nesse estado q. A segunda forma de bloqueio é chamada de *livelock*, que ocorre quando um conjunto de estados não marcados formam uma componente fortemente conexa, ou seja, os estados são acessíveis entre si, e não há transição que saia desse conjunto. Apesar de eventos sempre serem possíveis de ocorrer em um *livelock*, o estado marcado nunca é alcançado. Destaca-se que um autômato não coacessível não será bloqueante se a componente não coacessível for não acessível.

Se um autômato é acessível e coacessível, então diz-se que o autômato é aparado, ou *Trim*. Um autômato aparado é obtido por:

$$Trim(G) := CoAc[Ac(G)] = Ac[CoAc(G)]$$

As operações de projeção, projeção inversa e composição paralela também podem ser aplicadas diretamente sobre os autômatos. Seja G um autômato com alfabeto Σ. Considere que $\Sigma_m \subseteq \Sigma$. As projeções de $\mathcal{L}(G)$ e $\mathcal{L}_m(G)$ de Σ^* para Σ_m^*, $P_{\Sigma \to \Sigma_m}[\mathcal{L}(G)]$ e $P_{\Sigma \to \Sigma_m}[\mathcal{L}_m(G)]$, podem ser realizadas em G pela substituição de todos os eventos que estão em $\Sigma \setminus \Sigma_m$ por ε. O resultado é um autômato não-determinístico que tem as linguagens geradas e marcadas desejadas. O autômato não-determinístico pode então ser transformado em um determinístico que gera e marca as mesmas linguagens. Maiores informações sobre como obter um autômato determinístico a partir de um não determinístico podem ser obtidas em (CASSANDRAS; LAFORTUNE, 2007).

Com relação à projeção inversa, seja o alfabeto Σ_m tal que $\mathcal{L}(G) \subseteq \Sigma_m^*$. Seja também o um alfabeto Σ_M tal que $\Sigma_M \supset \Sigma_m$. Seja P a projeção de Σ_M para Σ_m. Um autômato que tem a linguagem gerada $P^{-1}[\mathcal{L}(G)]$ e marca $P^{-1}[\mathcal{L}_m(G)]$ pode ser obtido pela adição de auto laços para todos os eventos em $\Sigma_M \setminus \Sigma_m$ em todos os estados de G.

O produto de dois autômatos G_1 e G_2 é definido como

$$G_1 \times G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

onde

$$\delta((x_1, x_2), e) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{se } \delta_1(x_1, e) \neq \emptyset \wedge \delta_2(x_2, e) \neq \emptyset \\ \text{indefinido} & \text{caso contrário} \end{cases}$$

Nessa operação, as transições dos dois autômatos precisam sempre estar sincronizadas em um evento comum, ou seja, um evento em $\Sigma_1 \cap \Sigma_2$. Dessa forma, um evento só pode ocorrer se puder ocorrer nos dois autômatos. As linguagens marcada e geradas são obtidas por

$$\mathcal{L}(G_1 \times G_2) = \mathcal{L}(G_1) \cap \mathcal{L}(G_2)$$
$$\mathcal{L}_m(G_1 \times G_2) = \mathcal{L}_m(G_1) \cap \mathcal{L}_m(G_2)$$

Assim, a interseção de duas linguagens pode ser obtida através realização do produto de suas representações como autômatos.

A operação de composição paralela tem como resultado um autômato que modela o comportamento de um conjunto de dois ou mais autômatos que operam concorrentemente. Essa é, em geral, a operação utilizada para a construção de modelos completos de sistemas compostos de subsistemas menores. A composição paralela dos autômatos G_1 e G_2 é o autômato

$$G_1||G_2 = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

onde

$$\delta((x_1, x_2), e) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{se } \delta_1(x_1, e) \neq \emptyset \wedge \delta_2(x_2, e) \neq \emptyset \\ (\delta_1(x_1, e), x_2) & \text{se } \delta_1(x_1, e) \neq \emptyset \wedge e \notin \Sigma_2 \\ (x_1, \delta_2(x_2, e)) & \text{se } \delta_2(x_2, e) \neq \emptyset \wedge e \notin \Sigma_1 \\ \text{indefinido} & \text{caso contrário} \end{cases}$$

Na composição paralela um evento comum, ou seja, que pertença a $\Sigma_1 \cap \Sigma_2$, pode ser executado se os dois autômatos podem executá-lo simultaneamente. Isso faz com que

os autômatos sejam sincronizados nos eventos comuns. Os eventos privados, pertencentes a $(\Sigma_1 \setminus \Sigma_2) \cup (\Sigma_2 \setminus \Sigma_1)$, não sofrem nenhuma restrição e podem ser executados sempre que possível.

Se $\Sigma_1 = \Sigma_2$, então a composição paralela se torna igual ao produto, já que todas as transições são forçadas a serem sincronizadas. Se $\Sigma_1 \cap \Sigma_2 = \emptyset$, então não há transições sincronizadas e qualquer comportamento possível em G_1 e G_2 também é possível em $G_1||G_2$. As linguagens gerada e marcada são

$$\mathcal{L}(G_1||G_2) = P^{-1}_{\Sigma\to\Sigma_1}[\mathcal{L}(G_1)] \cap P^{-1}_{\Sigma\to\Sigma_2}[\mathcal{L}(G_2)]$$

$$\mathcal{L}_m(G_1||G_2) = P^{-1}_{\Sigma\to\Sigma_1}[\mathcal{L}_m(G_1)] \cap P^{-1}_{\Sigma\to\Sigma_2}[\mathcal{L}_m(G_2)]$$

Ao se tratar de autômatos e linguagens, é comum representar o autômato e a linguagem gerada por ele, pelo mesmo símbolo. A diferença se dá pelo contexto. Assim, pode-se dizer que A é um autômato e que $A \subseteq \Sigma^*$. No segundo caso, A é a linguagem gerada pelo autômato A.

3.3 Teoria de Controle Supervisório

Em geral, os SEDs são compostos de diversos subsistemas que funcionam concorrentemente e interagem entre si. É mais intuitivo realizar a modelagem de cada subsistema individualmente, na forma de n autômatos G_j, $j \in J = \{1, ..., n\}$, de modo que $\Sigma_{G_j} \cap \Sigma_{G_k} = \emptyset$, com $j \neq k$ e $j, k \in J$. O comportamento conjunto, ou a planta global G, de todos esses subsistemas é obtido pela composição paralela de todos eles, ou seja, $G = \|_{j\in J} G_j$.

O autômato G modela o comportamento em malha aberta de um SED. Deseja-se, no entanto, exercer alguma ação de controle sobre o sistema, evitando que o mesmo alcance determinados estados, que podem representar o bloqueio ou uma situação insegura. Essas restrições, ou especificações, são modeladas como m autômatos E_i, com $i \in I = \{1, ..., m\}$, e a especificação global E é obtida por $E = \|_{i\in I} E_i$. O controle de um SED consiste em garantir o funcionamento do sistema sem que as restrições sejam desrespeitadas. Isso é obtido pela proibição da ocorrência de determinados eventos em determinadas situações.

Surge então o conceito de eventos controláveis e não controláveis, que formam uma partição para o alfabeto Σ de um SED. Assim $\Sigma = \Sigma_c \cup \Sigma_u$, onde Σ_c e Σ_u são, respectivamente, o conjunto de eventos controláveis e não controláveis. Os eventos em Σ_c podem ser desabilitados a qualquer momento, enquanto Σ_u agrupa os eventos que não sofrem a ação de um agente controlador. Num sistema físico real, a ação sobre atuadores pode ser modelada

por eventos controláveis, enquanto a finalização de uma tarefa, a leitura de um sensor ou a quebra de uma máquina são representados por eventos não controláveis.

O agente controlador, chamado supervisor, para cada estado do sistema G sob controle, seleciona um conjunto de eventos que, se executados, não deixam que o comportamento de G desrespeite as restrições. É possível implementar o supervisor na forma de um autômato S, no qual a ação de controle de S sobre G está implícita na estrutura de transições. O comportamento de G sob o controle de S é obtido pela operação de produto. Assim, um evento e pode ocorrer quando $S \times G$ está no estado (x, q) somente se $\delta_S(x, e) \neq \emptyset$ e $\delta_G(q, e) \neq \emptyset$.

O problema na Teoria de Controle Supervisório, ou TCS, é modificar o comportamento em malha aberta $\mathcal{L}(G)$ de de um SED G de modo a restringi-lo a um comportamento desejado K, com $K \subseteq \mathcal{L}(G)$. Em geral, $K = E||G$. Surge então a questão: o comportamento desejado K é possível de ser alcançado por um supervisor? A resposta é dada pela propriedade da controlabilidade. Assim, $K \subseteq \Sigma^*$ é controlável se

$$\overline{K}\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{K}.$$

Essa propriedade estabelece que para K ser controlável em relação à $\mathcal{L}(G)$, todo prefixo w de uma cadeia em K, ou seja, $\forall w \in \overline{K}$, seguido de um evento $\sigma \in \Sigma_u$, de modo que $w\sigma \in \mathcal{L}(G)$, então ele precisa ser também um prefixo de uma cadeia em $\overline{K}$, ou $w\sigma \in \overline{K}$. Desde que eventos não controláveis não podem ser impedidos de acontecer, então se um desses eventos ocorrer ao longo de uma cadeia em K, a continuação dessa cadeia precisa continuar em K para que K seja um comportamento realizável em malha fechada.

Quando K é não controlável em relação à G, pode-se dizer que existe uma máxima sublinguagem controlável, denotada por $Sup\mathcal{C}(K, G) \subseteq K$. Assim, se K for não controlável, a máxima sublinguagem controlável contida em K preserva as restrições impostas por K, requirindo o mínimo de ações de controle e pode ser vista como a aproximação ótima e minimamente restritiva de K. Se os autômatos de K e G tiverem respectivamente k e l estados, então o cálculo de $Sup\mathcal{C}(K, G)$ tem complexidade polinomial em k e l. Porém, o número de estados de K e G cresce exponencialmente com o número de subsistemas G_j. A obtenção de um supervisor S, tal que $S = Sup\mathcal{C}(K, G)$, e $K = E||G$, é chamada de síntese monolítica, pois apenas um supervisor é obtido.

3.3.1 Controle Modular Local

Com o objetivo de evitar o problema da explosão de estados da síntese monolítica, que é um fator limitante para aplicação da TCS em sistemas muito complexos, Queiroz e

Cury (2000) desenvolveram uma abordagem conhecida como modular local.

O primeiro passo para aplicação dessa abordagem é a modelagem da planta como representação por sistema produto, ou RSP. Para obter uma RSP mais refinada possível, faz-se a composição paralela de todos os subsistemas que possuem eventos em comum, criando-se um conjunto com o maior número possível de subsistemas assíncronos distintos, cada qual com o mínimo de estados.

O comportamento desejado do sistema sob a ação do controle é expressado na forma de autômatos, chamados de especificações genéricas locais, que são equivalentes às especificações da TCS. Seja a RSP de um sistema G formado por subsistemas G_j, $j \in J$.Sejam as especificações genéricas locais $E_{gen,i}$, $\forall i \in I$. A planta local $G_{loc,i}$, $\forall i \in I$, associada à especificação $E_{gen,i}$ é definida por:

$$G_{loc,i} = \Big\| \limits_{j \in J_{loc,i}} G_j, \text{ com } J_{loc,i} = \{k \in J | \Sigma_{G_k} \cap \Sigma_{E_{gen,i}} \neq \emptyset\},$$

em que Σ_{G_k} e $\Sigma_{E_{gen,i}}$ são os alfabetos do subsistema G_k e da especificação genérica $E_{gen,i}$, respectivamente. Assim, a planta local $G_{loc,i}$ é composta apenas pelos subsistemas da modelagem original que estão diretamente restringidos por $E_{gen,i}$. A partir das especificações genéricas, é possível então obter as especificações locais $E_{loc,i} = E_{gen,i} || \mathcal{L}_m(G_{loc,i})$ e a linguagem desejada global $K_{glo,i} = E_{gen,i} || \mathcal{L}_m(G)$, com $G = \|_{j \in J} G_j$.

Com o objetivo de reduzir o esforço computacional exigido para a obtenção de supervisores, a abordagem de controle modular local explora a modularidade das especificações e a estrutura geralmente descentralizada dos sistemas de manufatura. Isso é realizado por meio da síntese de não apenas um, mas vários supervisores modulares locais $S_i = Sup\mathcal{C}(E_{loc,i}, G_{loc,i})$.

No entanto, ao colocar vários supervisores operando de forma concorrente, existe a possibilidade de que haja um bloqueio do sistema. Isso acontece quando dois ou mais supervisores são conflitantes. A Definição 2 estabelece a condição para que os supervisores sejam considerados não conflitantes.

Definição 2 (Supervisores não conflitantes)**.** *Um conjunto de supervisores S_j com $j \in J$, são ditos não conflitantes, ou localmente modulares, se a igualdade*

$$\Big\| \limits_{j \in J} \overline{S_j} = \overline{\Big\| \limits_{j \in J} S_j}$$

se verifica. □

É importante ressaltar que a verificação de conflito é uma operação computacionalmente custosa, já que envolve a composição paralela de todos os supervisores. Diversos trabalhos, como (FLORDAL; MALIK, 2006; PENA *et al.*, 2009; MALIK, 2015), propõem métodos eficientes para a verificação do conflito. Técnicas de resolução de conflitos são propostas em (WONG *et al.*, 1998; HILL *et al.*, 2008). Trabalhos mais recentes propõem métodos para síntese de supervisores que já são não conflitantes (MALIK; LEDUC, 2012; MOOR, 2014). Ao longo do restante do trabalho, será sempre assumido que os supervisores modulares locais são não conflitantes.

O Teorema 1 estabelece a relação entre a síntese modular local e a monolítica. A condição de modularidade local, garante que a síntese modular local não apresenta perda de performance em relação ao controle monolítico.

Teorema 1 ((QUEIROZ; CURY, 2000)). *Dados um sistema RSP formada por* G_j, $j = 1, ..., m$, *e as especificações* $E_{gen,i}$, $i = 1, ..., n$. *Se* $Sup\mathcal{C}(E_{loc,i}, G_{loc,i}), i = 1, ..., n$ *for localmente modular, então* $Sup\mathcal{C}(\bigcap_{i=1}^{n} K_{glo,i}, G) = \|_{i=1}^{n} Sup\mathcal{C}(E_{loc,i}, G_{loc,i})$. ♦

Em outras palavras, o Teorema 1 diz que se supervisores modulares locais são não conflitantes, então a composição paralela deles ($\|_{i=1}^{n} Sup\mathcal{C}(E_{loc,i}, G_{loc,i})$) é equivalente ao supervisor monolítico obtido a partir da mesma planta e especificações ($Sup\mathcal{C}(\bigcap_{i=1}^{n} K_{glo,i}, G)$).

A Figura 6 exemplifica a estrutura do controle modular local. Seja um sistema composto por subsistemas M_j, com $j \in J = \{1, 2, 3, 4\}$. As setas que ligam as especificações genéricas, $E_{gen,i}$, $\forall i \in I = \{1, 2, 3\}$, aos subsistemas indicam a restrição do comportamento de M_j por $E_{gen,i}$. Para cada especificação, obtém-se a planta local, $G_{loc,i}$, para $j \in J$. Depois da obtenção das especificações locais $E_{loc,i} = E_{gen,i}||\mathcal{L}_m(G_{loc,i})$, obtém-se os supervisores $S_i = Sup\mathcal{C}(E_{loc,i}, G_{loc,i}), \forall i \in I$.

Os subsistemas que compõem as plantas locais $G_{loc,i}$ podem ser divididos em plantas restritas e plantas comuns, apresentadas nas Definições 3 e 4, respectivamente.

Definição 3 (Planta restrita). *Uma planta restrita,* G_{ri}, *é a composição paralela de todos os subsistemas que compõem* $G_{loc,i}$ *e que não são compartilhados por outras plantas locais.* □

Definição 4 (Planta comum). *A planta comum,* $G_{c\{i,k\}}$, *é a composição de todos os subsistemas que fazem parte da composição das plantas locais* $G_{loc,i}$ *e* $G_{loc,k}$. □

Para exemplificar, na Figura 6, tem-se $G_{r1} = M_1$, $G_{c\{1,2\}} = M_2$, $G_{c\{2,3\}} = M_3$ e $G_{r3} = M_4$.

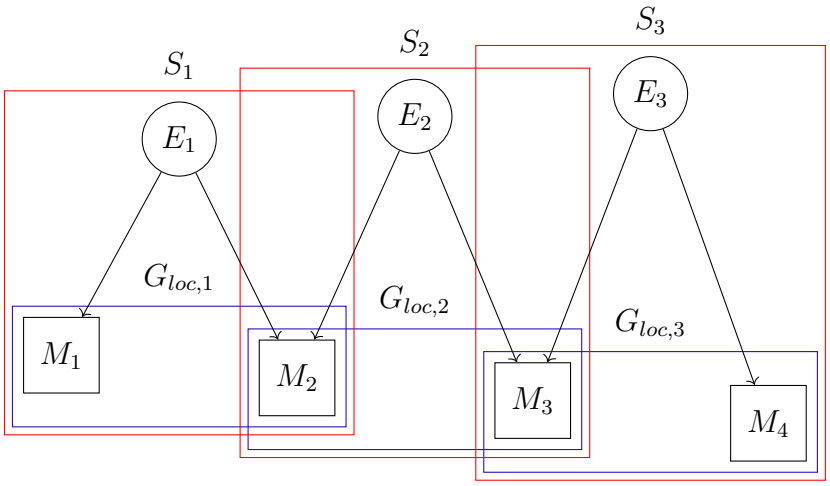


Figura 6 – Estrutura do controle modular local

4 Resultados Anteriores

Neste capítulo estão sintetizados os resultados de outros trabalhos e que serviram de base para a elaboração do resultado principal desse trabalho. A primeira seção apresenta as condições para aplicação da abstração em supervisores obtidos pela síntese monolítica. A segunda seção apresenta condições para que a propriedade distributiva de projeções em composições paralelas seja aplicável.

4.1 Abstrações de supervisores obtidos pela síntese monolítica

Em (VILELA; PENA, 2016) é apresentada uma abstração do supervisor, consistindo na projeção natural do mesmo para o alfabeto dos eventos controláveis, Σ_c. Salienta-se que ao utilizar a abstração é preciso garantir que qualquer plano obtido a partir da mesma seja capaz de ser executado no sistema físico, com a presença dos eventos não controláveis e ainda garantindo as condições de segurança e não bloqueio. A condição suficiente é que a projeção natural do supervisor para o alfabeto de eventos controláveis possua a propriedade do observador. Esse resultado é sintetizado no Teorema 2.

Teorema 2 ((VILELA; PENA, 2016))**.** *Seja G um sistema, S a máxima sublinguagem controlável contida na linguagem desejada K e $P_{\Sigma\to\Sigma_c} : \Sigma^* \to \Sigma_c^*$ a projeção natural. Para toda sequência $s \in P_{\Sigma\to\Sigma_c}(S)$ e $A = P^{-1}_{\Sigma\to\Sigma_c}(s) \cap S$, se $P_{\Sigma\to\Sigma_c}(S)$ possui a propriedade do observador, então A será controlável em relação a $\mathcal{L}(G)$.* ♦

Em palavras, o Teorema 2 garante que a linguagem recuperada A, que é composta de todas as cadeias de S que projetam para a cadeia s obtida na abstração do supervisor $P_{\Sigma\to\Sigma_c}(S)$, é capaz de ser executada na planta G. Isso se traduz na propriedade de controlabilidade de A em relação à G.

Para aplicação do Teorema 2, é necessário verificar se a projeção natural do supervisor para o alfabeto de eventos controláveis possui a propriedade do observador. No entanto, a necessidade desta verificação pode ser evitada com a utilização dos resultados adicionais apresentados por Vilela e Pena (2016). Inicialmente, faz-se duas definições:

Definição 5 (Sistema de Produção (VILELA; PENA, 2016))**.** *Um sistema de produção G*

é definido como um sistema composto por m *autômatos* $M_i = (\Sigma_i, Q_i, \delta_i, Q_i^m, q_i^0)$, $i \in I$, $I = \{1, ..., m\}$ *tal que:*

i) $G = \underset{i=1}{\overset{m}{\|}} M_i, \Sigma = \bigcup_{i=1}^{m} \Sigma_i$;

ii) $\Sigma_i \cap \Sigma_j = \emptyset, i, j \in I$ *e* $i \neq j$;

iii) M_i *é caracterizado por:*

a) $\forall \sigma \in \Sigma_i$ *tal que* $\delta(q_0, \sigma) \neq \emptyset$ *então* $\sigma \in \Sigma_{ic}$;

b) $P_{\Sigma_i \to \Sigma_{ic}}(M_i)$ *possui a propriedade do observador.*

□

A Definição 5 apresenta as características para que o modelo de um sistema seja considerado um sistema de produção. Basicamente, o sistema deve ser composto de subsistemas com alfabetos disjuntos, todo evento possível no estado inicial deve ser controlável e a projeção de cada subsistema para o alfabeto dos eventos controláveis deve possuir a propriedade do observador.

Definição 6 (Especificação de Coordenação (VILELA; PENA, 2016))**.** *A especificação* $E = (\Sigma_E, Q_E, \delta_E, Q_E^m, q_E^0)$ *é uma especificação de coordenação se:*

i) $\mathcal{L}(E)$ *é controlável em relação à* $\mathcal{L}(G)$;

ii) $\forall s \in \Sigma^*, t \in \Sigma_u^*, \sigma \in \Sigma_c, u \in \Pi(t). [st\sigma \in \mathcal{L}(E)]$ *e* $[su \in \mathcal{L}(E)] \implies su\sigma \in \mathcal{L}(E)$;

em que $\Pi(t)$ *é o operador de permutação de uma cadeia* t*, uma linguagem formada por todas as cadeias obtidas por permutações (com possíveis repetições) de eventos de* t*.* □

Uma especificação de coordenação, segundo a Definição 6, além de ser controlável em relação à $\mathcal{L}(G)$, não deve apresentar eventos controláveis que são factíveis ou não, dependendo da ordem de eventos não controláveis que os precedem.

As duas definições apresentam as características necessárias para a aplicação da Proposição 1:

Proposição 1 ((VILELA; PENA, 2016)). *Seja a planta G um sistema de produção de acordo com a Definição 5. Seja E uma especificação de coordenação, conforme a Definição 6 e $P_{\Sigma\to\Sigma_c}:\Sigma^*\to\Sigma_c^*$ uma projeção natural. Se $K=G||E$ é controlável ($S=Sup\mathcal{C}(K,G)=K$), então $P_{\Sigma\to\Sigma_c}(S)$ tem a propriedade do observador.* $\Diamond$

A Proposição 1 diz que se a planta é modelada como um sistema de produção e as especificações como especificações de coordenação, então a projeção natural para o alfabeto de eventos controláveis do supervisor obtido, $S=E||G$, possui a propriedade do observador. Assim, não há a necessidade de se realizar a verificação da presença ou não da propriedade do observador na projeção do supervisor.

4.2 Condições para aplicação da propriedade distributiva de projeções em composições paralelas

Em (PENA, 2007) é apresentado um método eficiente para a verificação de conflito entre supervisores modulares locais. Ao invés de realizar a verificação diretamente sobre os supervisores, a verificação do conflito sobre a abstração dos supervisores é proposta. Essa abstração consiste na projeção natural dos supervisores para um alfabeto de eventos relevantes, Σ_r. Esse trabalho apresenta então, critérios para a escolha dos eventos que compõem Σ_r de modo que a verificação do conflito na abstração dos supervisores seja equivalente à verificação do conflito sobre os supervisores originais.

Dois dos resultados intermediários são as Proposições 2 e 3, nas quais se mostra que, sob certas condições, as igualdades $P_{\Sigma\to\Sigma_r}(S_1||S_2)=P_{\Sigma_1\to\Sigma_{1r}}(S_1)||P_{\Sigma_2\to\Sigma_{2r}}(S_2)$ e $P_{\Sigma\to\Sigma_r}\left(\mathop{\|}_{j=1}^{m}S_j\right)=\mathop{\|}_{j=1}^{m}P_{\Sigma_j\to\Sigma_{jr}}(S_j)$ são verdadeiras. Estas igualdades serão utilizadas nas demonstrações dos resultados no próximo capítulo. No entanto, duas das condições necessárias para aplicação das Proposições 2 e 3, a saber $\bigcup_{j=1}^{m}(\Sigma_{E_j})\subseteq\Sigma_r$, e $S_j=\overline{S_j}\cap\mathcal{L}_m(G)$ não são atendidas, pois neste trabalho, $\Sigma_r=\Sigma_c$ e os supervisores são marcadores, ou seja, modificam a marcação da planta. Faz-se necessário então adequar a demonstração para que o resultado seja aplicável no contexto deste trabalho. Isso é realizado no próximo capítulo. As demonstrações das Proposições 2 e 3 foram omitidas, mas podem ser encontradas na íntegra em (PENA, 2007), na seção 6.1.

Inicialmente apresenta-se a Proposição 2.

Proposição 2 ((PENA, 2007)). *Sejam S_1 e S_2 linguagens implementadas pelos supervisores.*

A igualdade $P_{\Sigma\to\Sigma_r}(S_1||S_2) = P_{\Sigma_1\to\Sigma_{1r}}(S_1)||P_{\Sigma_2\to\Sigma_{2r}}(S_2)$ *se verifica se:*

1) $P_{\Sigma_1\to\Sigma_{1r}}(S_1)$ *e* $P_{\Sigma_2\to\Sigma_{2r}}(S_2)$ *são OP-abstrações;*

2) K_1 *e* K_2 *são controláveis em relação a* $\mathcal{L}(G_1)$ *e* $\mathcal{L}(G_2)$*, respectivamente (*$S_1 = K_1$ *e* $S_2 = K_2$*);*

3) $(\Sigma_{E_1} \cup \Sigma_{E_2}) \subseteq \Sigma_r$*;*

4) $S_1 = \overline{S_1} \cap \mathcal{L}_m(G_1)$ *e* $S_2 = \overline{S_2} \cap \mathcal{L}_m(G_2)$

$\Diamond$

Finalmente, apresenta-se a Proposição 3, que estende a proposição anterior para lidar com mais de dois supervisores.

Proposição 3 ((PENA, 2007))**.** *Sejam* S_j*,* $\forall j \in J$*, linguagens implementadas pelos supervisores. A igualdade* $P_{\Sigma\to\Sigma_r}\left(\mathop{\|}_{j=1}^{m} S_j\right) = \mathop{\|}_{j=1}^{m} P_{\Sigma_j\to\Sigma_{jr}}(S_j)$ *se verifica se,* $\forall j \in J$*,*

1) $P_{\Sigma_j\to\Sigma_{jr}}(S_j)$ *são OP-abstrações;*

2) K_j *são controláveis em relação a* $\mathcal{L}(G_j)$ $(S_j = K_j)$*;*

3) $\bigcup_{j=1}^{m}(\Sigma_{E_j}) \subseteq \Sigma_r$*;*

4) $S_j = \overline{S_j} \cap \mathcal{L}(G_j)$*.*

$\Diamond$

Os conceitos apresentados nesse capítulo e no Capítulo 3 fundamentam a contribuição principal deste trabalho, que será apresentada no próximo capítulo.

5 Resultados

O teorema apresentado na seção 4.1 trata da aplicação da abstração sobre o supervisor obtido pela síntese monolítica. No entanto, a aplicação dessa abordagem pode atingir os limites computacionais disponíveis, já que a síntese monolítica está sujeita à explosão de estados. A síntese modular local visa reduzir os recursos computacionais necessários, por meio do cálculo de não apenas um, mas vários supervisores. Apesar de o número de supervisores a serem calculados ser maior, o tamanho dos autômatos envolvidos é menor.

Antes de apresentar a principal contribuição deste trabalho, serão apresentados alguns resultados que auxiliarão a demonstração do mesmo. Inicialmente, serão apresentados os resultados obtidos para aplicação em problemas que tenham dois supervisores. Na seção seguinte, será feita a generalização destes mesmos resultados para aplicação em problemas com número arbitrário de supervisores.

Com relação à nomenclatura empregada, Σ_j é o alfabeto da planta local G_j e $\Sigma = \bigcup_{j \in J} \Sigma_j$ é o alfabeto global, união dos alfabetos de todas as plantas envolvidas. O alfabeto dos eventos controláveis é denominado Σ_c, enquanto o dos não controláveis é denotado por Σ_u (de *uncontrollable*). O subíndice duplo dos alfabetos, a não ser que seja indicado o contrário, é a interseção de dois alfabetos. Assim, $\Sigma_{jc} = \Sigma_j \cap \Sigma_c$. O alfabeto de eventos compartilhados é denotado por $\Sigma_s = \bigcup_{k \neq j} (\Sigma_k \cap \Sigma_j)$ com $k, j \in J$ (de *shared*).As plantas restritas de S_j são denotadas por G_{rj}, enquanto as plantas compartilhadas pelos supervisores S_j e S_k são denotadas por $G_{c\{j,k\}}$.

Inicialmente, apresenta-se a Definição 7, que é uma condição para aplicação dos resultados. Esta definição diz respeito à forma com que o sistema é modelado.

Definição 7 (Sistema Modular Local Coordenado)**.** *Sejam S_j, $\forall j \in J$, supervisores obtidos pela síntese modular local, em que $S_j = Sup\mathcal{C}(K_j, G_j)$, com $K_j = E_j || G_j$. O conjunto de supervisores S_j é um Sistema Modular Local Coordenado se possui as seguintes características:*

1) *As plantas G_j são Sistemas de Produção conforme a Definição 5;*
2) *As especificações E_j e a especificação $E = \|_{j=1}^{m} E_j$ são Especificações de Coordenação conforme a Definição 6. Além disso, $\forall E_j$, $Q_E^m = \{q_E^0\}$;*

3) Os supervisores S_j são não conflitantes conforme a Definição 2.

□

Se alguma das especificações não for controlável, deve-se encontrar uma nova especificação que seja controlável. Isso pode ser feito metodicamente através da redução de supervisores (PENA *et al.*, 2009). O supervisor $S = Sup\mathcal{C}(K, G)$, com $K = E||G$, é calculado e, através do algoritmo proposto por (SU; WONHAM, 2004), um supervisor reduzido S_{red}, tal que $S = S_{red}||G$, é obtido. Usa-se então S_{red} como a nova especificação E'. Assim $K' = E'||G$ é controlável e $S = K'$, atendendo ao item 2) da Definição 7.

5.1 Dois supervisores

A definição de Sistemas de Produção foi introduzida em (VILELA; PENA, 2016) e apresentada na Seção 4.1 (Definição 5). Quando utiliza-se esse tipo de modelo na síntese modular local, é de interesse saber se essa característica se estende à composição de sistemas de produção. Este resultado é apresentado no Lema 1.

Lema 1. *Se G_1 e G_2 são sistemas de produção conforme a Definição 5, então $G = G_1||G_2$ também será um sistema de produção.* ◇

Demonstração. Um sistema de produção, de acordo com a Definição 5, é caracterizado por:

i) $G = \mathop{\Big\|}_{i=1}^{m} M_i, \Sigma = \bigcup_{i=1}^{m} \Sigma_i$

ii) $\Sigma_i \cap \Sigma_j = \emptyset, i, j \in I$ e $i \neq j$

iii) M_i é caracterizado por:

   a) $\forall \sigma \in \Sigma_i$ tal que $\delta(q_0, \sigma) \neq \emptyset$ então $\sigma \in \Sigma_{ic}$

   b) $P_{\Sigma_i \to \Sigma_{ic}}(M_i)$ possui a propriedade do observador

Para provar que $G = G_1||G_2$ é um sistema de produção, basta mostrar que ele apresenta as características acima relacionadas. Isso pode ser feito de maneira direta, graças à propriedade associativa da operação de composição paralela.

i) Sejam G_1 e G_2 sistemas de produção compostos pelos subsistemas M_{1i} e M_{2j}, respectivamente, em que $i \in I = \{1, ..., m\}$ e $j \in J = \{1, ..., n\}$. Os alfabetos de M_{1i} e M_{2j} são Σ_{1i} e Σ_{2j}, respectivamente, para todo $i \in I$ e $j \in J$. Então $G_1 = \mathop{\Big\|}_{i=1}^{m} M_{1i}$ e $G_2 = \mathop{\Big\|}_{j=1}^{n} M_{2j}$. Logo,

$$G = \left(\mathop{\Big\|}_{i=1}^{m} M_{1i} \right) || \left(\mathop{\Big\|}_{j=1}^{n} M_{2j} \right)$$

.

Considerando a característica do controle modular local, é possível que G_1 e G_2 compartilhem subsistemas. Desta forma, para todo i e j tal que $\Sigma_{1i} = \Sigma_{2j}$ e $M_{1i} = M_{2j}$, o resultado ainda continua válido, já que nessa situação $M_{1i}||M_{2j} = M_{1i} = M_{2j}$.

ii) Como G_1 e G_2 são sistemas de produção, então os subsistemas que os compõem não compartilham eventos. Ou seja, para todo $i, k \in I$, com $i \neq k$, então $\Sigma_{1i} \cap \Sigma_{1k} = \emptyset$. Analogamente, para todo $j, l \in J$, com $j \neq l$, então $\Sigma_{2j} \cap \Sigma_{2l} = \emptyset$.

Caso G_1 e G_2 compartilhem subsistemas, ou seja, para todo $i \in I$ e $j \in J$ tal que $\Sigma_{1i} = \Sigma_{2j}$ e $M_{1i} = M_{2j}$, a característica *ii)* não é violada, já que ela afirma que subsistemas diferentes não podem compartilhar eventos, mas aqui tratam-se de subsistemas iguais.

Assim, pode-se dizer que G possui a característica *ii)*.

iii) Como os alfabetos dos subsistemas que compõem G são disjuntos, então a operação de composição paralela não altera as características *iii)a)* e *iii)b)*, pois $P_{\Sigma \to \Sigma_{1i}}(G) = M_{1i}$ e $P_{\Sigma \to \Sigma_{2j}}(G) = M_{2j}$, para todo $i \in I$ e $j \in J$.

□

A seguir é apresentada a Proposição 4. Ela afirma que a projeção natural para o alfabeto de eventos controláveis da composição paralela de supervisores que compõem um Sistema Modular Local Coordenado é uma OP-abstração.

Proposição 4. *Sejam S_1 e S_2 supervisores que compõem um Sistema Modular Local Coordenado, conforme a Definição 7. Então $P_{\Sigma \to \Sigma_c}(S_1||S_2)$ é OP-abstração.* ◊

Demonstração. Seja $K = E||G$, com $E = E_1||E_2$ e $G = G_1||G_2$. Como os supervisores são não conflitantes (item *3)* da Definição 7), então pelo Teorema 1, $S = S_1||S_2 = Sup\mathcal{C}(K, G)$. Sabe-se que E é especificação de coordenação (item *2)* da Definição 7) e que G é sistema de

produção (Lema 1). Assim, pela Proposição 1, $P_{\Sigma\to\Sigma_c}(S) = P_{\Sigma\to\Sigma_c}(S_1||S_2)$ é OP-abstração. □

Para a demonstração do resultado principal, faz-se necessário também que a igualdade $P_{\Sigma\to\Sigma_c}(S_1||S_2) = P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2)$ seja verdadeira. Este resultado intermediário é apresentado na Proposição 5. A demonstração desta proposição foi baseada na demonstração da Proposição 2, apresentada no capítulo anterior.

No entanto, algumas modificações tiveram que ser realizadas em relação à demonstração original. Esta modificação foi necessária pois na Proposição 2 duas das condições necessárias para sua aplicação não estão presentes no contexto deste trabalho. A primeira delas é a exigência de que $(\Sigma_{E_1} \cup \Sigma_{E_2}) \subseteq \Sigma_r$, onde Σ_r é chamado de alfabeto de eventos relevantes e é composto pelos eventos que são mantidos na projeção natural. Como neste trabalho o alfabeto de eventos relevantes é Σ_c, então $(\Sigma_{E_1} \cup \Sigma_{E_2}) \not\subseteq \Sigma_r = \Sigma_c$. A segunda condição não atendida é $S_i = \overline{S_i} \cap \mathcal{L}_m(G)$. Esta condição estabelece que S_i deve ser $\mathcal{L}_m(G)$-fechada, ou seja, que a marcação do autômato que representa o supervisor (e portanto da especificação) seja consistente com a marcação da planta. Isto também não é verdade aqui pois os supervisores podem modificar a marcação da planta.

Para facilitar o entendimento da demonstração da Proposição 5, foram criados dois lemas, apresentados a seguir. Sabe-se que $P_{\Sigma\to\Sigma_c}(S_1||S_2) \subseteq P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2)$ é sempre verdadeira. A prova da igualdade consiste portanto em mostrar que a relação $P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2) \subseteq P_{\Sigma\to\Sigma_c}(S_1||S_2)$ também é verdadeira. Partindo então de uma cadeia $a = \sigma_1\sigma_2...\sigma_n \in P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2) \subseteq \Sigma_c^*$, a ideia geral da demonstração é a de construir iterativamente cadeias s_1, t_1, s_2 e t_2 tais que

$$s_1t_1 \in S_1,$$

$$s_2t_2 \in S_2,$$

e

$$a \in P_{\Sigma\to\Sigma_c}(s_1t_1||s_2t_2) \subseteq P_{\Sigma\to\Sigma_c}(S_1||S_2).$$

Lema 2. *Sejam S_1 e S_2 supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7 e sejam as cadeias a, s_1' e s_1 tais que:*

$$\begin{aligned} a &= \sigma_1\sigma_2...\sigma_n \in P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2) \subseteq \Sigma_c^*; \\ s_1' &= \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^*; \\ s_1 &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1') \in \overline{S_1}. \end{aligned}$$

Para todo s_1' e s_1 com as características acima, existem também um s_2' e s_2 tais que:

$$\begin{aligned} P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_2}(s_2') &= s_2 \in \overline{S_2}; \\ P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_c}(s_2') &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}(s_1') = a; \\ P_{\Sigma_1\to\Sigma_s}(s_1) &= P_{\Sigma_2\to\Sigma_s}(s_2). \end{aligned}$$

◇

Demonstração. Partindo de

$$a \in P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2) \subseteq \Sigma_c^*, \tag{5.1}$$

então $P_{\Sigma_c\to\Sigma_{1c}}(a) \in P_{\Sigma_1\to\Sigma_{1c}}(S_1)$ e $P_{\Sigma_c\to\Sigma_{2c}}(a) \in P_{\Sigma_2\to\Sigma_{2c}}(S_2)$.

Como $a = \sigma_1\sigma_2...\sigma_n$ e $s_1' = \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^*$, então

$$P_{\Sigma_1\to\Sigma_{1c}}(s_1) = P_{\Sigma_1\to\Sigma_{1c}}[P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1')] = P_{\Sigma_c\to\Sigma_{1c}}(a).$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ e $i = \{1, ..., n\}$.

Hipótese: Para todo s_1' com as características acima, existe também um $s_2' = \mu_1^2\sigma_1\mu_2^2\sigma_2...\mu_n^2\sigma_n$ tal que

$$\begin{aligned} P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_2}(s_2') &= s_2 \in \overline{S_2}, \\ P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_c}(s_2') &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}(s_1') = a \end{aligned}$$

A demonstração usa a estratégia indutiva. Assume-se que para algum $i \geq 1$, $s_1'^{(i-1)} = \mu_1^1\sigma_1...\mu_{i-1}^1\sigma_{i-1}$, com $\mu_1^1...\mu_{i-1}^1$ definido de tal forma que $P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}(s_1') = a$ e

$$s_1^{(i-1)} = P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}\left(s_1'^{(i-1)}\right) = P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(\mu_1^1\sigma_1...\mu_{i-1}^1\sigma_{i-1}) \in \overline{S_1},$$

existe $s_2'^{(i-1)} = \mu_1^2\sigma_1...\mu_{i-1}^2\sigma_{i-1}$, com $\mu_1^2...\mu_{i-1}^2$ definido de tal forma que

$$\begin{aligned} P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_s}(\mu_i^2) &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_s}(\mu_i^1) \\ P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_c}(s_2') &= a \end{aligned}$$

e

$$s_2^{(i-1)} = P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_2}\left(s_2'^{(i-1)}\right) = P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_2}(\mu_1^2\sigma_1...\mu_{i-1}^2\sigma_{i-1}) \in \overline{S_2}, \tag{5.2}$$

$$P_{\Sigma_1 \to \Sigma_s}(s_1) = P_{\Sigma_2 \to \Sigma_s}(s_2) \tag{5.3}$$

pretende-se mostrar que para qualquer $s_1'^{(i)}$ e $s_1^{(i)}$com as características desejadas, existem $s_2'^{(i)}$ e $s_2^{(i)}$ tais que as Equações (5.2) e (5.3) são verdadeiras.

Base: para $i = 1$, $s_1'^{(0)} = \varepsilon$, $s_1^{(0)} = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(0)}\right) \in \overline{S_1}$ e então há um $s_2'^{(0)} = \varepsilon$, tal que $s_2^{(0)} = P_{(\Sigma_2 \cup \Sigma_c) \to \Sigma_2}\left(s_2'^{(0)}\right) \in \overline{S_2}$. Além disso, $P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}\left(s_1'^{(0)}\right) = P_{(\Sigma_2 \cup \Sigma_c) \to \Sigma_c}\left(s_2'^{(0)}\right) = \varepsilon \in \overline{a}$ e $P_{\Sigma_1 \to \Sigma_s}(s_1) = P_{\Sigma_2 \to \Sigma_s}(s_2) = \varepsilon$.

Passo de Indução: Para $2 \leq i \leq n$, se $s_1'^{(i)} = s_1'^{(i-1)}\mu_i^1\sigma_i = \mu_1^1\sigma_1 ... \mu_{i-1}^1\sigma_{i-1}\mu_i^1\sigma_i$, com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ definidos de tal forma que $s_1^{(i)} = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i)}\right) \in \overline{S_1}$, então

$$\begin{aligned} s_1^{(i)} &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i)}\right) = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i-1)}\mu_i^1\sigma_i\right) \\ &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i-1)}\right) P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\mu_i^1\sigma_i\right) \\ &= s_1^{(i-1)} P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\mu_i^1\sigma_i\right) = s_1^{(i-1)}\mu_i^1 P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\sigma_i\right) \in \overline{S_1} \end{aligned} \tag{5.4}$$

Pretende-se construir uma cadeia $\mu_i^2 \in (\Sigma_2 \backslash \Sigma_{2c})^*$ tal que $P_{(\Sigma_1 \backslash \Sigma_{1c}) \to \Sigma_s}(\mu_i^1) = P_{(\Sigma_2 \backslash \Sigma_{2c}) \to \Sigma_s}(\mu_i^2)$, $P_{(\Sigma_2 \cup \Sigma_c) \to \Sigma_2}\left(s_2'^{(i-1)}\mu_i^2\right) \in \overline{S_2}$ e $P_{\Sigma_1 \to \Sigma_s}\left(s_1^{(i-1)}\mu_i^1\right) = P_{\Sigma_2 \to \Sigma_s}\left(s_2^{(i-1)}\mu_i^2\right)$.

Como (a) $s_2^{(i-1)} \in \overline{S_2}$; (b) eventos em $P_{(\Sigma_2 \backslash \Sigma_{2c}) \to \Sigma_s}(\mu_i^2)$ são não controláveis (o que implica que eles nunca são desabilitados pelo supervisor S_2), pode-se então concluir que

$$s_2^{(i-1)}\mu_i^2 \in \overline{S_2} \tag{5.5}$$

Para mostrar que $s_2'^{(i)} = s_2'^{(i-1)}\mu_i^2\sigma_i$ é tal que $s_2^{(i)} = P_{(\Sigma_2 \cup \Sigma_r) \to \Sigma_2}\left(s_2'^{(i)}\right) \in \overline{S_2}$, significando que, ao completar $s_2^{(i-1)}\mu_i^2$ com o sufixo σ_i não vai levar a cadeia a não pertencer a $\overline{S_2}$, considera-se os dois casos, em relação ao pertencimento de σ_i em Σ_2:

a) $\sigma_i \in \Sigma_2$ (σ_i é um evento de S_2). Nesta situação, dois casos ainda podem ser analisados:

   i) $\sigma_i \in E_2$
   Como $\overline{S_2} = E_2 || \mathcal{L}(G_2)$, então $P_{\Sigma_2 \to \Sigma_{E_2}}\left(s_2^{(i-1)}\mu_i^2\right) \in \mathcal{L}(E_2)$.
   Pelo fato de E_2 ser uma especificação de coordenação, se para algum μ_i^2 verifica-se a veracidade de $P_{\Sigma_2 \to \Sigma_{E_2}}\left(s_2^{(i-1)}\mu_i^2\sigma_i\right) \in \mathcal{L}(E_2)$, então para todo μ_i^2 tal que $P_{\Sigma_2 \to \Sigma_{E_2}}\left(s_2^{(i-1)}\mu_i^2\right) \in \mathcal{L}(E_2)$, também será verdade que $P_{\Sigma_2 \to \Sigma_{E_2}}\left(s_2^{(i-1)}\mu_i^2\sigma_i\right) \in$

$\mathcal{L}(E_2)$ (item ii) da Definição 6). Além disso $E_2 \subseteq P_{\Sigma_2 \to \Sigma_{E_2}}(S_2)$ e, portanto $s_2^{(i-1)}\mu_i^2\sigma_i \in \overline{S_2}$.

ii) $\sigma_i \notin E_2$

Neste caso, o fato de $\sigma_i \notin E_2$, implica que o evento nunca será desabilitado pelo supervisor. Como $\sigma_i \in \Sigma_2$, logo $s_2^{(i-1)}\mu_i^2\sigma_i \in \overline{S_2}$.

b) $\sigma_i \notin \Sigma_2$ (σ_i não é um evento de S_2)

Neste caso,

$$\begin{aligned} s_2^{(i)} &= P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_2}\left(s_2'^{(i)}\right) = P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_2}\left(s_2'^{(i-1)}\mu_i^2\sigma_i\right) \\ &= P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_2}\left(s_2'^{(i-1)}\right)\mu_i^2 P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_2}\sigma_i \\ &= s_2^{(i-1)} P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_2}\mu_i^2\varepsilon = s_2^{(i-1)}\mu_i^2 \end{aligned} \tag{5.6}$$

De (5.5), juntamente com a Equação (5.6), pode-se dizer que $s_2^{(i)} \in \overline{S_2}$. Além disso, $P_{(\Sigma_1 \cup \Sigma_c)\to\Sigma_c}\left(s_1'^{(i)}\right) = P_{(\Sigma_2 \cup \Sigma_c)\to\Sigma_c}\left(s_2'^{(i)}\right) = \sigma_1....\sigma_i \in \overline{a}$ e $P_{\Sigma_1\to\Sigma_s}(s_1) = P_{\Sigma_2\to\Sigma_s}(s_2)$. Este passo finaliza a indução.

□

Lema 3. *Sejam S_1 e S_2 supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7 e sejam as cadeias a, s_1', s_1, s_2' e s_2 tais que*

$$\begin{aligned} a &= \sigma_1\sigma_2...\sigma_n \in P_{\Sigma_1\to\Sigma_{1c}}(S_1)||P_{\Sigma_2\to\Sigma_{2c}}(S_2) \subseteq \Sigma_c^*; \\ s_1' &= \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^*; \\ s_1 &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1') \in \overline{S_1}; \\ s_2' &= \mu_1^2\sigma_1\mu_2^2\sigma_2...\mu_n^2\sigma_n \subseteq (\Sigma_2 \cup \Sigma_c)^*; \\ s_2 &= P_{(\Sigma_2\cup\Sigma_c)\to\Sigma_2}(s_2') \in \overline{S_2}. \end{aligned}$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_c)^*$ e $\mu_i^2 \in (\Sigma_2 \setminus \Sigma_c)^*$. Além disso, $P_{\Sigma_1\to\Sigma_s}(s_1) = P_{\Sigma_2\to\Sigma_s}(s_2)$. Então existem as cadeias t_1 e t_2 tais que:

$$\begin{aligned} s_1t_1 &\in S_1; \\ s_2t_2 &\in S_2; \\ P_{\Sigma_1\to\Sigma_s}(t_1) &= P_{\Sigma_2\to\Sigma_s}(t_2). \end{aligned}$$

◇

Demonstração. Partindo de

$$s_1 \in \overline{S_1}$$

e

$$s_2 \in \overline{S_2},$$

como $P_{\Sigma_1 \to \Sigma_{1c}}(S_1)$ é uma OP-abstração, então $\exists b_1 \in \Sigma_{1c}^*$ e $\exists t_1 \in \Sigma_1^*$ tal que $P_{\Sigma_1 \to \Sigma_{1c}}(s_1 t_1) = P_{\Sigma_1 \to \Sigma_{1c}}(s_1) b_1$ e $s_1 t_1 \in S_1$. O mesmo raciocínio pode ser aplicado à cadeia s_2, pois $P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$ também é OP-abstração. Ou seja, $\exists b_2 \in \Sigma_{2c}^*$ e $\exists t_2 \in \Sigma_2^*$ tal que $P_{\Sigma_2 \to \Sigma_{2c}}(s_2 t_2) = P_{\Sigma_2 \to \Sigma_{2c}}(s_2) b_2$ e $s_2 t_2 \in S_2$.

Como somente eventos não controláveis podem ser concatenados a s_1 e s_2, escolhe-se $b_1 = b_2 = \varepsilon$ e como consequência, $P_{\Sigma_1 \to \Sigma_{1c}}(t_1) = P_{\Sigma_2 \to \Sigma_{2c}}(t_2) = \varepsilon$, ou seja, $t_1 \in \Sigma_{1u}^*$ e $t_2 \in \Sigma_{2u}^*$. Dessa forma, basta garantir que t_1 e t_2 são tais que $P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_2 \to \Sigma_s}(t_2)$.

Sejam $t_1' \in \Sigma_{1u}^*$ e $t_2' \in \Sigma_{2u}^*$, tais que $t_1' \leq t_1$, $t_2' \leq t_2$, $s_1 t_1 \in S_1$ e $s_2 t_2 \in S_2$. Para mostrar que $P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_2 \to \Sigma_s}(t_2)$ será usada a estratégia de indução. Deseja-se mostrar que para um dado t_1, sempre existirá t_2 tal que $P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_2 \to \Sigma_s}(t_2)$.

Passo Base: Dado $t_1' \leq t_1$, com $|t_1'| = 0$, existe t_2' tal que $t_2' \leq t_2$ e $P_{\Sigma_1 \to (\Sigma_1 \cap \Sigma_2)}(t_1') = P_{\Sigma_2 \to (\Sigma_1 \cap \Sigma_2)}(t_2')$. Neste caso, $t_2' \in [(\Sigma_2 \setminus \Sigma_1) \cap \Sigma_u]^*$

Hipótese: Dado $t_1' \leq t_1$, com $|t_1'| = k$, existe t_2' tal que $t_2' \leq t_2$ e $P_{\Sigma_1 \to \Sigma_s}(t_1') = P_{\Sigma_2 \to \Sigma_s}(t_2')$.

Passo de Indução: Dado que a hipótese é verdadeira, então para $t_1' \leq t_1$, com $|t_1'| = k+1$, existe t_2' tal que $t_2' \leq t_2$ e $P_{\Sigma_1 \to \Sigma_s}(t_1') = P_{\Sigma_2 \to \Sigma_s}(t_2')$.

Seja $t_1' \leq t_1$ com $|t_1'| = k$. Então, por hipótese, $\exists t_2'$ tal que $P_{\Sigma_1 \to \Sigma_s}(t_1') = P_{\Sigma_2 \to \Sigma_s}(t_2')$. Para formar uma cadeia de comprimento $k+1$, deve-se concatenar t_1' com um evento $\mu \in \Sigma_{1u}$. Dois casos são possíveis:

i) $\mu \in [(\Sigma_1 \setminus \Sigma_s) \cap \Sigma_u]$
Se $\mu \notin (\Sigma_u \cap \Sigma_s)$, então μ só faz parte da planta restrita G_{r1} e não tem influência na projeção $P_{\Sigma_1 \to \Sigma_s}(t_1') = P_{\Sigma_2 \to \Sigma_s}(t_2')$, pois $P_{\Sigma_1 \to \Sigma_s}(t_1' \mu) = P_{\Sigma_1 \to \Sigma_s}(t_1')$.

ii) $\mu \in [(\Sigma_1 \cap \Sigma_s) \cap \Sigma_u]$
Se $\mu \in \Sigma_s$, então ele faz parte da planta comum G_c. Como μ é não controlável, nenhum dos supervisores irá desabilitá-lo. Ou seja, $s_1 t_1' \mu \in \overline{S_1}$ e

$s_2 t'_2 \mu \in \overline{S_2}$. Assim, pode-se dizer que $t'_1\mu \leq t_1$ e $t'_2\mu \leq t_2$. Fazendo $t''_1 = t'_1\mu$ e $t''_2 = t'_2\mu$, então $P_{\Sigma_1 \to \Sigma_s}(t''_1) = P_{\Sigma_2 \to \Sigma_s}(t''_2)$.

Aplicando o princípio de indução matemática aos passos base e de indução, é possível afirmar que dado t_1 tal que $s_1 t_1 \in S_1$, então $\exists t_2$ tal que $P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_2 \to \Sigma_s}(t_2)$ e $s_2 t_2 \in S_2$.

□

Depois de apresentados os Lemas 2 e 3, é possível então apresentar a Proposição 5.

Proposição 5. *Sejam S_1 e S_2 supervisores que compõem um Sistema Modular Local Coordenado segundo a Definição 7. Então $P_{\Sigma \to \Sigma_c}(S_1||S_2) = P_{\Sigma_1 \to \Sigma_{1c}}(S_1)||P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$.* ◊

Demonstração. $P_{\Sigma \to \Sigma_c}(S_1||S_2) \subseteq P_{\Sigma_1 \to \Sigma_{1c}}(S_1)||P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$ é sempre verdadeira. A prova da igualdade consiste portanto em mostrar que $P_{\Sigma_1 \to \Sigma_{1c}}(S_1)||P_{\Sigma_2 \to \Sigma_{2c}}(S_2) \subseteq P_{\Sigma \to \Sigma_c}(S_1||S_2)$.

Seja

$$a \in P_{\Sigma_1 \to \Sigma_{1c}}(S_1)||P_{\Sigma_2 \to \Sigma_{2c}}(S_2) \subseteq \Sigma_c^*, \tag{5.7}$$

então $P_{\Sigma_c \to \Sigma_{1c}}(a) \in P_{\Sigma_1 \to \Sigma_{1c}}(S_1)$ e $P_{\Sigma_c \to \Sigma_{2c}}(a) \in P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$.

Escreve-se $a = \sigma_1\sigma_2...\sigma_n$. Seja

$$s'_1 = \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^* \tag{5.8}$$

tal que

$$\begin{aligned} P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}(s'_1) &= s_1 \in \overline{S_1}, \\ P_{\Sigma_1 \to \Sigma_{1c}}(s_1) &= P_{\Sigma_1 \to \Sigma_{1c}}[P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}(s'_1)] = P_{\Sigma_c \to \Sigma_{1c}}(a) \\ P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}(s'_1) &= a \end{aligned}$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ e $i = \{1, ..., n\}$.

Então, pelo Lema 2, para todo s'_1 com as características acima, existe também um $s'_2 = \mu_1^2\sigma_1\mu_2^2\sigma_2...\mu_n^2\sigma_n$ tal que

$$\begin{aligned} P_{(\Sigma_2 \cup \Sigma_c) \to \Sigma_2}(s'_2) &= s_2 \in \overline{S_2}; \\ P_{(\Sigma_2 \cup \Sigma_c) \to \Sigma_c}(s'_2) &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}(s'_1) = a; \\ P_{\Sigma_1 \to \Sigma_s}(s_1) &= P_{\Sigma_2 \to \Sigma_s}(s_2). \end{aligned}$$

Assim, pode-se afirmar, pelo Lema 3, que existem as cadeias t_1 e t_2 tais que

$$s_1 t_1 \in S_1,$$

$$s_2 t_2 \in S_2$$

e

$$P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_2 \to \Sigma_s}(t_2);$$

Dessa forma, $s_1 t_1 || s_2 t_2 \subseteq S_1 || S_2$ e $a \in P_{\Sigma \to \Sigma_c}(s_1 t_1 || s_2 t_2) \subseteq P_{\Sigma \to \Sigma_c}(S_1 || S_2)$. Então,

$$P_{\Sigma_1 \to \Sigma_{1c}}(S_1) || P_{\Sigma_2 \to \Sigma_{2c}}(S_2) = P_{\Sigma \to \Sigma_c}(S_1 || S_2)$$

□

Finalmente, o Teorema 3, que é a extensão do Teorema 2 para lidar com supervisores provenientes da síntese modular local, é introduzido.

Teorema 3. *Sejam S_1 e S_2 supervisores que compõem um Sistema Modular Local Coordenado, conforme a Definição 7. Sejam as projeções naturais $P_{\Sigma_1 \to \Sigma_{1c}} : \Sigma_1^* \to \Sigma_{1c}^*$ e $P_{\Sigma_2 \to \Sigma_{2c}} : \Sigma_2^* \to \Sigma_{2c}^*$, com $\Sigma_{ic} = \Sigma_i \cap \Sigma_c$. Para toda sequência $s \in P_{\Sigma_1 \to \Sigma_{1c}}(S_1) || P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$ e $A = P^{-1}_{\Sigma \to \Sigma_c}(s) \cap (S_1 || S_2)$, então A será controlável em relação a $G = G_1 || G_2$.* ♦

Demonstração. De acordo com a Proposição 5, $P_{\Sigma_1 \to \Sigma_{1c}}(S_1) || P_{\Sigma_2 \to \Sigma_{2c}}(S_2) = P_{\Sigma \to \Sigma_c}(S_1 || S_2)$. Como $P_{\Sigma \to \Sigma_c}(S_1 || S_2)$ é OP-abstração pela Proposição 4, então $P_{\Sigma_1 \to \Sigma_{1c}}(S_1) || P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$ é OP-abstração. Além disso, pelo Teorema 1, $S = S_1 || S_2$ é o supervisor monolítico. Assim, pelo Teorema 2, A será controlável em relação à $G = G_1 || G_2$. □

Em palavras, o Teorema 3 afirma que a linguagem recuperada A, que é composta de todas as cadeias que projetam para uma cadeia s, obtida na composição das abstrações de supervisores modulares locais, é controlável em relação à planta global G. A necessidade de A ser controlável em relação à G será evidenciada no exemplo de aplicação, apresentado ao final do capítulo.

5.2 Múltiplos Supervisores

Os resultados apresentados na seção anterior serão agora estendidos para que sejam aplicáveis em problemas com múltiplos supervisores. Nesse intuito apresenta-se o Lema 4.

Lema 4. *Se G_j, com $j \in J = \{1, ..., m\}$ são sistemas de produção conforme a Definição 6, então $G = \mathop{\|}_{j=1}^{m} G_j$ também será um sistema de produção.* ◇

Demonstração. A demonstração é semelhante àquela do Lema 1. Para mostrar que G é um sistema de produção, basta mostrar que ele apresenta as três características listadas na Definição 5

i) Sejam G_j sistemas de produção compostos de n_j subsistemas e G composto de m sistemas de produção. Então $G_j = \mathop{\|}_{i=1}^{n_j} M_{ji}$ e

$$G = \mathop{\|}_{j=1}^{m} G_j = \mathop{\|}_{j=1}^{m} \left(\mathop{\|}_{i=1}^{n_j} M_{ji} \right)$$

.

Se houver algum subsistema compartilhado entre os sistemas de produção, ou seja, para todos i, j, k e l tais que $\Sigma_{ji} = \Sigma_{kl}$ e $M_{ji} = M_{kl}$, para $j, k \in \{1, ...m\}$, $j \neq k$, $i \in \{1, ..., n_j\}$ e $l \in \{1, ..., n_k\}$, o resultado ainda continua válido, já que nesta situação $M_{ji}||M_{kl} = M_{ji} = M_{kl}$.

ii) Como G_j, com $j \in \{1, ..., m\}$, é sistema de produção, então os subsistemas que o compõem não compartilham eventos. Ou seja, para todo $i, k \in \{1, ..., n_j\}$, com $i \neq k$, então $\Sigma_{ji} \cap \Sigma_{jk} = \emptyset$.

Caso haja subsistemas compartilhados, ou seja, para todo $j, k \in \{1, ..., m\}$ e $i, l \in \{1, ...n_j\}$, com $j \neq k$ e $i \neq l$, tal que $\Sigma_{ji} = \Sigma_{kl}$ e $M_{ji} = M_{kl}$, a característica *ii)* não é violada, já que ela afirma que subsistemas diferentes não podem compartilhar eventos, mas aqui trata-se de subsistemas iguais.

Assim, pode-se dizer que G possui a característica *ii)*.

iii) Como os alfabetos dos subsistemas que compõem G são disjuntos, então a operação de composição paralela não altera as características *iii)a)* e *iii)b)*, pois $P_{\Sigma \to \Sigma_{ji}}(G) = M_{ji}$, para todo $j \in \{1, ..., m\}$ e $i \in \{1, ..., n_j\}$.

□

A Proposição 6, apresentada a seguir, é a extensão da Proposição 4 para lidar com múltiplos supervisores.

Proposição 6. *Sejam S_j, $\forall j \in J$, supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7. Então $P_{\Sigma\rightarrow\Sigma_c}\left(\mathop{\Vert}_{j=1}^{m} S_j\right)$ é OP-abstração.* $\Diamond$

Demonstração. Seja $K = E||G$, com $E = \mathop{\Vert}_{j=1}^{m} E_j$ e $G = \mathop{\Vert}_{j=1}^{m} G_j$. Como os supervisores são não conflitantes (item *3)* da Definição 7), então pelo Teorema 1, $S = \mathop{\Vert}_{j=1}^{m} S_j = Sup\mathcal{C}(K, G)$. Sabe-se que E é especificação de coordenação (item *2)* da Definição 7) e que G é um sistema de produção (Lema 4). Assim, pela Proposição 1, $P_{\Sigma\rightarrow\Sigma_c}(S) = P_{\Sigma\rightarrow\Sigma_c}\left(\mathop{\Vert}_{j=1}^{m} S_j\right)$ é OP-abstração.

□

Para a demonstração da forma mais geral do resultado principal, faz-se necessário também que a igualdade $P_{\Sigma\rightarrow\Sigma_c}\left(\mathop{\Vert}_{j=1}^{m} S_j\right) = \mathop{\Vert}_{j=1}^{m} P_{\Sigma_j\rightarrow\Sigma_{jc}}(S_j)$ seja verdadeira. Esse resultado intermediário é apresentado na Proposição 7. A demonstração desta proposição foi baseada na demonstração da Proposição 3, apresentada no capítulo anterior.

No entanto, algumas modificações tiveram que ser realizadas em relação à demonstração original. Esta modificação foi necessária pois na Proposição 3 duas das condições necessárias para sua aplicação não estão presentes no contexto deste trabalho. A primeira delas é a exigência de que $\bigcup_{j=1}^{m} \Sigma_{E_j} \subseteq \Sigma_r$, onde Σ_r é chamado de alfabeto de eventos relevantes e é composto pelos eventos que são mantidos na projeção natural. Como neste trabalho o alfabeto de eventos relevantes é Σ_c, então $\bigcup_{j=1}^{m} \Sigma_{E_j} \not\subseteq \Sigma_r = \Sigma_c$. A segunda condição não atendida é $S_j = \overline{S_j} \cap \mathcal{L}_m(G)$. Esta condição estabelece que S_j deve ser $\mathcal{L}_m(G)$-fechada, ou seja, que a marcação do autômato que representa o supervisor (e portanto da especificação) seja consistente com a marcação da planta. Isto também não é verdade aqui pois os supervisores podem modificar a marcação da planta.

Os lemas a seguir são as extensões dos Lemas 2 e 3 para o caso de múltiplos supervisores. Sabe-se que $P_{\Sigma\rightarrow\Sigma_c}\left(\mathop{\Vert}_{j=1}^{m} S_j\right) \subseteq \mathop{\Vert}_{j=1}^{m} P_{\Sigma_j\rightarrow\Sigma_{jc}}(S_j)$ é sempre verdadeira. A prova da igualdade consiste portanto em mostrar que a relação $\mathop{\Vert}_{j=1}^{m} P_{\Sigma_j\rightarrow\Sigma_{jc}}(S_j) \subseteq P_{\Sigma\rightarrow\Sigma_c}\left(\mathop{\Vert}_{j=1}^{m} S_j\right)$ também é verdadeira. Partindo então de uma cadeia $a = \sigma_1\sigma_2...\sigma_n \in \mathop{\Vert}_{j=1}^{m} P_{\Sigma_j\rightarrow\Sigma_{jc}}(S_j) \subseteq \Sigma_c^*$,

a ideia geral da demonstração é a de construir iterativamente cadeias s_1, t_1, s_j e t_j, $\forall j \in J$, tais que

$$s_1 t_1 \in S_1,$$

$$s_j t_j \in S_j,$$

e

$$a \in P_{\Sigma \to \Sigma_c}\left(\Big|\Big|_{j=1}^{m} s_j t_j\right) \subseteq P_{\Sigma \to \Sigma_c}\left(\Big|\Big|_{j=1}^{m} S_j\right).$$

Lema 5. *Sejam S_j, $\forall j \in J$, supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7 e sejam as cadeias a, s_1' e s_1 tais que:*

$$\begin{aligned} a &= \sigma_1\sigma_2...\sigma_n \in \Big|\Big|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j) \subseteq \Sigma_c^*; \\ s_1' &= \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^*; \\ s_1 &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}(s_1') \in \overline{S_1}. \end{aligned}$$

Para todo s_1' e s_1 com as características acima, existem também s_j' e s_j tais que:

$$\begin{aligned} P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}(s_j') &= s_j \in \overline{S_j}; \\ P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_c}(s_j') &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}(s_1') = a; \\ P_{\Sigma_1 \to \Sigma_s}(s_1) &= P_{\Sigma_j \to \Sigma_s}(s_j). \end{aligned}$$

◇

Demonstração. Partindo de

$$a \in \Big|\Big|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jr}}(S_j) \subseteq \Sigma_c^*, \tag{5.9}$$

então $P_{\Sigma_c \to \Sigma_{jc}}(a) \in P_{\Sigma_j \to \Sigma_{jc}}(S_j)$.

Como $a = \sigma_1\sigma_2...\sigma_n \in \Sigma_c^*$ e $s_1' = \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^*$, então

tal que

$$P_{\Sigma_1 \to \Sigma_{1c}}(s_1) = P_{\Sigma_1 \to \Sigma_{1c}}[P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}(s_1')] = P_{\Sigma_c \to \Sigma_{1c}}(a).$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ e $i = \{1, ..., n\}$.

Hipótese: Para todo s'_1 com as características acima, existem também

$$s'_j = \mu_1^j \sigma_1 \mu_2^j \sigma_2 ... \mu_n^j \sigma_n \in (\Sigma_j \cup \Sigma_c)^*, \forall j \in (J \setminus \{1\}),$$

tal que

$$\begin{aligned} P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}(s'_j) &= s_j \in \overline{S_j}, \\ P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_c}(s'_j) &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}(s'_1) = a \end{aligned}$$

A estratégia indutiva será utilizada nesta demonstração. Assume-se que para algum $i \geq 1$, $s_1'^{(i-1)} = \mu_1^1 \sigma_1 ... \mu_{i-1}^1 \sigma_{i-1}$, com $\mu_1^1 ... \mu_{i-1}^1$ definido de tal forma que $P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}(s'_1) = a$ e

$$s_1^{(i-1)} = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i-1)}\right) = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}(\mu_1^1 \sigma_1 ... \mu_{i-1}^1 \sigma_{i-1}) \in \overline{S_1},$$

existem, para cada $j \in J$, $s_j'^{(i-1)} = \mu_1^j \sigma_1 ... \mu_{i-1}^j \sigma_{i-1}$, com $\mu_1^j ... \mu_{i-1}^j$ definidos de tal forma que

$$\begin{aligned} P_{(\Sigma_j \cup \Sigma_c) \to (\Sigma_j \cap \Sigma_k)}(\mu_i^j) &= P_{(\Sigma_k \cup \Sigma_c) \to (\Sigma_k \cap \Sigma_j)}(\mu_i^k), \qquad \forall k \in J, j \neq k \\ P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_c}(s'_j) &= a \end{aligned}$$

e

$$s_j^{(i-1)} = P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}\left(s_j'^{(i-1)}\right) = P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}(\mu_1^j \sigma_1 ... \mu_{i-1}^j \sigma_{i-1}) \in \overline{S_j}, \tag{5.10}$$

$$P_{\Sigma_1 \to \Sigma_s}(s_1) = P_{\Sigma_j \to \Sigma_c}(s_j) \tag{5.11}$$

pretende-se mostrar que para qualquer $s_1'^{(i)}$ e $s_1^{(i)}$ com as características desejadas, existem $s_j'^{(i)}$ e $s_j^{(i)}$ tais que as Equações (5.10) e (5.11) são verdadeiras.

Base: para $i = 1$, $s_1'^{(0)} = \varepsilon$, $s_1^{(0)} = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(0)}\right) \in \overline{S_1}$ existem $s_j'^{(0)} = \varepsilon$, tais que $s_j^{(0)} = P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}\left(s_j'^{(0)}\right) \in \overline{S_j}$. Além disso, $P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_c}\left(s_1'^{(0)}\right) = P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_c}\left(s_j'^{(0)}\right) = \varepsilon \in \overline{a}$ e $P_{\Sigma_1 \to \Sigma_s}(s_1) = P_{\Sigma_j \to \Sigma_s}(s_j) = \varepsilon$.

Passo de Indução: Para $2 \leq i \leq n$, se $s_1'^{(i)} = s_1'^{(i-1)} \mu_i^1 \sigma_i = \mu_1^1 \sigma_1 ... \mu_{i-1}^1 \sigma_{i-1} \mu_i^1 \sigma_i$, com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ definidos de tal forma que $s_1^{(i)} = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i)}\right) \in \overline{S_1}$, então

$$\begin{aligned} s_1^{(i)} &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i)}\right) = P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i-1)} \mu_i^1 \sigma_i\right) \\ &= P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(s_1'^{(i-1)}\right) P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\mu_i^1 \sigma_i\right) \\ &= s_1^{(i-1)} P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\mu_i^1 \sigma_i\right) \\ &= s_1^{(i-1)} \mu_i^1 P_{(\Sigma_1 \cup \Sigma_c) \to \Sigma_1}\left(\sigma_i\right) \in \overline{S_1} \end{aligned} \tag{5.12}$$

Pretende-se construir uma cadeia $\mu_i^j \in (\Sigma_j \backslash \Sigma_{jc})^*$ tal que $P_{(\Sigma_1 \backslash \Sigma_{1c}) \to \Sigma_s}(\mu_i^1) = P_{(\Sigma_j \backslash \Sigma_{jc}) \to \Sigma_s}(\mu_i^j)$ e $P_{\Sigma_1 \to \Sigma_s}\left(s_1^{(i-1)}\mu_i^1\right) = P_{\Sigma_j \to \Sigma_s}\left(s_j^{(i-1)}\mu_i^j\right)$, $P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}\left(s_j'^{(i-1)}\mu_i^j\right) \in \overline{S_j}$.

Como Σ_s é a união das interseções dos conjuntos de eventos dos supervisores, o processo de construção de μ_i^j tem que ser feito iterativamente. Cada cadeia μ_i^j tem que ser tal que:

a) $P_{\Sigma_j \to (\Sigma_j \cap \Sigma_k)}(\mu_i^j) = P_{\Sigma_k \to (\Sigma_k \cap \Sigma_j)}(\mu_i^k)$, $\forall k$ tal que $\Sigma_k \cap \Sigma_j \neq \emptyset$;

b) $s_j^{(i-1)}\mu_i^j \in \overline{S_j}$.

Como todos os eventos comuns que não estão em Σ_c nunca são desabilitados, se um destes eventos aparece em um cadeia da linguagem implementada pelo supervisor, isso significa que o evento é possível na planta (no estado correspondente). Então o evento está habilitado no estado correspondente em todos os supervisores que compartilham a planta.

Portanto, a construção de μ_i^j, $\forall j \in J$, é apenas uma questão de encontrar uma ordenação para os eventos tal que as sequências de eventos em Σ_s sejam consistentes entre as cadeias e μ_i^j seja uma continuação válida para a cadeia $s_j^{(i-1)}$.

Como (a) $s_j^{(i-1)} \in \overline{S_j}$; (b) eventos em $P_{(\Sigma_j \backslash \Sigma_{jc}) \to \Sigma_s}(\mu_i^j)$ são não controláveis (o que implica que nunca são desabilitados pelo supervisor S_j), pode-se concluir que:

$$s_j^{(i-1)}\mu_i^j \in \overline{S_j} \tag{5.13}$$

Para mostrar que $s_j'^{(i)} = s_j'^{(i-1)}\mu_i^j\sigma_i$ é tal que $s_j^{(i)} = P_{(\Sigma_j \cup \Sigma_c) \to \Sigma_j}\left(s_j'^{(i)}\right) \in \overline{S_j}$, ou seja, que adicionando o sufixo σ_i à cadeia $s_j^{(i-1)}\mu_i^j$ não vai gerar uma cadeia que não está em $\overline{S_j}$, considera-se os dois casos seguintes, relacionados ao pertencimento de σ_i em Σ_j:

a) $\sigma_i \in \Sigma_j$ (σ_i é um evento de Σ_j) Nesta situação, dois casos ainda podem ser analisados:

   i) $\sigma_i \in E_j$
   Como $\overline{S_j} = E_j || \mathcal{L}(G_j)$, então $P_{\Sigma_j \to \Sigma_{E_j}}\left(s_j^{(i-1)}\mu_i^j\right) \in \mathcal{L}(E_j)$.
   Pelo fato de E_j ser uma especificação de coordenação, se para algum μ_i^j verifica-se a veracidade de $P_{\Sigma_j \to \Sigma_{E_j}}\left(s_j^{(i-1)}\mu_i^j\sigma_i\right) \in \mathcal{L}(E_j)$, então para todo μ_i^j tal que $P_{\Sigma_j \to \Sigma_{E_j}}\left(s_j^{(i-1)}\mu_i^j\right) \in \mathcal{L}(E_j)$, também será verdade que $P_{\Sigma_j \to \Sigma_{E_j}}\left(s_j^{(i-1)}\mu_i^j\sigma_i\right) \in \mathcal{L}(E_j)$ (item ii) da Definição 6). Além disso $E_j \subseteq P_{\Sigma_j \to \Sigma_{E_j}}(S_j)$ e, portanto $s_j^{(i-1)}\mu_i^j\sigma_i \in \overline{S_j}$.

ii) $\sigma_i \notin E_j$

Neste caso, o fato de $\sigma_i \notin E_j$, implica o evento nunca ser desabilitado pelo supervisor. Logo $s_j^{(i-1)}\mu_i^j\sigma_i \in \overline{S_j}$.

b) $\sigma_i \notin \Sigma_j$ (σ_i não é um evento de S_j)

Neste caso,

$$\begin{aligned} s_j^{(i)} &= P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}\left(s_j'^{(i)}\right) = P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}\left(s_j'^{(i-1)}\mu_i^j\sigma_i\right) \\ &= P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}\left(s_j'^{(i-1)}\mu_i^j\right)P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}(\sigma_i) \\ &= s_j^{(i-1)}P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}(\mu_i^j)\varepsilon = s_j^{(i-1)}\mu_i^j. \end{aligned} \tag{5.14}$$

De (5.13) e (5.14), pode-se dizer que $s_j^{(i)} \in \overline{S_j}$, $\forall j \in J$.

Além disso, $P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}\left(s_1'^{(i)}\right) = P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_c}\left(s_j'^{(i)}\right) = \sigma_1...\sigma_i \in \{\overline{a}\}$. Este passo finaliza a indução.

□

Lema 6. *Sejam S_j, $\forall j \in J$, supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7 e sejam as cadeias a, s_1', s_1, s_j' e s_j tais que*

$$\begin{aligned} a &= \sigma_1\sigma_2...\sigma_n \in \Big\|_{j=1}^{m} P_{\Sigma_j\to\Sigma_{jc}}(S_j) \subseteq \Sigma_c^*; \\ s_1' &= \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1\cup\Sigma_c)^*; \\ s_1 &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1') \in \overline{S_1}; \\ s_j' &= \mu_1^j\sigma_1\mu_2^j\sigma_2...\mu_n^j\sigma_n \subseteq (\Sigma_j\cup\Sigma_c)^*; \\ s_j &= P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}(s_j') \in \overline{S_j}. \end{aligned}$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_c)^*$ e $\mu_i^j \in (\Sigma_j \setminus \Sigma_c)^*$. Além disso, $P_{\Sigma_1\to\Sigma_s}(s_1) = P_{\Sigma_j\to\Sigma_s}(s_j)$, $\forall j \in J$. Então existem as cadeias t_1 e t_j, $\forall j \in J$, tais que:

$$\begin{aligned} s_1t_1 &\in S_1; \\ s_jt_j &\in S_j; \\ P_{\Sigma_1\to\Sigma_s}(t_1) &= P_{\Sigma_j\to\Sigma_s}(t_j). \end{aligned}$$

◇

Demonstração. Partindo de

$$s_j \in \overline{S_j}, \qquad \forall j \in J,$$

como $P_{\Sigma_j \to \Sigma_{jc}}(S_j)$ é uma OP-abstração, então $\exists b_j \in \Sigma_{jc}^*$ e $\exists t_j \in \Sigma_j^*$, $\forall j \in J$, tal que $P_{\Sigma_j \to \Sigma_{jc}}(s_j t_j) = P_{\Sigma_j \to \Sigma_{jc}}(s_j) b_j$ e $s_j t_j \in S_j$.

Como somente eventos não controláveis podem ser concatenados a s_j, escolhe-se $b_j = \varepsilon$ e como consequência, $P_{\Sigma_j \to \Sigma_{jc}}(t_j) = \varepsilon$, ou seja, $t_j \in \Sigma_{ju}^*$. Dessa forma, basta garantir que t_j e t_k, $\forall j, k \in J$ tal que $\Sigma_j \cap \Sigma_k \neq \emptyset$ e $j \neq k$, são tais que $P_{\Sigma_j \to \Sigma_s}(t_j) = P_{\Sigma_k \to \Sigma_s}(t_k)$.

Seja $t'_j \in \Sigma_{ju}^*$, tal que $t'_j \leq t_j$ e $s_j t_j \in S_j$, $\forall j \in J$. Utiliza-se a estratégia de indução para mostrar que $P_{\Sigma_j \to \Sigma_s}(t_j) = P_{\Sigma_k \to \Sigma_s}(t_k)$.

Passo Base: Dado $t'_j \leq t_j$, com $|t'_j| = 0$, existe t'_k tal que $t'_k \leq t_k$ e $P_{\Sigma_j \to \Sigma_s}(t'_j) = P_{\Sigma_k \to \Sigma_s}(t'_k)$, $\forall k \in J$ e $j \neq k$. Neste caso, $t'_k \in [(\Sigma_k \setminus \Sigma_j) \cap \Sigma_u]^*$.

Hipótese: Se $t'_j \leq t_j$, com $|t'_j| = l$, existe t'_k tal que $t'_k \leq t_k$ e $P_{\Sigma_j \to \Sigma_s}(t'_j) = P_{\Sigma_k \to \Sigma_s}(t'_k)$.

Passo de Indução: Assumindo que a hipótese é verdadeira, então para $t'_j \leq t_j$, com $|t'_j| = l + 1$, existe t'_k tal que $t'_k \leq t_k$ e $P_{\Sigma_j \to \Sigma_s}(t'_j) = P_{\Sigma_k \to \Sigma_s}(t'_k)$.

Seja $t'_j \leq t_j$ com $|t'_j| = l$. Então, por hipótese, $\exists t'_k$ tal que $P_{\Sigma_j \to \Sigma_s}(t'_j) = P_{\Sigma_k \to \Sigma_s}(t'_k)$. Para formar uma cadeia de comprimento $l + 1$, deve-se concatenar t'_j com um evento $\mu \in \Sigma_{ju}$. Dois casos são possíveis:

i) $\mu \in [(\Sigma_j \setminus \Sigma_s) \cap \Sigma_u]$
Se $\mu \notin (\Sigma_u \cap \Sigma_s)$, então μ só faz parte da planta restrita G_{rj} e não tem influência na projeção $P_{\Sigma_j \to \Sigma_s}(t'_j) = P_{\Sigma_k \to \Sigma_s}(t'_k)$, pois $P_{\Sigma_j \to \Sigma_s}(t'_j \mu) = P_{\Sigma_j \to \Sigma_s}(t'_j)$.

ii) $\mu \in [(\Sigma_j \cap \Sigma_k) \cap \Sigma_u]$
Se $\mu \in \Sigma_s$, então ele faz parte da planta comum $G_{c\{j,k\}}$. Como μ é não controlável, nenhum dos supervisores irão desabilitá-lo. Ou seja, $s_j t'_j \mu \in \overline{S_j}$ e $s_k t'_k \mu \in \overline{S_k}$. Pode-se então dizer que $t'_j \mu \leq t_j$ e $t'_k \mu \leq t_k$. Fazendo $t''_j = t'_j \mu$ e $t''_k = t'_k \mu$, então $P_{\Sigma_j \to \Sigma_s}(t''_j \mu) = P_{\Sigma_k \to \Sigma_s}(t''_k)$.

Aplicando o princípio de indução matemática os passos base e de indução, é possível afirmar que dado t_j tal que $s_j t_j \in S_j$, então $\exists t_k$ tal que $P_{\Sigma_j \to \Sigma_s}(t_j) = P_{\Sigma_k \to \Sigma_s}(t_k)$ e $s_k t_k \in S_k$.

□

Depois de apresentados os Lemas 5 e 6, é possível então apresentar a Proposição 7.

Proposição 7. *Sejam S_j, $\forall j \in J$, supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7. Então* $P_{\Sigma\to\Sigma_c}\left(\mathop{\|}_{j=1}^{m} S_j\right) = \mathop{\|}_{j=1}^{m}\left[P_{\Sigma_j\to\Sigma_{jc}}(S_j)\right]$. ◊

Demonstração. $P_{\Sigma\to\Sigma_c}\left(\mathop{\|}_{j=1}^{m} S_j\right) \subseteq \mathop{\|}_{j=1}^{m} P_{\Sigma_j\to\Sigma_{jc}}(S_j)$ é sempre verdadeira. Para provar a igualdade, é suficiente mostrar que $\mathop{\|}_{j=1}^{m} P_{\Sigma_j\to\Sigma_{jc}}(S_j) \subseteq P_{\Sigma\to\Sigma_c}\left(\mathop{\|}_{j=1}^{m} S_j\right)$.

Seja

$$a \in \mathop{|\!|}_{j=1}^{m} P_{\Sigma_j\to\Sigma_{jc}}(S_j) \subseteq \Sigma_c^*, \tag{5.15}$$

então $P_{\Sigma_j\to\Sigma_{jc}}(a) \in P_{\Sigma_j\to\Sigma_{jc}}(S_j)$. Seja

$$s_1' = \mu_1^1\sigma_1\mu_2^1\sigma_2...\mu_n^1\sigma_n \subseteq (\Sigma_1 \cup \Sigma_c)^* \tag{5.16}$$

tal que

$$\begin{aligned} P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1') &= s_1 \in \overline{S_1}, \\ P_{\Sigma_1\to\Sigma_{1c}}(s_1) &= P_{\Sigma_1\to\Sigma_{1c}}[P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_1}(s_1')] = P_{\Sigma_c\to\Sigma_{1c}}(a) \\ P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}(s_1') &= a \end{aligned}$$

com $\mu_i^1 \in (\Sigma_1 \setminus \Sigma_{1c})^*$ e $i = \{1, ..., n\}$.

Então, pelo Lema 5, para todo s_1' com as características acima, existe também um $s_j' = \mu_1^j\sigma_1\mu_2^j\sigma_2...\mu_n^j\sigma_n$ tal que

$$\begin{aligned} P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_j}(s_j') &= s_j \in \overline{S_j}; \\ P_{(\Sigma_j\cup\Sigma_c)\to\Sigma_c}(s_j') &= P_{(\Sigma_1\cup\Sigma_c)\to\Sigma_c}(s_1') = a; \\ P_{\Sigma_1\to\Sigma_s}(s_1) &= P_{\Sigma_j\to\Sigma_s}(s_j). \end{aligned}$$

Assim, pode-se afirmar, pelo Lema 6, que existem as cadeias t_1 e t_j, $\forall j \in J$, tais que

$$s_1t_1 \in S_1,$$

$$s_jt_j \in S_j$$

e

$$P_{\Sigma_1 \to \Sigma_s}(t_1) = P_{\Sigma_j \to \Sigma_s}(t_j);$$

Assim, $\|_{j=1}^{m} s_j t_j \subseteq \|_{j=1}^{m} S_j$ e $a \in P_{\Sigma \to \Sigma_c}\left(\|_{j=1}^{m} s_j t_j\right) \subseteq P_{\Sigma \to \Sigma_c}\left(\|_{j=1}^{m} S_j\right)$. Então

$$\|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j) = P_{\Sigma \to \Sigma_c}\left(\|_{j=1}^{m} S_j\right)$$

□

Finalmente, apresenta-se o Teorema 4, que é extensão do Teorema 3 para lidar com um número de supervisores arbitrário. O Teorema 4 é a forma mais geral do resultado principal.

Teorema 4. *Sejam S_j, $\forall j \in J$, supervisores que compõem um Sistema Modular Local Coordenado conforme a Definição 7. Sejam as projeções naturais $P_{\Sigma_j \to \Sigma_{jc}} : \Sigma_j^* \to \Sigma_{jc}^*$, com $\Sigma_{jc} = \Sigma_j \cap \Sigma_c$. Para toda sequência $s \in \|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j)$ e $A = P^{-1}_{\Sigma \to \Sigma_c}(s) \cap \left(\|_{j=1}^{m} S_j\right)$, então A será controlável em relação a $G = \|_{j=1}^{m} G_j$.*

♦

Demonstração. Pela Proposição 7, $\|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j) = P_{\Sigma \to \Sigma_c}\left(\|_{j=1}^{m} S_j\right)$. Como $P_{\Sigma \to \Sigma_c}\left(\|_{j=1}^{m} S_j\right)$ é OP-abstração pela Proposição 4, então $\|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j)$ é OP-abstração. Além disso, pelo Teorema 1, $S = \|_{j=1}^{m} S_j$ é o supervisor monolítico. Assim, pelo Teorema 2, A será controlável em relação à $G = \|_{j=1}^{m} G_j$. □

Em palavras, o Teorema 4 diz que a linguagem recuperada A, que é composta por todas as cadeias que projetam para uma cadeia s, obtida na composição das abstrações de supervisores modulares locais, é controlável em relação à planta global G.

Como resultado adicional, o Lema 7, apresentado no Apêndice A, diz que se E_j, $\forall j \in J$ são especificações de coordenação e $\Sigma_{E_j} \cap \Sigma_{E_k} \neq \emptyset$, com $j, k \in J$ e $j \neq k$, então

$E = \|_{j=1}^{m} E_j$ será especificação de coordenação. Neste caso, ao verificar se a modelagem do sistema atende à Definição 7, não há a necessidade de se verificar se $E = \|_{j=1}^{m} E_j$ é especificação de coordenação.

5.3 Resultados Complementares

Nesta seção serão apresentados uma definição e um corolário, que juntos, tornam a aplicação dos Teoremas 3 e 4 mais eficientes.

A composição paralela da abstração dos supervisores, por ser um autômato cíclico, contém todas as sequências de eventos controláveis que levam à produção de lotes de produtos de todos os tamanhos. No entanto, durante o processo de planejamento, não é interessante comparar cadeias que levam à produção de lotes de tamanhos diferentes. É necessário então selecionar todas as cadeias que levam à produção de um lote de produtos de determinado tamanho. Com este objetivo apresenta-se a Definição 8.

Definição 8 (Especificação de quantidade)**.** *Um autômato E_q é uma especificação de quantidade, para a produção de um lote de tamanho de k produtos, se apresenta as seguintes características:*

1) $\Sigma_{E_q} = \{\sigma\}$*, em que σ é o último evento controlável da sequência de eventos que levam à produção de 1 produto;*

2) $\mathcal{L}_m(E_q) = v \subseteq \Sigma_{Eq}^*$*, onde* $|v| = k$*.*

□

Apresenta-se a seguir o Corolário 1, que afirma que qualquer cadeia s obtida num subconjunto de $\|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j)$ também resultará em um linguagem A controlável em relação à G.

Corolário 1. *Seja um Sistema Modular Local Coordenado segundo a Definição 7, seja uma especificação de quantidade segundo a Definição 8. Seja* $s \in E_q || \left[\|_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j) \right]$*, então* $A = P^{-1}_{\Sigma_j \to \Sigma_{jc}}(s) \cap \left(\|_{j=1}^{m} S_j \right)$ *é controlável.* ∘

Demonstração. A especificação de quantidade, quando composta com a composição paralela das abstrações dos supervisores, resulta num autômato cuja linguagem marcada é composta de todas as cadeias de eventos controláveis que levam à produção de um lote de tamanho k. Como $\Sigma_{E_q} \subset \Sigma$, então

$$\mathcal{L}_m(E_q)||\left[\mathop{||}_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j)\right] \subset \mathop{||}_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j).$$

Ou seja, pelo Teorema 4, se A é controlável para toda cadeia s então A obtido a partir de $s \in \mathcal{L}_m(E_q)||\left[\mathop{\|}_{j=1}^{m} P_{\Sigma_j \to \Sigma_{jc}}(S_j)\right]$ também será controlável.

□

A próxima seção mostra um exemplo de aplicação dos resultados apresentados.

5.4 Exemplo de Aplicação

Nesta seção será apresentado um exemplo de como aplicar a abstração em supervisores obtidos pela síntese modular local.

Exemplo 1. *Seja um sistema de manufatura composto de 3 máquinas e 2 buffers, como mostrado na Figura 7. Para que um insumo bruto se torne um produto acabado, ele deve passar, nessa ordem, pelas máquinas M_1, M_2 e M_3. Os buffers tem capacidade unitária e os eventos α_i são controláveis, enquanto os eventos β_i são não controláveis. Como restrição de segurança, deve-se garantir que não haja a ocorrência de underflow ou overflow nos buffers.*

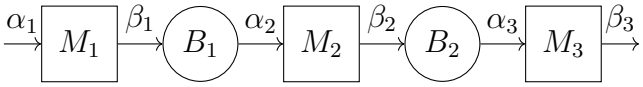


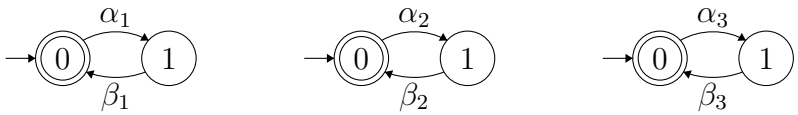
Figura 7 – Sistema de manufatura do Exemplo 1

.

Suponha que deseja-se realizar a produção de um lote de produto de tamanho 2 no menor tempo possível. Na solução desse problema, será utilizada a abstração proposta nesse trabalho.

Para que seja possível aplicar a abstração, inicialmente é preciso modelar o sistema e especificações como autômatos, de maneira a atender a Definição 7. Cada uma das máquinas será modelada como tendo apenas dois estados: ociosa e trabalhando, como mostrado

na Figura 8. Observa-se também que esse modelo está em conformidade com o item 1) da Definição 7, ou seja, os modelos são sistemas de produção.



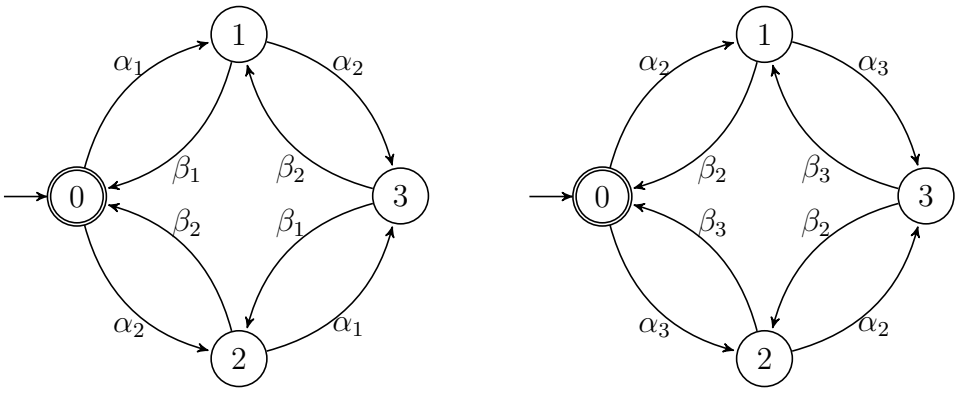
(a) Modelo da Máquina M_1 (b) Modelo da Máquina M_2 (c) Modelo da Máquina M_3

Figura 8 – Modelo das máquinas

A restrição de underflow e overflow nos buffers é modelada pelos autômatos da Figura 9. Observe que as mesmas especificações também garantem a ordem correta de produção. Para que estes estejam de acordo com o item 2) da Definição 7, é necessário verificar se as especificações são controláveis em relação à planta. Neste caso a planta é a planta local de cada especificação, que é obtida pela composição paralela de todos os autômatos que tem eventos em comum com a especificação. Assim, obtém-se as plantas locais G_1 e G_2, mostradas na Figura 10.

(a) E_1: Especificação do *buffer* B_1 (b) E_2: Especificação do *buffer* B_2

Figura 9 – Modelos das Especificações

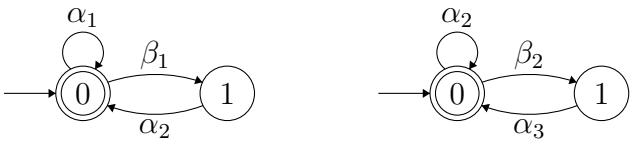


(a) Planta local G_1 (b) Planta local G_2

Figura 10 – Plantas locais

Para verificar se as especificações são controláveis, basta verificar se a igualdade $S_i = K_i$, com $S_i = Sup\mathcal{C}(K_i, G_i)$ e $K_i = E_i||G_i$, é verdadeira. Ao realizar essa verificação, percebe-se que a igualdade não é verdadeira. Obtém-se então os supervisores reduzidos,

como discutido na Seção 5.1, mostrados na Figura 11. As especificações E_1 e E_2 serão então substituídas pelos supervisores reduzidos, $S_{1,red}$ e $S_{2,red}$, respectivamente, dando origem às especificações E_1' e E_2', que são especificações de coordenação, segundo a Definição 6. Além disso, a especificação E', obtida pela composição de E_1' e E_2', também é especificação de coordenação, como pode ser visto na Figura 12.



(a) E_1': Supervisor Reduzido $S_{1,red}$ (b) E_2': Supervisor reduzido $S_{2,red}$

Figura 11 – Supervisores reduzidos

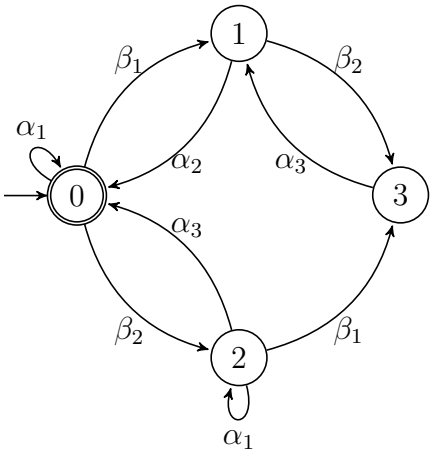
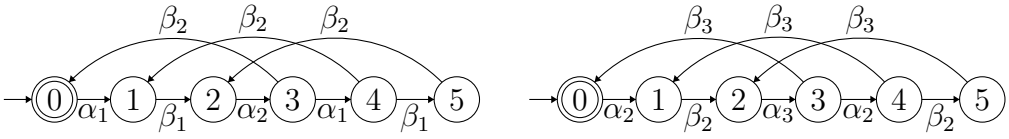


Figura 12 – Composição paralela das especificações controláveis E_1' e E_2'

O próximo passo é então obter os supervisores S_1 e S_2, com $S_i = E_i'||G_i$, que são não conflitantes. Eles são exibidos na Figura 13. Esses supervisores compõem um Sistema Modular Local Coordenado, o que permite a aplicação do resultado principal deste trabalho.



(a) Supervisor S_1 (b) Supervisor S_2

Figura 13 – Supervisores

Aplica-se agora a abstração em cada um dos supervisores obtidos. O resultado é mostrado na Figura 14.

(a) $P_{\Sigma_1 \to \Sigma_{1c}}(S_1)$ (b) $P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$

Figura 14 – Projeção natural dos supervisores para o alfabeto de eventos controláveis

Para explicitar as sequências de eventos que levam à produção de um lote de tamanho 2, utiliza-se uma especificação de quantidade conforme a Definição 8, mostrada na Figura 15. Faz-se então a operação $Trim$ sobre o resultado da composição paralela das abstrações dos supervisores e da especificação de quantidade. O resultado obtido é mostrado na Figura 16.

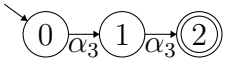


Figura 15 – Especificação de quantidade para a produção de 2 produtos

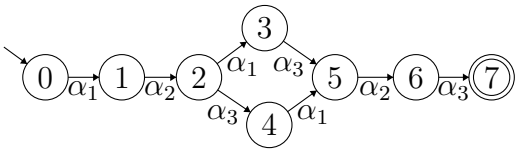


Figura 16 – Composição paralela entre $P_{\Sigma_1 \to \Sigma_{1c}}(S_1)$, $P_{\Sigma_2 \to \Sigma_{2c}}(S_2)$ e a especificação de quantidade para 2 produtos

O autômato da Figura 16 explicita todas as sequências de eventos controláveis que levam a produção de dois produtos. Nesse caso, tem-se as sequências

$$s_1 = \alpha_1\alpha_2\alpha_3\alpha_1\alpha_2\alpha_3$$

$$s_2 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\alpha_3$$

Esse autômato será o modelo do sistema, que é uma das entradas do Planejador da Figura 2. O Teorema 4 será aplicado sobre esse autômato. Suponha que o Planejador tenha retornado como plano ótimo s_{ot} a sequência s_2. Assim,

$$s_{ot} = s_2 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\alpha_3$$

O Teorema 4 diz que a linguagem A, que é uma sublinguagem do comportamento em malha fechada que projeta para a sequência s_{ot}, é controlável. Ou seja, ao adicionar os eventos não controláveis, a nova sequência será factível na planta. Isso é verdade para qualquer sequência que o Planejador possa escolher. A Figura 17 apresenta a associação

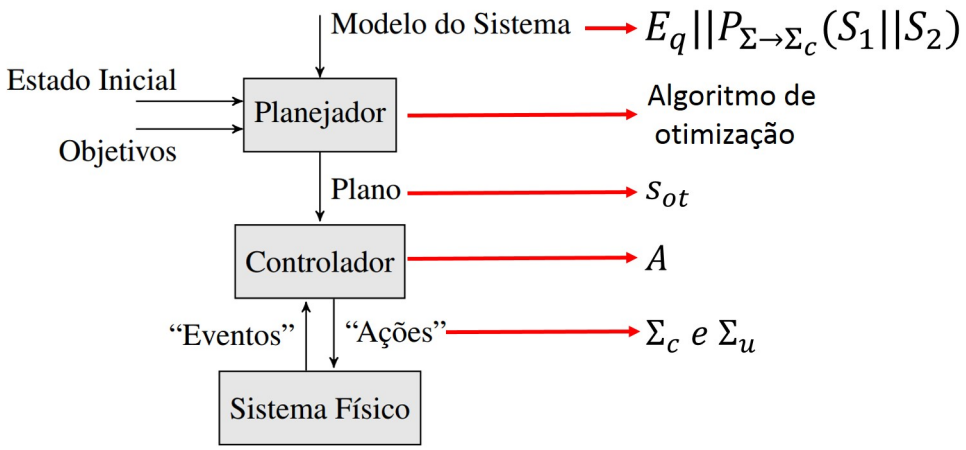


Figura 17 – Associação do modelo conceitual para a solução do problema de planejamento com os conceitos apresentados no Teorema 4

.

entre o modelo conceitual apresentado no Capítulo 2 e os conceitos envolvidos no Teorema 4.

Constrói-se $A = P^{-1}_{\Sigma\to\Sigma_c}(s_{ot}) \cap (S_1||S_2)$*, representada pelo autômato mostrado na Figura 18.*

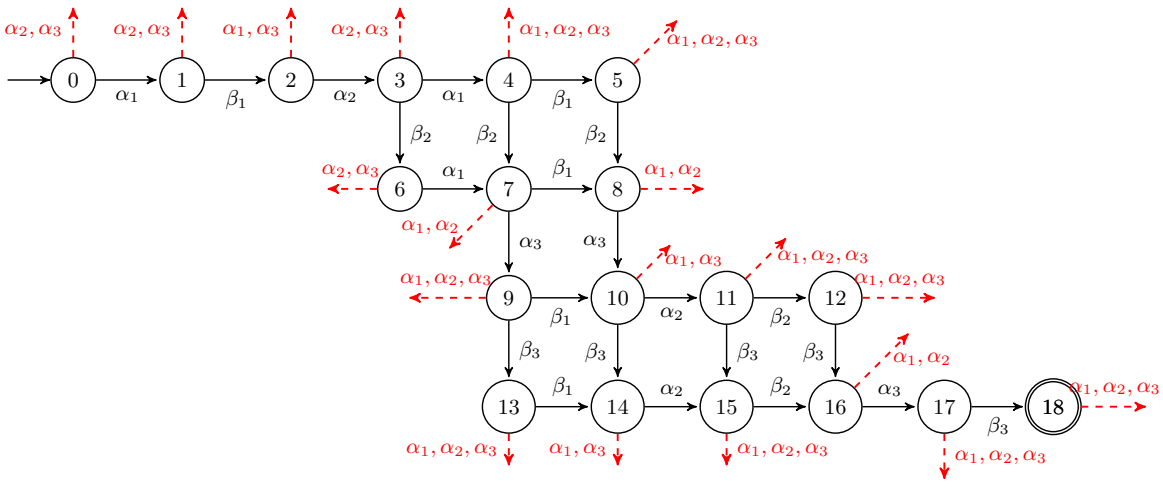


Figura 18 – Autômato que implementa A.

A linguagem A *corresponderá ao comportamento do Controlador, na Figura 17, sendo um subconjunto do comportamento dos supervisores que respeita as restrições e leva à produção de um lote de tamanho 2. O Controlador deve garantir que a execução do Sistema Físico seja restrita a uma das cadeias da linguagem marcada de* A*. Para que isso seja possível, o Controlador deve desabilitar, em cada estado do Sistema Físico, um conjunto de eventos que o levariam a sair de* $\mathcal{L}_m(A)$*. Como o Controlador só pode atuar sobre eventos controláveis, é essencial garantir que todos os eventos que ele deve desabilitar sejam con-*

troláveis. Isso é equivalente à linguagem A ser controlável em relação ao Sistema Físico G.

As setas tracejadas da Figura 18 representam as desabilitações de eventos. Como pode ser observado, somente eventos controláveis são desabilitados, mostrando assim a controlabilidade de A em relação à planta.

△

Para fins de comparação, o mesmo problema do Exemplo 1 será resolvido com a aplicação do Teorema 2 (VILELA; PENA, 2016). Inicialmente obtém-se a planta global $G = M_1||M_2||M_3$ e a especificação global $E = E_1||E_2$, mostrada na Figura 19. Pelas características do problema, é fácil concluir que G é sistema de produção. No entanto, a especificação E não pode ser considerada especificação de coordenação, pois não é controlável em relação à planta. Aplicando o procedimento já descrito, obtém-se uma nova especificação E', que é idêntica àquela mostrada na Figura 12, que é uma especificação de coordenação.

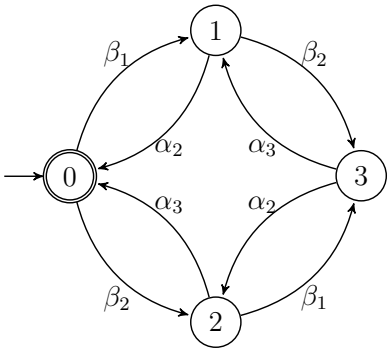


Figura 19 – Especificação global E

Calcula-se então o supervisor monolítico, que é um autômato de 18 estados e 32 transições, muito grande para ser mostrado com clareza. Em seguida, aplica-se a abstração sobre o supervisor, resultando no autômato mostrado na Figura 20. Este autômato é idêntico à composição paralela entre os autômatos da Figura 14 e por isso os próximos passos são idênticos aos apresentados no Exemplo 1, motivo pelo qual serão omitidos.

É importante ressaltar que o resultado final é idêntico usando as duas abordagens. A diferença reside no tamanho dos autômatos intermediários envolvidos, sendo que eles são maiores na abordagem monolítica. A Tabela 2 mostra o tamanho dos autômatos intermediários e finais. Na abordagem monolítica, o maior autômato é a linguagem desejada K, com 32 estados e 64 transições. Na abordagem modular local, a linguagem desejada K_i é com-

posta de dois autômatos de 8 estados e 12 transições. Essa linguagem desejada K_i é obtida pela composição da planta local pela especificação não controlável, que é o passo anterior à obtenção da especificação controlável. Finalmente, pode-se perceber que a abstração do supervisor monolítico tem o mesmo tamanho da composição paralela das abstrações dos supervisores modulares locais. De fato, elas são iguais.

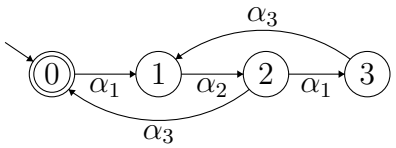


Figura 20 – $P_{\Sigma \to \Sigma_c}(S)$

Tabela 2 – Tamanho (x, y) dos autômatos envolvidos

	Autômato	Tamanho do autômato (x, y)
Abordagem Monolítica	G	8, 24
	E	4, 8
	K	32, 64
	S	18, 32
	$P_{\Sigma \to \Sigma_c}(S)$	4, 5
Abordagem Modular Local	G_i	4, 8
	E_i	2, 3
	K_i	8, 12
	S_i	6, 8
	$P_{\Sigma_i \to \Sigma_{ic}}(S_i)$	2, 2
	$\Vert_{i=1}^{n} P_{\Sigma_i \to \Sigma_{ic}}(S_i)$	4, 5

[a] O tamanho dos autômatos nas tabelas é representado pelo par (x, y), onde x é o número de estados e y o número de transições.

A Tabela 3 mostra o tamanho dos autômatos quando se tem o aumento da complexidade do problema. Esse aumento de complexidade consiste em aumentar o número de máquinas no sistema mostrado na Figura 7, mantendo a característica de que as máquinas são dispostas de forma sequencial e que sempre há um *buffer* de capacidade unitária entre duas máquinas. Percebe-se que o tamanho dos autômatos na abordagem modular local não aumenta. O que cresce é o número de autômatos, sendo que tem-se $m - 1$ supervisores, onde m é o número de máquinas. Por outro lado, o tamanho do supervisor cresce de forma exponencial na abordagem monolítica.

Compara-se agora o utilização do supervisor e de sua abstração como modelo do sistema utilizado pelo Planejador da Figura 2. A Tabela 4 apresenta o número de sequências

Tabela 3 – Tamanho (x, y) dos autômatos em função do número de máquinas

Número de máquinas (m)	Abordagem Monolítica			Abordagem Modular Local		
	K	S	$P_{\Sigma\to\Sigma_c}(S)$	K_i	S_i	$P_{\Sigma_i\to\Sigma_{ic}}(S_i)$
3	32, 64	18, 32	4, 5	8, 12	6, 8	2, 2
4	128, 320	54, 120	8, 12	8, 12	6, 8	2, 2
5	512, 1536	162, 432	16, 28	8, 12	6, 8	2, 2
6	2048, 7168	486, 1512	32, 64	8, 12	6, 8	2, 2
7	8192, 32768	1458, 5184	64, 144	8, 12	6, 8	2, 2

factíveis para diferentes números de máquinas e quantidade de produtos. Para obtenção dos dados foi utilizado o algoritmo apresentado no Apêndice B. Para cada caso o autômato G foi obtido após a operação $Trim$ no autômato resultante da composição paralela entre o supervisor ou sua abstração com a especificação de quantidade para diferentes quantidades de produtos. Para o cálculo do supervisor, sua abstração e para a implementação do Algoritmo 1, foi utilizada a ferramenta computacional UltraDES (ALVES; MARTINS, 2017).

Tabela 4 – Número de sequências factíveis

Número de Máquinas (m)	Número de produtos (n)	Número de sequências factíveis no supervisor	Número de sequências factíveis na abstração do supervisor
3	2	22	2
	3	640	4
	4	18.760	8
	5	550.000	16
	6	16.124.800	32
4	2	211	5
	3	104.861	29
	4	60.373.651	169
	5	35.320.468.661	985
	6	20.568.776.097.611	5741
5	2	2306	214
	3	25.702.802	290
	4	444.855.492.680	6392
	5	$8{,}52 \times 10^{15}$	141.696
	6	$1{,}66 \times 10^{20}$	3.142.704

Em seguida, foi realizada a contagem das sequências factíveis, ou seja, sequências que levam o sistema do estado inicial ao estado marcado, respeitando as restrições de segurança e não bloqueio. O autômato da Figura 16 corresponde à abstração do supervisor para o caso de 3 máquinas e 2 produtos, em que há apenas 2 sequências factíveis.

É possível notar a explosão do número de sequências à medida em que se aumenta

o problema, quando o supervisor sem abstração é utilizado. Esse número elevado, que é o tamanho do espaço de busca, dificulta o processo de otimização. Mesmo com a aplicação da abstração ainda pode ser inviável avaliar todas as soluções possíveis, mas ela reduz de maneira importante o tamanho do espaço de busca.

5.5 Discussão

Neste trabalho, optou-se por apresentar inicialmente o teorema que estabelece as condições para aplicação da abstração em supervisores modulares locais para o caso de dois supervisores. Posteriormente, apresentou-se o segundo teorema, estendendo o resultado para múltiplos supervisores. Esta abordagem foi adotada por facilitar tanto a elaboração dos resultados quanto a compreensão dos mesmos.

As condições impostas sobre a modelagem do sistema físico para que os resultados apresentados possam ser aplicados podem parecer a princípio uma dificuldade à aplicação das abstrações. No entanto, a modelagem dos sistemas físicos como sistema de produção e as especificações como especificação de coordenação acontece de forma natural. A maior dificuldade pode residir em obter uma especificação controlável. Nesse caso pode-se aplicar o procedimento já descrito, que é o de utilizar o supervisor reduzido como especificação controlável.

Ademais, em vários dos resultados apresentados faz-se o uso da hipótese de que $E = \|_{j=1}^{m} E_j$ é uma especificação de coordenação, quando E_j são especificações de coordenação. A preservação dessa característica após a composição paralela foi observada em diversos casos. Acredita-se que sob certas condições, menos restritivas que $\Sigma_{E_j} \cap \Sigma_{E_k} = \emptyset, \forall j, k \in J$, $j \neq k$, é possível garantir que a composição paralela de especificações de coordenação dão origem a uma especificação coordenação.

Assume-se também que os supervisores modulares locais são sempre não conflitantes. Isso porque a resolução do conflito não é trivial e vários outros trabalhos já trataram desse assunto. A solução mais simples para eliminar o conflito é a utilização da abordagem monolítica sobre o conjunto de especificações que resultam em supervisores conflitantes.

É importante ressaltar que as sequências obtidas na abstração, são as que respeitam os requisitos de segurança e não bloqueio, impostas pelas especificações. Ou seja, pode-se dizer que estas sequências são logicamente factíveis. No entanto, num sistema real, cada equipamento terá um tempo próprio para a realização de determinada tarefa. Levando em consideração esse tempo, nem todas as cadeias logicamente factíveis serão também tempo-

ralmente factíveis. Se os tempos que as máquinas levam para a realização de uma tarefa são determinísticos, então há uma redução significativa do número de sequências temporalmente factíveis em relação às que são logicamente factíveis (dependendo dos tempos de operação das máquinas, pode-se ter mais de uma sequência que tem o mesmo *makespan*). Isso significa que o autômato da Figura 18, que é a linguagem recuperada da cadeia $s_{ot} = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\alpha_3$, apesar de ter 17 sequências logicamente factíveis, se tiver o tempo das operações levado em consideração, passará a ter apenas algumas poucas cadeias temporalmente factíveis.

Essa distinção entre as cadeias logicamente factíveis daquelas que são temporalmente factíveis é realizada pelo Planejador da Figura 2, durante o processo de otimização, como acontece em (PENA *et al.*, 2016). Para avaliar o *makespan* de uma determinada sequência obtida na abstração, o Planejador interage com um simulador que, para cada evento controlável, retorna o evento não controlável correspondente depois de um determinado intervalo de tempo, que é uma característica do processo.

Como pode ser observado na Tabela 4, a redução do tamanho do espaço de busca quando se compara o supervisor com sua abstração é de no mínimo 10 vezes, podendo chegar à ordem de grandeza de 10^{14}, para o caso de 5 máquinas e 6 produtos. Esta diferença se dá pelo fato de que, para cada sequência logicamente factível na abstração, há no supervisor inúmeras cadeias com a mesma ordem de eventos controláveis que diferem apenas na ordem em que os eventos não controláveis aparecem.

Outro ponto interessante a ser observado é que a sequência obtida a partir das abstrações dos supervisores modulares locais, como mostrado no Exemplo 1, foi retirada a partir da composição paralela das mesmas. O resultado dessa operação é uma abstração que é igual aquela obtida pela abordagem monolítica. Na verdade, a composição paralela das abstrações não é obrigatória, mas essa operação expressa o fato de que as sequências obtidas em cada uma das abstrações devem ser sincronizadas em relação aos eventos compartilhados. Ao realizar a composição paralela, automaticamente se garante que qualquer cadeia obtida na abstração será sincronizada.

Finalmente, a Tabela 3 mostra o aumento da complexidade dos autômatos envolvidos na abordagem monolítica ao passo que, na abordagem modular local, a complexidade se mantém constante. Além disso, o cálculo da abstração, devido à presença da propriedade do observador, possui complexidade polinomial no número de estados do supervisor (WONG, 1998). Estes fatos evidenciam a grande vantagem da utilização dessa segunda abordagem sobre a primeira.

6 Conclusões

Este trabalho introduz as condições necessárias para aplicação de abstrações sobre supervisores obtidos pela síntese modular local. O principal propósito dessa abstração é o de reduzir o espaço de busca da solução de problemas de planejamento. A abstração consiste na projeção natural, com a propriedade do observador, de cada um dos supervisores para o respectivo alfabeto de eventos controláveis. O resultado principal garante que qualquer cadeia obtida nas abstrações, desde que esteja sincronizada nos eventos compartilhados, será executável no sistema físico, independentemente da ordem em que os eventos não controláveis apareçam.

A extensão proposta neste trabalho se justifica pelo fato de que quando o problema é muito grande e o supervisor monolítico não pode ser computado, uma das técnicas que realizam a decomposição do problema é a abordagem do controle modular local. Assim, é legítimo propor a utilização das abstrações sobre os supervisores modulares locais. Enquanto o tamanho do supervisor monolítico cresce exponencialmente com o tamanho do problema, os supervisores modulares locais tem tamanhos aproximadamente constantes.

Deste modo, técnicas para a solução do problema de planejamento que se utilizam do supervisor obtido pela TCS, podem apresentar um ganho de eficiência com a utilização de supervisores modulares locais e suas abstrações. Além da redução de complexidade para o cálculo dos supervisores, os dados experimentais para o exemplo apresentado mostram que as abstrações reduzem no mínimo 10 vezes o tamanho do espaço de busca.

As abstrações apresentadas neste trabalho, no entanto, não são a solução final para o problema de planejamento, mas sim uma ferramenta, pois as abstrações em si não trazem informações que permitem quantificar o quão boa uma solução é. Faz-se necessário então, informações sobre o tempo de operação de cada máquina, ou alguma outra informação de custo.

Como proposta de continuação a este trabalho está o desenvolvimento de técnicas para a solução do problema de planejamento que se utilizem das vantagens apresentadas pela modelagem por sistemas a eventos discretos, da síntese modular local e de suas abstrações.

Referências

ABDEDDAÏM, Y.; MALER, O. Job-Shop Scheduling using Timed Automata. *CAV 01: Proc. of 13th Conf. on Computer Aided Verification*, v. 2102, p. 478–492, 2001. Citado na página 33.

AHANI, G.; ASYABANI, M. A Tabu Search Algorithm for No-Wait Job Shop Scheduling Problem. *International Journal of Operational Research*, v. 19, n. 2, p. 246, 2014. Citado na página 30.

ALVES, L. V. R. *et al.* Planning on Discrete Events Systems: A logical approach. *IEEE International Conference on Automation Science and Engineering*, v. 2016-Novem, p. 1055–1060, 2016. Citado na página 35.

ALVES, L. V. R.; MARTINS, L. R. R. UltraDES - A Library for Modeling , Analysis and Control of Discrete Event Systems. *Proceedings of the 20th World Congress of the International Federation of Automatic Control*, p. 5996–6001, 2017. Citado na página 82.

ALVES, M. R. C.; PENA, P. N. Abstrações de Supervisores Localmente Modulares para Aplicação na Solução de Problemas de Planejamento. *XIII Simpósio Brasileiro de Automação Inteligente*, Porto Alegre - RS, p. 699–704, 2017. Citado na página 36.

ARISHA, A.; YOUNG, P.; BARADIE, M. E. Job Shop Scheduling Problem: an Overview. *International Conference for Flexible Automation and Intelligent Manufacturing (FAIM 01)*, p. 682–693, 2001. Citado 4 vezes nas páginas 25, 27, 29 e 30.

BAKER, K. R.; TRIETSCH, D. *Principles of Sequencing and Scheduling*. 1st. ed. New Jersey: John Wiley & Sons, 2009. 1–493 p. Citado 2 vezes nas páginas 25 e 29.

BHATT, N.; CHAUHAN, N. R. Genetic Algorithm Applications on Job Shop Scheduling Problem: A Review. *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, p. 7–14, 2015. Citado na página 30.

BILKAY, O.; ANLAGAN, O.; KILIC, S. E. Job Shop Scheduling Using Fuzzy Logic. *The International Journal of Advanced Manufacturing Technology*, v. 23, n. 7-8, p. 606–619, 2004. Citado na página 30.

BÜRGY, R. A Neighborhood for Complex Job Shop Scheduling Problems with Regular Objectives. *Journal of Scheduling*, Springer US, v. 20, n. 4, p. 391–422, 2017. Citado na página 30.

CASSANDRAS, C.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. 2nd. ed. New York: Springer, 2007. v. 11. 776 p. Citado 4 vezes nas páginas 37, 38, 40 e 43.

CUNHA, A. E. C.; CURY, J. E. R. Hierarchical Supervisory Control Based on Discrete Event Systems with Flexible Marking. *IEEE Transactions on Automatic Control*, v. 52, n. 12, p. 2242–2253, 2007. Citado na página 32.

CURY, J. E. R.; TORRICO, C. R. C.; CUNHA, A. E. C. Supervisory Control of Discrete Event Systems with Flexible Marking. *European Journal of Control*, v. 10, p. 47–60, 2004. Citado na página 32.

FABRE, E.; JEZEQUEL, L. Distributed Optimal Planning: An Approach by Weighted Automata Calculus. *Proceedings of the IEEE Conference on Decision and Control*, n. section III, p. 211–216, 2009. Citado na página 35.

FLORDAL, H.; MALIK, R. Modular Nonblocking Verification using Conflict Equivalence. *Proceedings of the 8th International Workshop on Discrete Event Systems, WODES 2006*, p. 100–106, 2006. Citado na página 48.

GHALLAB, M.; NAU, D.; TRAVERSO, P. *Automated Planning - Theory and Practice*. [S.l.]: Elsevier, 2004. Citado 3 vezes nas páginas 27, 28 e 29.

HILL, R. C.; LAFORTUNE, S. Planning Under Abstraction within a Supervisory Control Context. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, n. Cdc, p. 4770–4777, 2016. Citado 2 vezes nas páginas 25 e 36.

HILL, R. C.; TILBURY, D. M.; LAFORTUNE, S. Modular Supervisory Control with Equivalence-Based Conflict Resolution. *2008 American Control Conference*, p. 491–498, 2008. Citado 2 vezes nas páginas 32 e 48.

HU, S. J. Evolving Paradigms of Manufacturing: From Mass Production to Mass Customization and Personalization. *Procedia CIRP*, Elsevier B.V., v. 7, p. 3–8, 2013. Citado na página 25.

JEZEQUEL, L.; FABRE, E. Turbo Planning. *11th International Workshop on Discrete Event Systems, WODES 2012*, IFAC, v. 45, n. 29, p. 301–306, 2012. Citado na página 35.

KOBETSKI, A.; FABIAN, M. Scheduling of Discrete Event Systems using Mixed Integer Linear Programming. *Proceedings of the 8th International Workshop on Discrete Event Systems*, p. 76–81, 2006. Citado na página 34.

KOBETSKI, A. *et al.* Minimization of Expected Cycle Time in Manufacturing Cells with Uncontrollable Behavior. *Proceedings of the 3rd IEEE International Conference on Automation Science and Engineering, IEEE CASE 2007*, p. 14–19, 2007. Citado na página 35.

KU, W.-Y.; BECK, J. C. Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis. *Computers & Operations Research*, v. 73, p. 165–173, 2016. Citado na página 30.

LI, Y.; CHEN, Y. A Genetic Algorithm for Job-Shop Scheduling. *Journal of Software*, v. 5, n. 3, p. 269–274, 2010. Citado na página 30.

MAHMOOD, B.; BASHIR, A. Approach to Job-Shop Scheduling Problem using Rule Extraction Neural Network Model. *Global Journal of Computer Science and Technology*, v. 11, n. 7, 2011. Citado na página 30.

MALIK, R. Advanced Selfloop Removal in Compositional Nonblocking Verification of Discrete Event Systems. *IEEE International Conference on Automation Science and Engineering*, p. 819–824, 2015. Citado na página 48.

MALIK, R.; LEDUC, R. Hierarchical interface-based supervisory control using the conflict preorder. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, IFAC, p. 163–168, 2012. ISSN 14746670. Citado na página 48.

MARCHAND, H. *et al.* On Optimal Control of a Class of Partially-Observed Discrete Event Systems. *Automatica*, Elsevier, v. 36, n. 2, 2002. Citado na página 34.

MOOR, T. Natural Projections for the Synthesis of Non-Conflicting Supervisory Controllers. *Proceedings of the 12th Workshop on Discrete Event Systems*, IFAC, v. 12, n. 2, p. 300–305, 2014. Citado 2 vezes nas páginas 33 e 48.

MOOR, T.; BAIER, C.; WITTMANN, T. Consistent Abstractions for the Purpose of Supervisory Control. *Proceedings of the IEEE Conference on Decision and Control*, p. 7291–7296, 2013. Citado na página 32.

NING, T. *et al.* A Novel Hybrid Method for Solving Flexible Job-Shop Scheduling Problem. *The Open Cybernetics & Systemics Journal*, v. 10, n. 1, p. 13–19, 2016. Citado na página 30.

OPPENHEIM, A. V.; WILLLSKY, A. S. *Sinais e Sistemas*. 2ª. ed. São Paulo: Pearson Education do Brasil, 2010. 568 p. Citado na página 37.

PANEK, S.; STURSBERG, O.; ENGELL, S. Job-Shop Scheduling by Combining Reachability Analysis with Linear Programming. *7th Int. IFAC Workshop on Discrete Event Systems*, Elsevier, v. 37, n. 18, p. 199–204, 2004. Citado na página 34.

PENA, P. N. *Verificação de Conflito na Supervisão de Sistemas Concorrentes usando Abstrações*. Tese (Doutorado), 2007. Citado 2 vezes nas páginas 53 e 54.

PENA, P. N. *et al.* Verification of the Observer Property in Discrete Event Systems. *IEEE Transactions on Automatic Control*, v. 59, n. 8, p. 2176–2181, 2014. Citado na página 41.

PENA, P. N. *et al.* Control of Flexible Manufacturing Systems under model uncertainty using Supervisory Control Theory and evolutionary computation schedule synthesis. *Information Sciences*, v. 329, p. 491–502, 2016. Citado 3 vezes nas páginas 30, 35 e 84.

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, 12 2009. Citado 4 vezes nas páginas 32, 33, 48 e 56.

PEZZELLA, F.; MERELLI, E. A Tabu Search Method Guided by Shifting Bottleneck for the Job Shop Scheduling Problem. *European Journal of Operational Research*, v. 120, n. 2, p. 297–310, 2000. Citado na página 30.

PINEDO, M. L. *Scheduling - Theory, Algorithms and Systems*. 3ª. ed. [S.l.]: Springer, 2008. Citado na página 28.

QUEIROZ, M. H.; CURY, J. E. R. Modular Supervisory Control of Large Scale Discrete Event Systems. *Discrete Event Systems: Analysis and Control*, p. 103–110, 2000. Citado 3 vezes nas páginas 31, 47 e 48.

RAMADGE, P. J. G.; WONHAM, W. M. The Control of Discrete Event Systems. *Proceedings of the IEEE*, v. 77, n. 1, p. 81–98, 1989. Citado 2 vezes nas páginas 31 e 38.

RAMKUMAR, R.; TAMILARASI, A.; DEVI, T. Multi Criteria Job Shop Schedule Using Fuzzy Logic Control for Multiple Machines Multiple Jobs. *International Journal of Computer Theory and Engineering*, v. 3, n. 2, p. 282–286, 2011. Citado na página 30.

RONCONI, D.; BIRGIN, E. Mixed-Integer Programming Models for Flowshop Scheduling Problems Minimizing the Total Earliness and Tardiness. *Just-in-Time systems*, p. 1–14, 2012. Citado na página 30.

ROSHANAEI, V. *et al.* A Variable Neighborhood Search for Job Shop Scheduling with Set-Up Times to Minimize Makespan. *Future Generation Computer Systems*, Elsevier B.V., v. 25, n. 6, p. 654–661, 2009. Citado na página 30.

SACHE, R. G. Neural Network for Solving Job-Shop Scheduling Problem. *IOSR Journal of Computer Engineering*, v. 16, n. 6, p. 18–25, 2014. Citado na página 30.

SU, R. Abstraction-Based Synthesis of Timed Supervisors for Time-Weighted Systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, IFAC, v. 639798, p. 128–134, 2012. Citado na página 35.

SU, R.; SCHUPPEN, J. H. V.; ROODA, J. E. The Synthesis of Tme Optimal Supervisors by using Heaps-of-Pieces. *IEEE Transactions on Automatic Control*, v. 57, n. 1, p. 105–118, 2012. Citado na página 35.

SU, R.; WONHAM, W. M. Supervisor Reduction for Discrete-Event Systems. *Discrete Event Dynamic Systems*, v. 14, p. 31–53, 2004. Citado na página 56.

TANG, J. *et al.* A Hybrid Algorithm for Flexible Job-Shop Scheduling Problem. *Procedia Engineering*, v. 15, p. 3678–3683, 2011. Citado na página 30.

TAVAKKOLI-MOGHADDAM, R. *et al.* A Hybrid Method for Solving Stochastic Job Shop Scheduling Problems. *Applied Mathematics and Computation*, v. 170, n. 1, p. 185–206, 2005. Citado na página 30.

TEIXEIRA, M. *et al.* Supervisory Control of DES with Extended Finite-State Machines and Variable Abstraction. *IEEE Transactions on Automatic Control*, v. 60, n. 1, p. 118–129, 2015. Citado na página 33.

VILELA, J. N.; PENA, P. N. Supervisor Abstraction to Deal with Planning Problems in Manufacturing Systems. *13th International Workshop on Discrete Event Systems, WODES 2016*, p. 117–122, 2016. Citado 6 vezes nas páginas 36, 51, 52, 53, 56 e 80.

WAKATAKE, M.; NISHI, T.; INUIGUCHI, M. Decomposition of Timed Automata for Solving Jobshop Scheduling Problems. *Iccas-Sice, 2009*, n. C, p. 1–3, 2009. Citado na página 34.

WARE, S.; SU, R. Progressive Time Optimal Control of Reactive Systems. *54th IEEE Annual Conference on Decision and Control*, n. Cdc, p. 3542–3547, 2015. Citado na página 35.

WONG, K. C. On the Complexity of Projections of Discrete-Event Systems. *Journal of Discrete Event Dynamic Systems*, v. 13, 1998. Citado na página 84.

WONG, K. C. *et al.* Supervisory Control of Distributed Systems: Conflict Resolution. *Proceedings of the 37th IEEE Conference on Decision & Control*, p. 3275–3280, 1998. Citado 3 vezes nas páginas 32, 40 e 48.

WONG, K. C.; WONHAM, W. M. Hierarchical Control of Discrete-Event Systems. *Discrete Event Dynamic Systems*, v. 6, n. 3, p. 241–273, 1996. Citado na página 32.

YAMADA, T.; NAKANO, R. Job-Shop Scheduling. In: *Genetic algorithms in engineering systems*. [S.l.]: The Institution of Electrical Engineers, 1997. cap. 7, p. 134–160. Citado 2 vezes nas páginas 27 e 28.

APÊNDICE A – Resultado Adicional

Lema 7. *Se E_j, para todo $j \in J$, é especificação de coordenação conforme a Definição 6 e $\Sigma_{E_j} \cap \Sigma_{E_k} = \emptyset$, $\forall j, k \in J$ e $j \neq k$, então $E = \mathop{\Vert}_{j=1}^{m} E_j$ também será uma especificação de coordenação.* ◇

Demonstração. Para que um autômato seja considerado especificação de coordenação, o mesmo deve possuir as seguintes características:

i) $\mathcal{L}(E)$ é controlável em relação à $\mathcal{L}(G)$

ii) $\forall s \in \Sigma^*, t \in \Sigma_u^*, \sigma \in \Sigma_c, u \in \Pi(t).\ [st\sigma \in \mathcal{L}(E)]$ e $[su \in \mathcal{L}(E)] \implies su\sigma \in \mathcal{L}(E)$

Se E_j é controlável em relação à G_j, então $S_j = E_j || G_j$. Como os alfabetos das especificações são disjuntos, então os supervisores são não conflitantes. Assim, pelo Teorema 1, é possível afirmar que

$$S = \mathop{\Vert\Vert}_{j=1}^{m} S_j = E_1||G_1||E_2||G_2||...||E_m||G_m = E_1||E_2||...||E_m||G_1||G_2||...||G_m$$

Então $E = \mathop{\Vert}_{j=1}^{m} E_j$ será controlável em relação a $G = \mathop{\Vert}_{j=1}^{m} G_j$ e é possível afirmar que E apresenta a característica i).

Sejam as cadeias $s_j \in \mathcal{L}(E_j)$, $t_j \in (\Sigma_u \cap \Sigma_{E_j})^*$, $u_j \in \Pi(t_j)$ e os eventos $\sigma_j \in (\Sigma_c \cap \Sigma_{E_j})$, tais que

$$s_j t_j \sigma_j \in \mathcal{L}(E_j) \text{ e } s_j u_j \in \mathcal{L}(E_j)$$

Como E_j é especificação de coordenação, então

$$s_j u_j \sigma_j \in \mathcal{L}(E_j)$$

Se $\Sigma_{E_j} \cap \Sigma_{E_k} = \emptyset$, $\forall j, k \in J$ e $j \neq k$, então

$$\mathop{\Vert\Vert}_{j=1}^{m} s_j t_j \sigma_j \subseteq \mathcal{L}\left(\mathop{\Vert\Vert}_{j=1}^{m} E_j\right)$$

Para todo $a \in \bigcap_{j=1}^{m} P^{-1}_{\bigcup_{k=1}^{m} \Sigma_{E_k} \to \Sigma_{E_j}}(s_j t_j \sigma_j)$, tem-se que

$$P_{\bigcup_{k=1}^{m} \Sigma_{E_k} \to \Sigma_{E_j}}(a) = s_j t_j \sigma_j$$

Também existem $s \in \Sigma^*$, $t \in \Sigma_u^*$ tal que

$$st \in \overline{a}$$

e

$$st \in \Big\|_{j=1}^{m} s_j t_j$$

Então

$$P_{\bigcup_{k=1}^{m} \Sigma_{E_k} \to \Sigma_{E_j}}(st) = s_j t_j$$

Seja $u \in \Pi(t)$ tal que

$$P_{\bigcup_{k=1}^{m} \Sigma_{E_k} \to \Sigma_{E_j}}(su) = s_j u_j$$

Como os alfabetos são disjuntos, então se $s_j t_j \sigma_j \in \mathcal{L}(E_j)$ tem-se que

$$st\sigma_j \in \mathcal{L}(E)$$

Como E_j é especificação de coordenação, então

$$s_j u_j \sigma_j \in \mathcal{L}(E_j)$$

e tem-se que

$$su\sigma_j \in \mathcal{L}(E), \forall \sigma_j \text{ tal que } st\sigma_j \in \mathcal{L}(E)$$

Isso corresponde à característica ii) e portanto, $E = \Big\|_{j=1}^{m} E_j$ é uma especificação de coordenação.

□

APÊNDICE B – Algoritmo para contagem de sequências

O Algoritmo 1 foi utilizado para a realização da contagem de sequências que levam o estado inicial ao estado marcado de autômatos acíclicos. Além disso, o autômato só deve ter um estado marcado, ou seja, $|Q_m|$[1] $= 1$. Nesse caso, o estado marcado é o elemento $q_m \in Q_m$. Autômatos desse tipo são obtidos ao realizar a operação $Trim$ sobre o resultado da composição paralela de um supervisor ou sua abstração com uma especificação de quantidade, como a da Figura 15, introduzida no Exemplo 1.

Algoritmo 1 Contagem de sequências

Entrada: $G : (\Sigma, Q, \delta, q_0, Q_m)$ - Autômato acíclico
Saída: N - Número de sequências

1: $M \leftarrow$ Número_de_Níveis(G)
2: $\Phi(q_m) \leftarrow 1$
3: **para todo** $m \in \{2, ..., M\}$ **faça**
4: $\quad \Lambda^{-1} \leftarrow$ Estados_do_Nível(m-1)
5: $\quad \Lambda \leftarrow$ Estados_do_Nível(m)
6: $\quad$ **para todo** $s \in \Lambda$ **faça**
7: $\quad\quad \Upsilon \leftarrow \emptyset$
8: $\quad\quad$ **para todo** $\sigma \in \Sigma$ tal que $\delta(s, \sigma) \neq \emptyset$ **faça**
9: $\quad\quad\quad$ **se** $\delta(s, \sigma) \in \Lambda^{-1}$ **então**
10: $\quad\quad\quad\quad \Upsilon \leftarrow \Upsilon \cup \{\delta(s, \sigma)\}$
11: $\quad\quad\quad$ **fim se**
12: $\quad\quad$ **fim para**
13: $\quad\quad$ soma_parcial $\leftarrow 0$
14: $\quad\quad$ **para todo** $u \in \Upsilon$ **faça**
15: $\quad\quad\quad$ soma_parcial $\leftarrow$ soma_parcial + $\Phi(u)$
16: $\quad\quad$ **fim para**
17: $\quad\quad$ $\Phi(s) \leftarrow$ soma_parcial
18: $\quad$ **fim para**
19: **fim para**

[1] A operação $|Q|$, quando Q é um conjunto, retorna a cardinalidade de Q, ou seja, o número de elementos que o compõe.

20: **retorna** $\Phi(q_0)$

Seja o autômato acíclico da Figura 21. Os estados desse autômato podem ser particionados em subconjuntos chamados de níveis, como mostrado na Figura 22. O primeiro nível é aquele que contêm o estado marcado. O segundo nível é composto dos estados que com apenas um evento é possível chegar a um estado do primeiro nível. Um novo nível é formado pelos estados que estão a apenas uma transição de distância dos estados do nível anterior. O último nível é formado pelo estado inicial.

Define-se também a função $\Phi : Q \to \mathbb{N}$, que associa a cada estado $q \in Q$ um número natural que indica o número de sequências que levam o estado q ao estado marcado q_m. Por definição, $\Phi(q_m) = 1$. A ideia básica do algoritmo é a de construir iterativamente a função Φ.

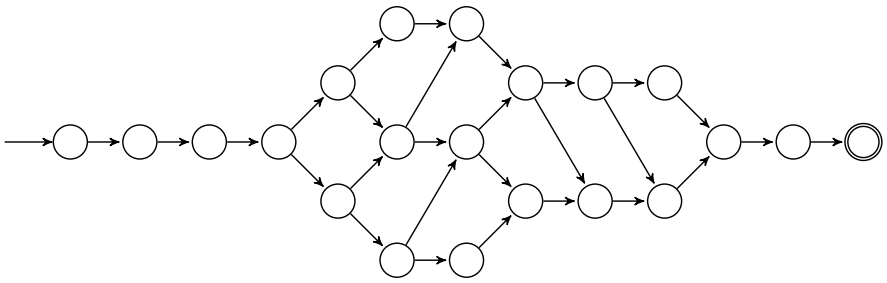


Figura 21 – Autômato acíclico G

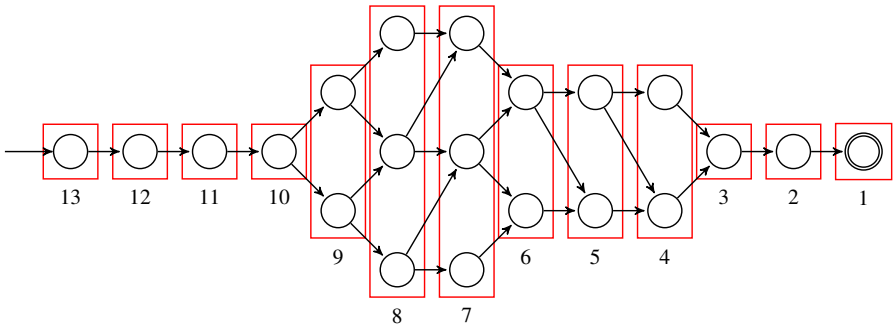


Figura 22 – Partição dos estados de G - O número abaixo do retângulo indica o nível

Ao fim de 3 iterações do laço mais externo do algoritmo (linha 3), o valor de Φ dos dois estados mais à esquerda na Figura 23(a) é igual a 1. Na iteração seguinte, Figura 23(b), o estado superior mais à esquerda tem Φ igual a 2, pois esse estado possui duas transições, cada uma levando a estados cujo valor de Φ é 1. O processo é repetido, como pode ser visto nas demais figuras, até que não haja mais níveis a serem analisados. Nessa situação o número de sequências é $\Phi(q_0) = 22$. O autômato G apresentado como exemplo é obtido

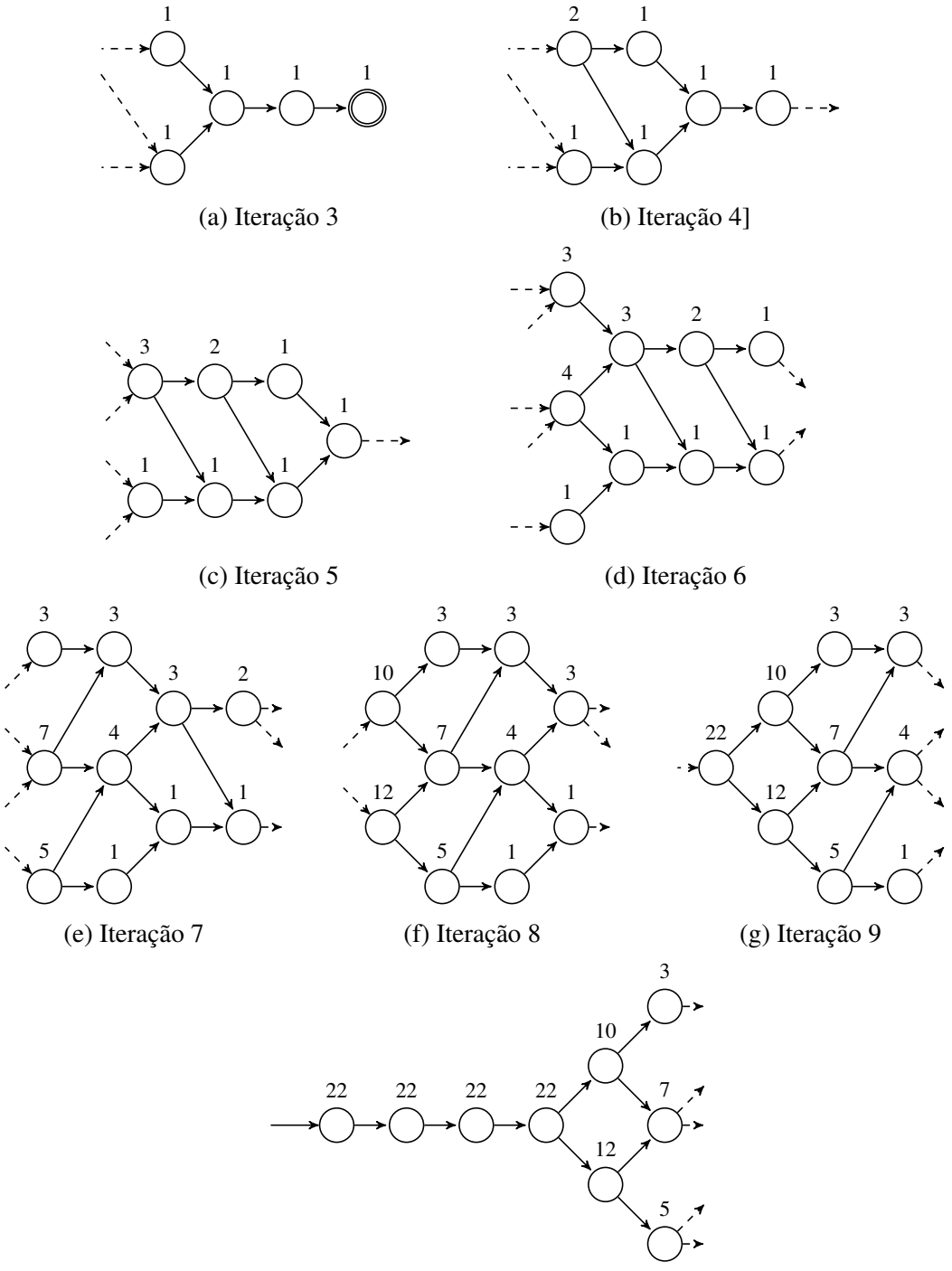


Figura 23 – Iterações do laço externo do Algoritmo 1. O número sobre um estado q é o valor de $\Phi(q)$

pela composição do supervisor monolítico do problema apresentado no Exemplo 1 com a especificação de quantidade para a produção de 2 produtos.