XIII Simpósio Brasileiro de Automação Inteligente
Porto Alegre – RS, 1º – 4 de Outubro de 2017

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

Michel Rodrigo das C. Alves*, Patrícia N. Pena†

*Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brasil

†Departamento de Engenharia Eletrônica
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brasil

Emails: michelrodrigo@ufmg.br, ppena@ufmg.br

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 extension makes feasible the use of abstractions even when it is not possible to deal with the monolithic problem. An application example of the result is presented.

Keywords— Discrete event systems, Planning, Supervisory Control, Observer property.

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. Esta 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. A extensão torna factível o uso das abstrações mesmo quando não é possível tratar o problema monolítico. Um exemplo de aplicação do resultado é apresentado.

Palavras-chave— Sistemas a eventos discretos, planejamento, Controle Supervisório, Propriedade do observador.

1 Introdução

Dentre os diferentes tipos de processos industriais, aqueles de interesse nesse trabalho são os de manufatura, que por sua vez, podem ser analisados sob diferentes aspectos. O escalonamento ou o planejamento de um processo de manufatura trata a alocação de recursos com o objetivo de otimizar um ou mais objetivos (Pinedo, 2008). Para um sistema de manufatura, os recursos são as máquinas e o objetivo pode ser o tempo gasto para a produção de um produto (*makespan*), por exemplo.

Várias técnicas são empregadas para resolver problemas de escalonamento de forma eficiente, aplicando técnicas baseadas em redes neurais (Weckman et al., 2008), lógica fuzzy (Abdullah and Abdolrazzagh-Nezhad, 2014), métodos heurísticos e metaheurísticos (Ziaee, 2014) e autômatos temporizados (Asarin and Maler, 1999). Em muitos casos, no entanto, as abordagens não empregam nenhuma técnica formal que modela o comportamento do sistema de manufatura e que assegura a segurança da operação, evitando que o mesmo entre em estados não seguros ou bloqueantes.

Sistemas de manufatura podem ser classificados como Sistemas a Eventos Discretos (SED) e podem ser descritos por autômatos de estados finitos determinísticos (AFD), o que permite a utilização da Teoria de Controle Supervisório (TCS) (Ramadge and Wonham, 1989). A TCS gera AFDs controladores para os SEDs, os supervisores, que são minimamente restritivos, ou seja, permitem a execução de todas as sequências que são legais. No entanto, a TCS possui uma característica que pode restringir seu uso em sistemas muito complexos, que é a explosão do número de estados durante o procedimento de cálculo do supervisor.

A TCS, usada em conjunto com métodos de otimização para encontrar uma sequência ótima de tarefas, restringe o espaço de busca somente às sequências factíveis. Por outro lado, o tamanho do espaço de busca ainda pode ser um fator que torna um método de otimização exato inviável. Como forma de contornar esse problema, abstrações podem ser aplicadas sobre o supervisor, com o objetivo de reduzir seu tamanho. No entanto, é essencial que qualquer sequência escolhida a partir da abstração deve corresponder à uma sublinguagem do comportamento em malha fechada que seja controlável em relação à planta.

Em (Vilela and Pena, 2016) é apresentada uma abstração do supervisor que mantém apenas

os eventos controláveis e que garante a controlabilidade em relação à planta das sequências obtidas a partir da mesma. A condição para que a abstração seja aplicável em um supervisor é que sua projeção natural para o subconjunto de eventos controláveis possua a propriedade do observador.

O uso da referida abstração em conjunto com métodos de otimização, pode tornar a busca pela sequência ótima num problema de planejamento mais eficiente, já que há uma redução do espaço de busca. No entanto, a complexidade da síntese do supervisor da TCS pode ser um obstáculo em aplicações muito grandes, já que seu número de estados aumenta exponencialmente com o número de componentes do sistema.

Esse artigo apresenta uma extensão do resultado principal de (Vilela and Pena, 2016), para a aplicação sobre os supervisores obtidos pela Síntese Modular Local (Queiroz and Cury, 2000), uma extensão da TCS que busca contornar o problema de complexidade computacional. A condição para que seja possível a realização da abstração é que a projeção natural no subconjunto de eventos controláveis de cada um dos supervisores possua a propriedade do observador.

O artigo é organizado da seguinte maneira. A Seção 2 introduz os conhecimentos necessários para a compreensão do presente trabalho. A Seção 3 apresenta a abstração para o supervisor monolítico e o resultado principal desse trabalho. A Seção 4 apresenta um exemplo de aplicação em que a abstração do supervisor monolítico e dos supervisores modulares locais é aplicada. Finalmente, a Seção 5 apresenta os comentários finais, conclusões e propostas de continuidade.

2 Preliminares

O comportamento de um SED é modelado usando cadeias de eventos pertencentes a um alfabeto finito Σ. O Fechamento Kleene Σ^* é o conjunto de todas as cadeias finitas formadas por elementos de Σ, incluindo a cadeia vazia ε. Um subconjunto $L \subseteq \Sigma^*$ é chamado linguagem. A operação de concatenação de duas cadeias $s, u \in \Sigma^*$ é escrita como su. A cadeia s é prefixo de t, $s \leq t$, se existe $u \in \Sigma^*$ tal que $su = t$. O prefixo fechamento $\overline{L}$ de uma linguagem $L \subseteq \Sigma^*$ é definido como $\overline{L} = \{s \in \Sigma^* | s \leq t \text{ para algum } t \in L\}$.

Uma linguagem é regular se pode ser representada por um autômato de estados-finitos, representado por uma quíntupla $G = (\Sigma, Q, \hat{\delta}, q_0, Q_m)$, onde Σ é um conjunto finito de eventos, Q é um conjunto finito de estados, $\hat{\delta} : Q \times \Sigma \to Q$ é a função de transição, $q_0 \in Q$ é o estado inicial e $Q_m \subseteq Q$ é o conjunto finito de estados marcados. G é determinístico se $\hat{\delta}(x, \sigma) = y_1$ e $\hat{\delta}(x, \sigma) = y_2$ sempre implica em $y_1 = y_2$. A função de transição pode ser estendida para lidar com cadeias, $\delta : Q \times \Sigma^* \to Q$. Neste caso, $\delta(q, \epsilon) = \epsilon$ e $\delta(q, s\sigma) = \hat{\delta}(\delta(q, s), \sigma)$. A linguagem gerada de um autômato G é definida como $\mathcal{L}(G) = \{s \in \Sigma^* | \delta(q_0, s) \text{ está definida}\}$ e a linguagem marcada é $\mathcal{L}_m(G) = \{s \in \Sigma^* | \delta(q_0, s) \in Q_m\}$.

A projeção natural é uma operação realizada sobre linguagens, definida a partir de dois conjuntos de eventos Σ_i e Σ, em que $\Sigma_i \subseteq \Sigma$. A projeção natural $P_i : \Sigma^* \to \Sigma_i^*$ mapeia as cadeias de Σ^* para cadeias em Σ_i^*, apagando todos os eventos que não estão em Σ_i. A projeção inversa $P_i^{-1} : \Sigma_i^* \to \Sigma^*$ é definida como $P_i^{-1}(L) = \{s \in \Sigma^* | P_i(s) \in L\}$. A noção de projeção inversa pode ser utilizada para prover a definição formal da operação de composição paralela de linguagens $L_i \subseteq \Sigma_i^*, i \in I$, com $\Sigma = \bigcup_{i \in I} \Sigma_i$:

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

A projeção natural pode ter uma propriedade conhecida como propriedade do observador, apresentada na Definição 2.1.

Definição 2.1 *(Wong et al., 1998) Seja uma linguagem $L \subseteq \Sigma^*$, um alfabeto $\Sigma_i \subseteq \Sigma$ e $P : \Sigma^* \to \Sigma_i^*$ a projeção natural de cadeias em Σ^* para cadeias em Σ_i^*. Se $(\forall a \in \overline{L})(\forall b \in \Sigma_i^*), P(a)b \in P(L) \implies (\exists c \in \Sigma^*)P(ac) = P(a)b$ e $ac \in L$, então a projeção natural $P(L)$ tem a propriedade do observador.*

Uma abstração obtida pela operação de projeção natural e que contenha 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).

2.1 Teoria de Controle Supervisório

A Teoria de Controle Supervisório é um método formal baseado na teoria de linguagens e autômatos, para o cálculo sistemático de supervisores.

A planta é modelada por um autômato $G = (Q, \Sigma, \hat{\delta}, q_0, Q_m)$ em que $\Sigma = \Sigma_c \cup \Sigma_{uc}$. Σ_c é o conjunto de eventos controláveis e Σ_{uc} é o conjunto de eventos não controláveis. O comportamento em malha aberta da planta é representado por duas linguagens, o comportamento gerado $\mathcal{L}(G)$ e o comportamento marcado $\mathcal{L}_m(G)$. O comportamento desejado K é obtido pela composição paralela entre as especificações E e $\mathcal{L}_m(G)$. Assim, o supervisor S irá atuar sobre os eventos controláveis de modo a restringir o comportamento da planta à linguagem desejada. Uma linguagem K é controlável em relação à G se $\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$. Se essa relação não for verificada, obtém-se o supervisor S que implementa a máxima sub-linguagem controlável $Sup\mathcal{C}(K, G)$, comportamento mais permissivo e não bloqueante

do sistema que respeita as especificações. O supervisor S implementa $S = Sup\mathcal{C}(K, G)$ (Ramadge and Wonham, 1989).

Seja G um sistema composto de subsistemas assíncronos G_i, com $i \in I = \{1, ..., n\}$. Os subsistemas G_i devem ser restringidos pelas especificações genéricas locais $E_{gen,j}$, com $j = \{1, ..., m\}$. Para cada $E_{gen,j}$, a planta local associada $G_{loc,j}$ é obtida pela composição dos subsistemas da modelagem original que estão diretamente restringidos por $E_{gen,j}$. Obtém-se então o conjunto de linguagens $K_j = E_{gen,j}||\mathcal{L}_m(G_{loc,j})$ e um supervisor $S_j = Sup\mathcal{C}(E_{loc,j}, G_{loc,j})$ para cada $j \in J$.

A ação dos supervisores obtidos só é equivalente ao supervisor monolítico se o conjunto de supervisores S_j, $j = \{1, ..., m\}$, é não conflitante. Para verificar se são não conflitantes, deve-se verificar a igualdade $||_{i=1}^{n}\overline{S_i} = \overline{||_{i=1}^{n}S_i}$ (de Queiroz and Cury, 2000).

2.2 CSO - Controle Supervisório e Otimização

Em (Pena et al., 2016) é proposta uma abordagem para o problema de escalonamento ótimo de tarefas em um sistema de manufatura. O comportamento controlável e minimamente restritivo é obtido utilizando a TCS. Com objetivo de se encontrar a sequência de produção de um lote de produtos que minimize o *makespan*, agrega-se ao comportamento controlado do sistema a informação dos tempos das operações dos subsistemas.

A metodologia CSO é composta de duas partes principais: (i) otimizador; (ii) avaliador de sequências. Em linhas gerais, o otimizador gera indivíduos que são avaliados pelo avaliador de sequências. O avaliador de sequências recebe uma sequência do otimizador, avalia seu *makespan* e devolve para o otimizador. Este procedimento é repetido até que uma condição de parada seja alcançada.

Todo o processo inicia-se com sequências factíveis e, a partir delas, outras sequências (vizinhos) são geradas pela movimentação de eventos ou de blocos de eventos na própria sequência, de forma aleatória. Estes movimentos geram, muitas vezes, indivíduos infactíveis. Neste caso, cabe ao avaliador informar em qual posição a sequência tornou-se infactível e auxiliar o otimizador na correção da sequência.

Utilizando o resultado apresentado neste trabalho, pretende-se eliminar a geração de indivíduos infactíveis pela utilização de uma abstração do supervisor para guiar a geração dos indivíduos que populam as gerações no procedimento de otimização.

É possível ainda aplicar o resultado obtido com o intuito de se reduzir o espaço de busca em (Marchand et al., 2002), onde pesos são atribuídos aos eventos e busca-se encontrar a melhor sequência em relação a uma função custo.

3 Resultado Principal

O supervisor obtido na abordagem monolítica contém a informação de todas as cadeias possíveis de serem executadas sem violar nenhuma das restrições de segurança. Determinar qual dessas cadeias é ótima em relação a algum critério não é trivial, pois o espaço de busca é geralmente muito grande.

Considera-se válido o argumento usado em (Pena et al., 2016) de que o escalonamento de tarefas em uma planta deve ser realizada apenas pela ordenação de comandos, que consistem nas ações que podem ser controladas pelo operador (eventos controláveis). No entanto, ao determinar a sequência de comandos a ser executada na planta, sabendo-se da existência dos eventos não controláveis, é importante garantir que uma sequência modelada apenas pelos comandos representará uma sublinguagem controlável do comportamento em malha fechada.

No intuito de reduzir a complexidade do problema, é possível reduzir o espaço de busca da solução ótima, sob certas condições, com o uso de OP-abstrações. Este resultado está expresso no Teorema 1.

Teorema 1 *(Vilela and Pena, 2016) Seja G um sistema, S a máxima sublinguagem controlável contida na linguagem desejada K e $P : \Sigma^* \to \Sigma_c^*$ a projeção natural. Para toda sequência $s_{ot} \in P(S)$ e $A = P^{-1}(s_{ot}) \cap S$, se $P(S)$ possui a propriedade do observador, então A sempre será controlável em relação a G.*

Neste trabalho, pretende-se estender este teorema para lidar, não com um supervisor monolítico, mas com supervisores obtidos pela síntese modular local. Esta extensão se justifica pela dificuldade, muitas vezes, de se verificar a propriedade do observador na projeção de um supervisor monolítico. Ao lidar com vários supervisores menores, esta dificuldade é reduzida.

Se uma projeção possui a propriedade do observador, então o resultado da projeção será sempre menor do que o modelo original. Contudo, a complexidade computacional para o cálculo da projeção natural do supervisor monolítico ainda pode ser um fator limitante para sua utilização, já que no pior caso o cálculo da projeção tem uma complexidade exponencial em termos do número de estados do autômato (Wong, 1998). Sendo a projeção aplicada sobre os supervisores modulares, aumentam as chances desta projeção ser computável.

Assim sendo, a principal contribuição desse trabalho é apresentada no Teorema 2 e estende o Teorema 1 para lidar com supervisores obtidos pela síntese modular local.

Teorema 2 *Seja G um sistema, S_1 e S_2 supervisores não conflitantes obtidos pela síntese modular local. Sejam as projeções naturais $P_1 : \Sigma_1^* \to \Sigma_{1c}^*$ e $P_2 : \Sigma_2^* \to \Sigma_{2c}^*$ com $\Sigma_{ic} = \Sigma_i \cap \Sigma_c$. Para toda sequência $s \in P_1(S_1)||P_2(S_2)$ e $A = P^{-1}_{\Sigma \to \Sigma_c}(s) \cap (S_1||S_2)$, se $P_1(S_1)$ e $P_2(S_2)$ possuem a propriedade do observador, então A será controlável em relação à G.*

Prova: A prova é omitida por questão de espaço. □

Em palavras, o Teorema 2 estabelece que, dados dois supervisores S_1 e S_2, se as projeções $P_1(S_1)$ e $P_2(S_2)$ possuírem a propriedade do observador, então qualquer cadeia de eventos presente em $P_1(S_1)||P_2(S_2)$ será "executável" na planta. Por "executável" na planta, entende-se que a sublinguagem do comportamento em malha fechada que projeta para esta cadeia é controlável. Como os supervisores envolvidos no controle modular local são, tipicamente, menores do que o obtido pela síntese monolítica, então a complexidade envolvida no cálculo de $P_1(S_1)||P_2(S_2)$ será menor do que para a projeção do supervisor monolítico.

4 Exemplo de aplicação

Seja um sistema composto por 3 máquinas $M_i, i = \{1, 2, 3\}$, intercaladas por dois *buffers* $B_j, j = \{1, 2\}$ de capacidade unitária, mostrado na Figura 1. Um insumo bruto deve ser processado pelas 3 máquinas, em sequência, para que se torne um produto final. O comportamento da máquina M_i é modelado pelo autômato mostrado na Figura 2(a). No estado inicial, a máquina está desligada e aguardando um produto para processá-lo. O evento que representa a máquina sendo ligada é o α_i, levando-a para o estado 1, que simboliza seu funcionamento. O evento β_i, por sua vez, representa o fim do processamento do produto pela máquina M_i, levando-a a seu estado inicial. Os eventos α_i são controláveis, enquanto os eventos β_i são não controláveis.

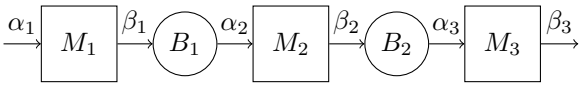
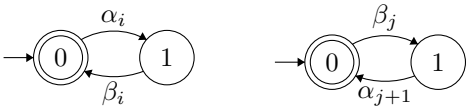


Figura 1: Sistema de manufatura



(a) Modelo das máquinas, $i \in \{1, 2, 3\}$

(b) Especificações $j \in \{1, 2\}$

Figura 2: Modelos das máquinas e especificações

A restrição do problema é a de que não pode haver *underflow* ou *overflow* nos *buffers* B_j. Essa restrição é modelada pelo autômato mostrado na Figura 2(b).

Para a produção de n produtos finais, é necessária uma sequência de eventos em que cada evento α_i e β_i apareça exatamente n vezes, desde que seja respeitada a ordem de precedência das máquinas. Considerando apenas os eventos controláveis, então é necessário uma cadeia em que os eventos α_i apareçam exatamente n vezes.

Usando a abordagem CSO, um indivíduo seria gerado e vizinhos seriam construídos e avaliados. Por exemplo, $t_1 = \alpha_1\alpha_2\alpha_3\alpha_1\alpha_2\alpha_3$. O procedimento para formação de outros indivíduos (vizinhos (Pena et al., 2016) ou clones (Costa et al., 2013), dependendo da técnica utilizada) é o de escolher aleatoriamente posições e trocar os eventos de lugar. Possíveis vizinhos seriam $t'_1 = \alpha_3\alpha_2\alpha_3\alpha_1\alpha_2\alpha_1$ e $t''_1 = \alpha_3\alpha_1\alpha_1\alpha_2\alpha_2\alpha_3$. Ambos são infactíveis e teriam de ser corrigidos antes de serem avaliados para o *makespan*. Em (Pena et al., 2016), as sequências são modeladas com invariantes que reduzem de forma importante a ocorrência de infactibilidades, sem nunca chegar perto de zerar suas ocorrências.

O uso do resultado apresentado no Teorema 1 permite que a geração de novos indivíduos seja realizada na abstração do supervisor, caso esta seja uma OP-abstração. Por questão de clareza, omite-se o autômato do supervisor, que tem 18 estados e 32 transições. A abstração do supervisor é apresentada na Figura 3 e possui apenas 4 estados.

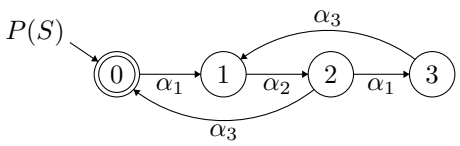


Figura 3: Projeção do Supervisor Monolítico, $P(S)$

Este autômato de quatro estados modela todas as sequências de comandos possíveis (legais) para este sistema. Para explicitar o conjunto de todas as cadeias que geram dois produtos, bastaria compor $P(S)$ com uma especificação que implemente a cadeia $\alpha_3\alpha_3$ (Figura 4). Esta composição gera o autômato apresentado na Figura 5.

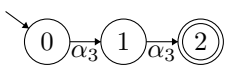


Figura 4: Especificação de quantidade para a produção de 2 produtos

Pode-se perceber que o conjunto de sequências de comandos para produção de 2 produtos se resume a duas sequências de eventos controláveis, $s_1 = \alpha_1\alpha_2\alpha_3\alpha_1\alpha_2\alpha_3$ e $s_2 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\alpha_3$.

Considerando que todo este procedimento busca escolher uma cadeia de comandos que mi-

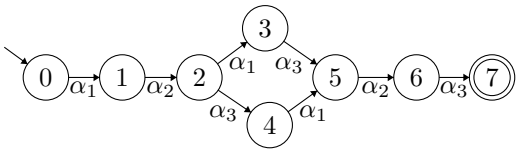


Figura 5: Composição paralela entre $P(S)$ e a especificação de quantidade para 2 produtos

nimize alguma função objetivo (*makespan*, por exemplo) é importante que se garanta que esta cadeia, quando executada na planta consiga executar até o final (ou seja, que a ocorrência espontânea de eventos não controláveis não cause a infactibilidade da sequência). Esta verificação é traduzida, matematicamente, na verificação de que a sublinguagem do comportamento em malha fechada do sistema que projeta para a cadeia escolhida seja controlável.

Para ilustrar a ideia, em etapas, escolhe-se uma das cadeias de comandos que produzem 2 produtos,

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

Constrói-se a linguagem $A = P^{-1}_{\Sigma\rightarrow\Sigma_c}(s_{ot}) \cap (S_1||S_2)$. A linguagem A, se comparada ao comportamento em malha fechada do sistema, terá as desabilitações indicadas por setas tracejadas, na Figura 8. Pode-se perceber que A é controlável, tendo em vista que não foi necessário desabilitar nenhum evento não controlável.

4.1 Uso de Supervisores obtidos pela Síntese Modular Local

Em diversas ocasiões, há a possibilidade de utilizar um conjunto de supervisores modulares no lugar do supervisor monolítico, implementando o mesmo comportamento em malha fechada. Nestes casos, o planejamento pode ser feito sobre este conjunto de supervisores, usando abstrações dos mesmos. A vantagem de se usar abstrações dos supervisores modulares está na maior facilidade em obter as abstrações pela projeção natural e também de verificar a propriedade do observador. Considerando o mesmo exemplo, apresentam-se os supervisores $S_j = Sup\mathcal{C}(E_j, G_j)$ obtidos pela Síntese Modular Local (Queiroz and Cury, 2000) (Figura 6).

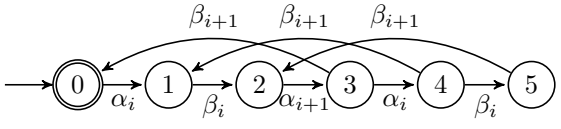
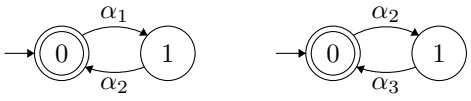


Figura 6: Supervisores modulares locais $S_i = Sup\mathcal{C}(E_i, G_i)$, com $G_i = M_i||M_{i+1}$, com $i \in \{1, 2\}$.

Inicialmente é feita a projeção natural dos supervisores no alfabeto composto apenas pelos eventos controláveis. Os autômatos resultantes são mostrados na Figura 7, com $P_1 : \Sigma^* \rightarrow \Sigma^*_{1c}$ e $P_2 : \Sigma^* \rightarrow \Sigma^*_{2c}$ e são OP-abstrações, o que pode ser constatado pela aplicação do algoritmo OP-verificador (Pena et al., 2014).



(a) $P_1(S_1)$ (b) $P_2(S_2)$

Figura 7: Projeções dos Supervisores Modulares Locais

A composição paralela das abstrações resulta em um autômato idêntico ao apresentado na Figura 3.

A partir do autômato composto, dado um tamanho de lote, pode-se obter o conjunto de sequências de comando que são legais para este sistema. Para o caso de um lote de tamanho 2, o autômato resultante coincide com a Figura 5.

Para problemas de pequeno porte, é possível listar a avaliar todas as sequências de comando legais, e decidir entre elas. No caso em que isto não é possível, pode-se aplicar algum método de otimização sobre este autômato, de forma a encontrar uma solução que seja boa.

5 Conclusão

Apresenta-se neste trabalho uma extensão do Teorema 1 que usa os supervisores obtidos pela síntese modular local no lugar do supervisor monolítico. Esta extensão torna viável a aplicação do resultado anterior para os casos em que a verificação da propriedade do observador sobre a projeção do supervisor monolítico ou a própria projeção não possam ser computadas.

Uma observação adicional é que nos exemplos testados até agora, a igualdade $P(S) = P_1(S_1)||P_2(S_2)$ se verifica. No entanto, sabe-se que esta condição não se verifica, necessariamente. A condição adicional à propriedade do observador é que os eventos compartilhados por S_1 e S_2 sejam mantidos pela projeção, o que não é o caso. Acredita-se que, sob certas condições relacionadas às características estruturais dos supervisores, modelos de plantas e especificações, é possível demonstrar a igualdade para o caso em que a projeção apaga eventos não controláveis. Esta é uma proposta de continuidade deste trabalho.

Outra proposta de continuidade consiste na modificação do mecanismo de geração de novos indivíduos da CSO, para eliminar o custo computacional de lidar com infactibilidades nas sequências que são avaliadas.

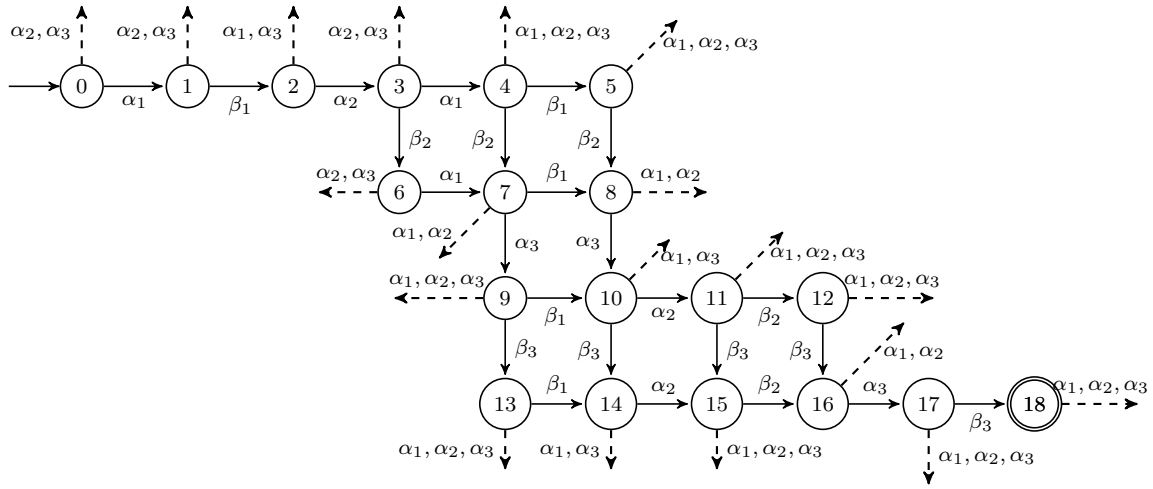


Figura 8: Autômato que implementa A.

Agradecimentos

O presente trabalho foi realizado com o apoio financeiro da Fapemig, CAPES - Brasil e CNPq.

Referências

Abdullah, S. and Abdolrazzagh-Nezhad, M. (2014). Fuzzy job-shop scheduling problems: A review, *Information Sciences* **278**: 380–407.

Asarin, E. and Maler, O. (1999). As Soon as Possible : Time Optimal Control for Timed Automata, *Proceedings of 2nd International Workshop Of Hybrid Systems: Computer Control* **1596**: 19–30.

Costa, T. A., Oliveira, A. C., Pena, P. N. and Takahashi, R. H. C. (2013). Clonal Selection Algorithms For Task Scheduling In A Flexible Manufacturing Cell With Supervisory Control, *IEEE Congress on Evolutionary Computation* pp. 2515–2522.

de Queiroz, M. H. and Cury, J. E. R. (2000). Modular control of composed systems, *American Control Conference, 2000. Proceedings of the 2000* **6**(June): 4051–4055 vol.6.

Marchand, H., Boivineau, O., Lafortune, S. and Id, H. (2002). On Optimal Control of a Class of Partially-Observed Discrete Event Systems, *Automatica* **36**(2).

Pena, P. N., Bravo, H. J., Da Cunha, A. E. C., Malik, R. and Lafortune, S. (2014). Verification of the Observer Property in Discrete Event Systems, *IEEE Transactions on Automatic Control* **59**(8): 2176–2181.

Pena, P. N., Costa, T. A., Silva, R. S. and Takahashi, R. H. C. (2016). Control of Flexible Manufacturing Systems under model uncertainty using Supervisory Control Theory and evolutionary computation schedule synthesis, *Information Sciences* **329**: 491–502.

Pinedo, M. L. (2008). *Scheduling - Theory, Algorithms and Systems*, 3ª edn, Springer.

Queiroz, M. H. and Cury, J. E. R. (2000). Modular Supervisory Control of Large Scale Discrete Event Systems, *Discrete Event Systems: Analysis and Control* pp. 103–110.

Ramadge, P. J. G. and Wonham, W. M. (1989). The Control of Discrete Event Systems, *Proceedings of the IEEE* **77**(1): 81–98.

Vilela, J. N. and Pena, P. N. (2016). Supervisor abstraction to deal with planning problems in manufacturing systems, *13th International Workshop on Discrete Event Systems, WODES 2016* pp. 117–122.

Weckman, G. R., Ganduri, C. V. and Koonce, D. A. (2008). A neural network job-shop scheduler, *Journal of Intelligent Manufacturing* **19**(2): 191–201.

Wong, K. C. (1998). On the Complexity of Projections of Discrete-Event Systems, *Journal of Discrete Event Dynamic Systems* **13**.

Wong, K. C., Thistle, J. G., Malhamk, R. P. and Hoang, H.-H. (1998). Supervisory Control of Distributed Systems: Conflict Resolution.

Ziaee, M. (2014). A heuristic algorithm for solving flexible job shop scheduling problem, *The International Journal of Advanced Manufacturing Technology* **71**(1-4): 519–528.