Abstração do Supervisor para Sistemas com Retrabalho visando a Solução de um Problema de Planejamento

Franklin M. Garcia-Acevedo * Patrícia N. Pena **

* Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, (e-mail: franklinmga@ufmg.br)

** Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, (e-mail: ppena@ufmg.br)

Abstract: Finite State Automata and Supervisory Control Theory have been used to model and solve scheduling problems in manufacturing systems. By working with abstractions of the supervisor that are OP-Abstractions allowed to reduce the search universe of the planning problem. In the present work, an extension of the previous result of Vilela and Pena (2016) to the case when the abstraction of the closed-loop behavior is not an OP-Abstraction is proposed. We present a procedure to obtain a language, from a sequence chosen by its performance, that is allowed to be executed until the end. The procedure makes it possible to use abstractions of the closed-loop behavior in systems that are not OP-Abstractions by nature. Finally, an application example is presented.

Resumo: Autômatos de Estados Finitos e a Teoria de Controle Supervisório têm sido usados para modelar e solucionar problemas de escalonamento de tarefas em sistemas de manufatura. Trabalhar com PO-Abstrações do comportamento em malha fechada permite reduzir o universo de busca do problema de planejamento. No presente trabalho uma extensão do resultado de Vilela and Pena (2016) para o caso em que a abstração do comportamento em malha fechada não é PO-Abstração é proposta. Desenvolve-se um procedimento para construir uma linguagem, a partir de uma sequência selecionada por seu desempenho, que pode ser executada até o final. Esta extensão torna possível o uso de abstrações do comportamento em malha fechada de sistemas que por natureza não são PO-Abstrações. Por último, um exemplo de aplicação é apresentado.

Keywords: Supervisory Control Theory; abstraction; observer property; planning.

Palavras-chaves: Teoria de Controle Supervisório; abstração; propriedade do observador, planejamento.

1. INTRODUÇÃO

O problema de planejamento de tarefas em sistemas de manufatura consiste em encontrar uma sequência de operação ótima que minimize uma função objetivo (*makespan* ou *energia consumida*, por exemplo) satisfazendo os recursos e as restrições de precedência (Nishi and Wakatake, 2014). Existem trabalhos que lidam com o problema do planejamento no contexto da Teoria de Controle Supervisório (TCS) tais como (Pinha et al., 2011), (Huang and Kumar, 2007), (Su, 2012), (Alves et al., 2016) entre outros.

A Teoria de Controle Supervisório (TCS) (Ramadge and Wonham, 1989) propõe a síntese de controladores para *Sistemas a Eventos Discretos* (SED), matematicamente fundamentada na teoria de linguagens e autômatos (Cassandras and Lafortune, 2009). Um dos principais problemas da TCS é o custo computacional para sintetizar o supervisor minimamente restritivo, controlável e não blo-

queante. O problema surge como resultado do crescimento exponencial no número de estados do modelo, que está em função do número de componentes do sistema (Teixeira et al., 2013).

A dificuldade principal na solução de problemas de planejamento com SED é a explosão de estados. Geralmente, muitos dos problemas de planejamento são conhecidos por serem problemas de otimização combinatória NP-Difícil (Nishi and Wakatake, 2014). Usualmente, neste tipo de problema, o conjunto de todas as soluções é dada por $(N!)^M$, N representa o número de tarefas e M o número de máquinas (Arisha et al., 2001). Uma das abordagens que têm sido propostas para lidar com a explosão de estados é trabalhar com abstrações (projeção natural) do comportamento em malha fechada sobre o conjunto de eventos controláveis (Vilela and Pena, 2016). A principal vantagem de trabalhar com abstrações é a redução do espaço de busca de um problema de planejamento. Com um espaço de busca menor a busca das sequências ótimas torna-se, em termos computacionais, mais eficiente.

* O presente trabalho foi realizado com o apoio financeiro das agências Brasileiras CNPq, CAPES e FAPEMIG.

Em (Vilela and Pena, 2016) uma abstração do comportamento em malha fechada que garante a controlabilidade em relação à planta para todas as sequências obtidas na abstração é apresentada. O requisito para que a abstração seja aplicável é que a projeção natural feita sobre o conjunto dos eventos controláveis deve possuir a Propriedade do Observador (PO) que foi definida por (Wong, 1998).

O uso de abstrações com a propriedade do observador, em conjunto com métodos de otimização, pode tornar o problema de planejamento mais eficiente (Rafael and Pena, 2018). No entanto, existem sistemas de manufatura reais que, ao realizar a abstração do comportamento em malha fechada apenas sobre o conjunto de eventos controláveis não possuem a propriedade do observador.

Este artigo apresenta uma extensão do resultado principal de (Vilela and Pena, 2016) para o caso em que a projeção natural do supervisor não é PO-abstração, usando o método de renomeação de eventos proposto em (Pena et al., 2010).

Na Seção 2 são introduzidos os conceitos básicos para a compreensão do presente trabalho, bem como o resultado anterior de Vilela and Pena (2016). Na Seção 3 um exemplo em que as condições para a aplicação do resultado de (Vilela and Pena, 2016) não estão presentes é apresentado. Na Seção 4 introduz-se o resultado principal do trabalho e na Seção 5 aplica-se o resultado principal ao exemplo da Seção 3. Finalmente, na Seção 6, apresentam-se os comentários finais, conclusões e propostas de continuidade.

2. PRELIMINARES

Este artigo é definido no âmbito da Teoria de Controle Supervisório, apresentada de forma detalhada por Cas-sandras and Lafortune (2009). Seja Σ um conjunto finito de símbolos (eventos). Uma sequência de eventos tirada do conjunto Σ forma uma cadeia. Σ^* denota o conjunto de todas as cadeias finitas formadas com elementos de Σ , incluindo a cadeia vazia ϵ . Dadas duas cadeias $s, t \in \Sigma^*$, s é prefixo de t se houver $u \in \Sigma^*$ tal que $su = t$, em que su denota a concatenação de s e u .

Qualquer subconjunto $L \subseteq \Sigma^*$ é chamado de linguagem. A linguagem L é dita ser prefixo fechada se $\bar{L} = L$, em que $\bar{L} = \{u \in \Sigma^* : u \text{ é um prefixo de } s \text{ para } s \in L\}$ é o prefixo fechamento de L . Duas linguagens L e B são não-conflitantes quando $\bar{L} \cap \bar{B} = \bar{L} \cap \bar{B}$.

Na Teoria de Controle Supervisório de Ramadge and Wonham (1989) a planta é modelada por um autômato de estados finitos determinístico, representada por uma quintupla $G = (Q, \Sigma, \delta, q_0, Q_m)$, sendo Q um conjunto finito de estados, Σ um conjunto finito de eventos $\delta : Q \times \Sigma \rightarrow Q$ a função de transição de estados, $q_0 \in Q$ o estado inicial e $Q_m \subseteq Q$ o subconjunto de estados marcados. A notação Γ representa $\Gamma : Q \rightarrow 2^\Sigma$ a função do conjunto de eventos factíveis. O conjunto de eventos Σ é particionado em dois subconjuntos, Σ_c (eventos controláveis, que podem ser desabilitados pelo controle) e Σ_u (eventos não-controláveis, que representam a resposta espontânea da planta, e não podem ser desabilitados).

A função de transição pode ser estendida para lidar com cadeias, $\delta : Q \times \Sigma^* \rightarrow Q$ em que, $\delta(q, \epsilon) = q$ e $\delta(q, s\sigma) =$

$\delta(\delta(q, s), \sigma)$. O comportamento da planta G é representado por uma linguagem gerada definida como $\mathcal{L}(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \text{ está definida}\}$ e $\mathcal{L}(G) \subseteq \Sigma^*$. $\mathcal{L}(G)$ é uma linguagem que contém todas as sequências de eventos que podem ser geradas. A linguagem marcada de G está definida como $\mathcal{L}_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$ e $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$. A linguagem marcada contém o conjunto de cadeias que define que determinados objetivos foram alcançados. Um autômato é *trim* se todos os estados são alcançados desde o estado inicial e têm uma continuação para um estado marcado.

A projeção natural $P : \Sigma^* \rightarrow \Sigma_i^*$ é uma operação sobre linguagens definida sobre o conjunto de eventos Σ e Σ_i , com $\Sigma_i \subseteq \Sigma$. A projeção natural mapeia cadeias de Σ^* para cadeias em Σ_i^* apagando todos os eventos que não estão em Σ_i . Esta operação está definida para cadeias como:

$$P_{\Sigma \rightarrow \Sigma_i}(\epsilon) := \epsilon$$

$$P_{\Sigma \rightarrow \Sigma_i}(s\sigma) := \begin{cases} P_{\Sigma \rightarrow \Sigma_i}(s) & \text{se } s \in \Sigma^*, \sigma \notin \Sigma_i \\ P_{\Sigma \rightarrow \Sigma_i}(s)\sigma & \text{se } s \in \Sigma^*, \sigma \in \Sigma_i. \end{cases}$$

O conceito é estendido para linguagens como $P_{\Sigma \rightarrow \Sigma_i}(L) = \{t \in \Sigma_i^* \mid t = P(s) \text{ para algum } s \in L\}$. A projeção inversa é definida como $P_{\Sigma \rightarrow \Sigma_i}^{-1}(L) = \{s \in \Sigma^* \mid P_{\Sigma \rightarrow \Sigma_i}(s) \in L\}$. A projeção inversa pode ser usada para definir formalmente a operação de composição paralela de linguagens $L_i \subseteq \Sigma_i^*, i \in I$, com $\Sigma = \bigcup_{i \in I} \Sigma_i$ sendo dada por $\parallel_{i \in I} L_i = \bigcap_{i \in I} P_{\Sigma \rightarrow \Sigma_i}^{-1}(L_i)$

A projeção natural pode ter a propriedade do observador, conforme a Definição 1.

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

A propriedade do observador pode ser verificada utilizando o algoritmo apresentado em Pena et al. (2014). O algoritmo tem complexidade quadrática e não é preciso calcular a projeção natural propriamente dita. Se a propriedade do observador é satisfeita, então a projeção é chamada de PO-Abstração. Caso a propriedade não seja verificada, Pena et al. (2010) propõem um algoritmo (chamado *OP-Search*) com complexidade, no pior caso, de $O(|Q|^3(|\Sigma|+1)^2)$ (Q é o conjunto de estados do autômato e $|\Sigma|$ é a cardinalidade do alfabeto) que busca um conjunto de instâncias de eventos, neste trabalho chamado de Σ_{Ru} , que, ao ser mantido na projeção, torna a mesma uma PO-Abstração.

2.1 Teoria de Controle Supervisório

A linguagem desejada para um sistema em malha fechada é definida como $K = \mathcal{L}(E) \parallel \mathcal{L}(G)$, sendo E a especificação e G a planta em malha aberta. K é controlável com respeito a G se $\bar{K}\Sigma_u \cap \mathcal{L}(G) \subseteq \bar{K}$. Se K não é controlável, então a máxima sublinguagem controlável e não bloqueante $\sup \mathcal{C}(K, G)$ pode ser sintetizada e implementada por um supervisor (Ramadge and Wonham, 1989). S é dito ser não bloqueante se $\mathcal{L}(S/G) = \bar{\mathcal{L}_m}(S/G)$.

2.2 Resultado Anterior

A Teoria de Controle Supervisório proporciona um supervisor S não bloqueante que garante todas as restrições de segurança mas não provê desempenho. A abordagem monolítica contém todas as sequências a serem executadas na planta G , mas determinar qual delas será a ótima não é um processo trivial.

O trabalho de Vilela and Pena (2016) propõe uma solução para reduzir o espaço de busca das soluções (Teorema 1).

Teorema 1 (Vilela and Pena (2016)). *Seja G uma planta, S a máxima sublinguagem controlável contida na linguagem desejada K e $P_{\Sigma \rightarrow \Sigma_c} : \Sigma^* \rightarrow \Sigma_c^*$ a projeção natural nos eventos controláveis. Para toda sequência $s_{ot} \in P_{\Sigma \rightarrow \Sigma_c}(S)$ e $A = P_{\Sigma \rightarrow \Sigma_c}^{-1}(s_{ot}) \cap S$, se $P_{\Sigma \rightarrow \Sigma_c}(S)$ possui a propriedade do observador, então A será controlável com relação a $\mathcal{L}(G)$.*

Essa linguagem A é vista como um novo supervisor, mas que implementa a linguagem composta de todas as cadeias de $\mathcal{L}_m(S/G)$ que projetam em s_{ot} . Se esta linguagem A for controlável, pode-se assegurar a implementação da sequência s_{ot} . A condição principal do Teorema 1 é que a $P_{\Sigma \rightarrow \Sigma_c}(S)$ seja uma PO-abstração.

Antes de apresentar o resultado principal, apresenta-se um exemplo em que as condições para a aplicação do Teorema 1 não estão presentes.

3. EXEMPLO MOTIVADOR

Considere o sistema de manufatura (Zhang et al., 2017) mostrado na Figura 1, composto por 2 máquinas (M_1 e M_2) e uma unidade de teste (TU), conectados por um *buffer* de capacidade 3 (B_1) e um *buffer* unitário (B_2). A máquina M_1 modelada pelo autômato da Figura 2(a) recebe o insumo bruto (evento α_1) e após processá-lo o deposita no *buffer* B_1 (evento β_1). A máquina M_2 retira as peças (evento α_2) de trabalho do *buffer* de entrada B_1 , processa e deposita as mesmas (evento β_2) no *buffer* de saída (B_2). A máquina TU , modelada pelo autômato da Figura 2(b), testa a qualidade das peças (evento α_3) que foram retiradas de B_2 e aprovada (evento β_3) ou rejeita (evento β_4). Desse modo, um insumo bruto deve ser processado pelos 3 subsistemas, em sequência para que torne um produto final. O conjunto de eventos controláveis é $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3\}$ e o conjunto de eventos não controláveis é $\Sigma = \{\beta_1, \beta_2, \beta_3, \beta_4\}$.

Figura 1. Sistema de manufatura com reatrabalho.

A restrição deste problema é a de não haver *underflow* ou *overflow* nos *buffers* B_1 e B_2 modelados pelos autômatos da Figura 3.

Para aplicar o resultado de Vilela and Pena (2016), seria necessário que a projeção natural do comportamento em malha fechada, projetado no conjunto de eventos controláveis, tivesse a propriedade do observador. No caso

Figura 2. Modelos das máquinas.

Figura 3. Modelos das especificações.

deste sistema, não acontece, o que é esperado já que a sequência para produção de uma peça depende de eventos não controláveis (β_3 e β_4), ou seja, $\alpha_1\alpha_2\alpha_3$ produz um produto, caso seja seguido de β_3 . Alternativamente, quando falha o teste (β_4), a produção de um produto é dada pela sequência $\alpha_1\alpha_2\alpha_3\alpha_2\alpha_3$.

A aplicação do algoritmo de busca da propriedade do observador (Pena et al., 2010) retorna que todas as instâncias de β_3 e β_4 devem ser incluídas na projeção, ou seja, $\Sigma_{Ru} = \{\beta_3, \beta_4\}$, de tal forma que $P : \Sigma^* \rightarrow (\Sigma_c \cup \Sigma_{Ru})^*$ é PO-abstração.

4. RESULTADO PRINCIPAL

Neste trabalho pretende-se estender o Teorema 1 para lidar com o caso em que $P_{\Sigma \rightarrow \Sigma_c}$ não é PO-abstração. A estratégia proposta consiste em estender o conjunto de eventos da projeção (Pena et al., 2010), para obter uma PO-abstração e lidar com as modificações necessárias nas definições e teorema, causados pela presença de eventos não-controláveis.

Na sequência estão apresentadas algumas definições que serão usadas no resultado principal deste trabalho.

O sistema de produção, adaptado de (Vilela and Pena, 2016), é apresentado na Definição 2.

Definição 2 (Sistema de Produção). *Seja G um sistema composto por m subsistemas. Cada subsistema é modelado por um autômato $M_i = (Q_i, \Sigma_i, \delta_i, Q_i^m, q_i^0)$, $i \in I = \{1, \dots, m\}$ tal que:*

- i) $G = \parallel_{i=1}^m M_i, \bigcup_{i=1}^m \Sigma_i, \Sigma_i = \Sigma_{i_c} \cup \Sigma_{i_u}$;
- ii) $\Sigma_i \cap \Sigma_j = \emptyset, i, j \in I$ e $i \neq j$;
- iii) M_i é caracterizada por:
 - a) $\forall \sigma \in \Sigma_i$ tal que $\delta(q_i^0, \sigma) = q_a, \sigma \in \Sigma_{i_c}$;
 - b) $\exists w_{i_k} = \sigma_{i_k}^\uparrow t_{i_k}^\downarrow \in \mathcal{L}_m(M_i)$ com $k \geq 1, \sigma_{i_k}^\uparrow \in \Sigma_{i_c}$ e $t_{i_k}^\downarrow \in \Sigma_{i_u}$ tal que $\delta(q_i^0, \sigma_{i_k}^\uparrow) = q_a, \delta(q_a, t_{i_k}^\downarrow) = q_i^0$, e q_a é denominado estado **ativo**.

$$iv) \Sigma_{\sigma\uparrow} = \{\sigma_{i_k}^\uparrow \mid \forall w_{i_k} \in \mathcal{L}_m(M_i), i = 1, \dots, m\}, \Sigma_{t\downarrow} = \{t_{i_k}^\downarrow \mid \forall w_{i_k} \in \mathcal{L}_m(M_i), i = 1, \dots, m\}.$$

A seguinte definição calcula o rótulo de uma composição paralela entre dois autômatos.

Definição 3 (Rótulo de uma Composição). *Seja o autômato $A_i = (Q_i, \Sigma_i, \delta_i, -, -)$, com $i \in I = \{1, \dots, m\}$,*

$A = (Q, -, -, -, -)$ obtido por $A = \parallel_{i=1}^m A_i$, com $q \in$

Q , $q_i \in Q_i$, então $r_A : Q \rightarrow Q_1 \times \dots \times Q_m$, e $P_i : \Sigma^ \rightarrow \Sigma_i^*$ com $q = \delta(q_0, s)$. A função que apresenta o rótulo de cada estado da composição síncrona é: $r_A(q) = ((\delta_1(q_1^0, P_1(s)), \delta_2(q_2^0, P_2(s)), \dots, \delta_m(q_m^0, P_m(s)))$.*

A notação $-$ na tupla indica que o parâmetro correspondente não é relevante para a definição que se apresenta. A função que calcula o rótulo para cada estado do autômato de uma projeção é apresentada na Definição 4.

Definição 4 (Rótulo de uma Projeção). *Seja o autômato $A_p : P_{\Sigma_A \rightarrow \Sigma_B}(C)$ uma projeção natural com a propriedade do observador, em que $A_p = (Q_p, -, \delta_p, q_0^P, -)$, $C = (Q_c, -, \delta_c, q_0^C, -)$. Estabelece-se a função $d'(q) : Q_p \rightarrow 2^{Q_1 \dots Q_c}$ em que o rótulo de uma projeção está definido como: $d'(\delta_p(q_0^P, t)) = \{r_C(q') \mid q' \in \{\delta_c(q_0^C, s) \mid s \in P^{-1}(t) \cap \mathcal{L}(C)\}\}$. A função $d(d'(q))$ converte um conjunto de tuplas em uma tupla de conjuntos.*

Exemplo 1. *Seja o sistema apresentado na Seção 3, que possui 3 equipamentos, M_1, M_2, M_{TU} e $G = M_1 \parallel M_2 \parallel M_{TU}$. Cada um dos sistemas M_i atende as restrições da Definição 2, ou seja, alfabetos assíncronos (Def.2 i,ii), todos os eventos nos estados iniciais são controláveis (Def.2 iii) a), há os seguintes ciclos de trabalho: $w_{11} = \alpha_1\beta_1$, $w_{21} = \alpha_2\beta_2$ e $w_{TU1} = \alpha_3\beta_3$, um em cada máquina (Def. 2 iii) b)). Pode-se ainda definir os conjuntos de eventos $\Sigma_{\sigma\uparrow} = \{\alpha_1, \alpha_2, \alpha_3\}$, e $\Sigma_{t\downarrow} = \{\beta_1, \beta_2, \beta_3\}$, (Def.2 iv)). Seja a composição de M_1 e M_{TU} , obtém-se o autômato $C = (Q_c, -, \delta_c, q_0^C, -)$, da Figura 4(a). O rótulo do estado 2 de C é obtido fazendo $r(2) = (\delta_1(q_1^0, P_1(\alpha_1\alpha_3)), \delta_3(q_3^0, P_3(\alpha_1\alpha_3))) = (\delta_1(q_1^0, \alpha_1), \delta_3(q_3^0, \alpha_3)) = (q_1^1, q_3^1)$, pela Def. 3, sendo $s = \alpha_1\alpha_3$ tal que $\delta(0, s) = 2$. O autômato $A_p = (Q_p, -, \delta_p, q_0^P, -)$ da Fig. 4(b) implementa a linguagem $P_{\Sigma \rightarrow \Sigma_c}(C)$, e esta projeção natural possui a PO. O rótulo do estado 1, $d'(1) = d'(\delta_p(q_0^P, \alpha_3\alpha_1))$ de A_p é obtido identificando, inicialmente, quais estados de C que são alcançados por cadeias que projetam em $\alpha_3\alpha_1$ (no caso, $\alpha_3\alpha_1, \alpha_3\alpha_1\beta_1$, que levam aos estados $\{3, 2\} \subset Q_c$), e depois aplicando o rótulo da composição nestes estados. Assim, $d'(1) = d'(\delta_p(q_0^P, \alpha_3\alpha_1)) = \{r_C(q') \mid q' \in \{3, 2\}\} = \{(q_1^0, q_3^1), (q_1^1, q_3^1)\}$. Por fim, $d(d'(1)) = (\{q_1^0, q_1^1\}, \{q_3^1\})$.*

Figura 4. Exemplo 1: conceitos básicos.

O objetivo desta extensão é tratar de problemas cujo comportamento em malha fechada “depende” de eventos não-controláveis, evidenciada pela necessidade de se incluir eventos não controláveis na projeção. No resultado anterior, escolhia-se uma sequência qualquer da projeção. Nesta extensão, é necessário partir de uma linguagem, $\mathcal{F}^\uparrow$, que será construída por meio de um procedimento apresentado a seguir.

4.1 Algoritmo que Calcula a Linguagem $\mathcal{F}^\uparrow$

O seguinte algoritmo calcula a linguagem $\mathcal{F}^\uparrow$ que será usada no Teorema 2 proposto neste trabalho. O algoritmo consiste de uma série de passos e procedimentos para calcular a linguagem $\mathcal{F}^\uparrow$, análoga à cadeia s_{ot} usada no Teorema 1.

O primeiro passo deste procedimento consiste na construção da linguagem composta de todas as cadeias que produzem um número n de produtos, pela composição da PO-abstração com a especificação de quantidade, E_q . No caso do problema apresentado na Seção 3, entre as cadeias estão aquelas que não passaram no teste e tiveram que ser retrabalhadas pelo menos uma vez. Escolhe-se, dentre elas, alguma cadeia formada só dos eventos componentes dos ciclos de trabalho das máquinas, ou seja, $(\Sigma_{\sigma\uparrow} \cup \Sigma_{t\downarrow})^*$, projeta-se nos eventos controláveis obtendo-se s_{ot} e um autômato que reconhece $\{s_{ot}\}$. A seguir, os passos do algoritmo são apresentados.

- 1) **Encontrar uma sequência s_{ot} :**
 - (i) $T_{E_q} = \text{Trim}(P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S) \parallel E_q)$, E_q representa a especificação de quantidade de n produtos;
 - (ii) $s \in T_{E_q} \cap (\Sigma_{\sigma\uparrow} \cup \Sigma_{t\downarrow})^*$;
 - (iii) $s_{ot} = P_{\Sigma_s \rightarrow \Sigma_c}(s)$;
 - (iv) $\mathcal{L}_m(G_{s_{ot}}) \leftarrow \{s_{ot}\}$, $G_{s_{ot}} = (Q_{s_{ot}}, \Sigma_{\sigma\uparrow}, \delta, q_0, Q_m)$.

O segundo passo consiste em adicionar auto-laços de alguns eventos em $G_{s_{ot}}$, sem tornar o mesmo não-determinístico. Os eventos de $\Sigma_u \setminus \Sigma_{t\downarrow}$ são eventos que podem ocorrer no sistema mas não contribuem para a produção e, cada um deles, deve ter um evento controlável (que deve ser identificado por inspeção do projetista), não necessariamente da mesma planta, que retoma a produção. Este conjunto de eventos controláveis é denotado por Σ_{fix}^C .

- 2) **Auto-laços de Σ_{fix}^C e Σ_{Ru} em $G_{s_{ot}}$:**
 - (i) $G_{s_{ot}}' = \{(\forall q_i \in Q_{s_{ot}})(\forall b \in \Sigma_{fix}^C \cup \Sigma_{Ru}), \text{ se } b \notin \Gamma(q_i) \implies \delta(q_i, b) = \{q_i\}\}$;
 - (ii) $G_{s_{ot}}' = (Q, \Sigma_{s_{ot}}', \delta, q_0^{s_{ot}}, Q_m)$.

Seja um conjunto $M^{Ru} = \{M_i \mid \Sigma_i \cap \Sigma_{Ru} \neq \emptyset\}$. O terceiro passo consiste em identificar, a cada estado de $G_{s_{ot}}'$, quais são as máquinas de $M_i \in M^{Ru}$ que estão em estados ativos, pela aplicação da função $d(d'(q))$, $q \in Q_{s_{ot}}$, e acrescentar, nestes estados, auto-laços com os eventos $\sigma_{i_k}^\uparrow$.

- 3) **Auto-laços do evento $\sigma_i^\uparrow$ em cada estado $q \in Q_{s_{ot}}$ em que o subsistema $M_i \in M^{Ru}$ está no estado ativo q_a :**
 - (i) Redefine δ de $G_{s_{ot}}'$, para incluir $\delta(q, \sigma_{i_k}^\uparrow) = q$, se o i -ésimo elemento da tupla de conjuntos obtida

fazendo $d(d'(q))$ inclui o estado $q_a \in Q_i$ e $M_i \in M^{Ru}$;
(ii) $\mathcal{F}^\dagger \leftarrow L(G_{s_{ot}}')$.

Após apresentar o procedimento para obtenção da linguagem $\mathcal{F}^\dagger$, enuncia-se o Teorema 2, que estende o teorema original de Vilela and Pena (2016).

Teorema 2. *Seja G um sistema, S a máxima sublinguagem controlável contida na linguagem desejada de K e $P : \Sigma^* \rightarrow (\Sigma_c \cup \Sigma_{Ru})^*$ a projeção natural com a propriedade do observador. Para toda linguagem $\mathcal{F}^\dagger \subseteq (\Sigma_c \cup \Sigma_{Ru})^*$ e $A = P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}^{-1}(\mathcal{F}^\dagger) \cap S$, como $P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S)$ tem a propriedade do observador, então A será controlável com relação a $\mathcal{L}(G)$.*

A prova do teorema proposto neste trabalho segue o mesmo caminho da prova feita para o Teorema 1 em Vilela and Pena (2016).

5. EXEMPLO DE APLICAÇÃO

Considere o sistema de manufatura (Zhang et al., 2017) apresentado na seção 1. A síntese do supervisor monolítico é realizada para os modelos e especificações das Figs. 2 e 3, e tem 28 estados e 65 transições.

Conforme mencionado na Seção 3, $P_{\Sigma \rightarrow \Sigma_c}(S)$ não é PO-abstração. Usando o algoritmo de (Pena et al., 2010), obtém-se $\Sigma_{Ru} = \{\beta_3, \beta_4\}$ e $P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S)$ é PO-abstração.

Para exemplificar a aplicação do procedimento, compõe-se o autômato do comportamento em malha fechada com uma especificação (E_q , na Figura 5) que implementa a cadeia $\beta_3\beta_3$, que restringe o comportamento somente às cadeias que geram dois produtos.

O autômato resultante da composição é denominado $T_{E_q} = Trim(P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S) || E_q)$ e é apresentado na Figura 6.

Figura 5. Autômato E_q que implementa a produção de 2 produtos.

Conforme mencionado na Seção 3, $P_{\Sigma \rightarrow \Sigma_c}(S)$ não é PO-abstração. Usando o algoritmo de Pena et al. (2010), obtém-se $\Sigma_{Ru} = \{\beta_3, \beta_4\}$. O autômato resultante da composição é denominado $T_{E_q} = Trim(P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S) || E_q)$ e é apresentado na Figura 6.

A inclusão dos eventos não controláveis faz com que ciclos possam se formar (no caso, quando ocorre o evento β_4 , que retorna o produto para a máquina M_2). Como estes eventos, β_3 e β_4 , são não-controláveis, não é possível prevenir que eles aconteçam e sim deve-se lidar com as possibilidades. Os estados em cinza são aqueles que possuem os dois eventos β_3 e β_4 possíveis. Considerando o equipamento M_{TU} , há um ciclo $w_{TU} = \alpha_3\beta_3$, $\sigma_{TU}^\dagger = \alpha_3$ e $t_{TU}^\dagger = \beta_3$.

O primeiro passo é escolher uma sequência $s \in \mathcal{L}_m(E_q) \cap (\Sigma_{\sigma^\dagger} \cup \Sigma_{t^\dagger})^*$, ou seja, que contenha apenas eventos que contribuem para a produção. Embora a linguagem $\mathcal{F}^\dagger$

Figura 6. Autômato $T_{E_q} = Trim(P_{\Sigma \rightarrow (\Sigma_c \cup \Sigma_{Ru})}(S) || E_q)$.

seja infinita, o número de cadeias que só possuem estes eventos é finito, no caso específico, são sete sequências: $s_1 = \alpha_1\alpha_2\alpha_3\alpha_1\beta_3\alpha_2\alpha_3\beta_3$, $s_2 = \alpha_1\alpha_2\alpha_3\beta_3\alpha_1\alpha_2\alpha_3\beta_3$, $s_3 = \alpha_1\alpha_2\alpha_1\alpha_3\beta_3\alpha_2\alpha_3\beta_3$, $s_4 = \alpha_1\alpha_1\alpha_2\alpha_3\beta_3\alpha_2\alpha_3\beta_3$, $s_5 = \alpha_1\alpha_1\alpha_2\alpha_3\alpha_2\beta_3\alpha_3\beta_3$, $s_6 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\beta_3\alpha_3\beta_3$, $s_7 = \alpha_1\alpha_2\alpha_3\alpha_1\alpha_2\beta_3\alpha_3\beta_3$.

A escolha de uma dessas cadeias busca minimizar alguma função objetivo (makespan, consumo energético etc). A cadeia escolhida, quando executada na planta deve executar até o final, ou seja, a espontaneidade dos eventos não controláveis não deve causar infactibilidade (que se manifesta pela controlabilidade da linguagem A dos teoremas 1 e 2). No caso desta sistema, qualquer uma destas cadeias não gerará uma linguagem A controlável já que não é possível impedir β_4 de ocorrer nos estados em cinza da Figura 6.

Para ilustrar a construção da linguagem $\mathcal{F}^\dagger$, escolhe-se uma das cadeias de comandos que produzem 2 produtos:

$$s_6 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\beta_3\alpha_3\beta_3.$$

Faz-se a projeção da cadeia no conjunto dos eventos controláveis ($s_{ot} = P_{\Sigma \rightarrow \Sigma_c}(s_6)$) e obtém-se o autômato $G_{s_{ot}}$ que implementa a linguagem $\{s_{ot}\}$.

Obtém-se $G_{s_{ot}}'$ pela inclusão de auto-laços de $\Sigma_{Ru} = \{\beta_3, \beta_4\}$ e $\Sigma_{fix}^C = \{\alpha_2\}$, Figura 7. Evento α_2 retorna a peça para a produção, e por isso é o evento de Σ_{fix}^C .

Figura 7. Autômato $G_{s_{ot}}'$ com auto-laços de eventos dos conjuntos Σ_{fix}^C e Σ_{Ru} .

Para realizar o passo 3) do algoritmo para construção de $\mathcal{F}^\dagger$ utiliza-se um conjunto $M^{Ru} = \{M_{TU}\}$. A aplicação de $d(d'(q))$ é detalhada na Tabela 1

Segundo o passo 3 do procedimento, deve-se ter α_3 em todos os estados de $G_{s_{ot}}'$, em que M_{TU} estiver no estado ativo, ou seja, nos estados em que o terceiro elemento da tupla de conjuntos possui o estado q_1^3 . Percebe-se pela Tabela 1 que são os estados $q = 4$, $q = 5$ e $q = 6$. Como no estado $q = 5$ α_3 já está factível originalmente, coloca-se auto-laços apenas nos estados 4 e 6 (em cinza). O autômato resultante é apresentado na Figura 8. A linguagem marcada pelo autômato resultante

Tabela 1. Passo 3): aplicação de $d'(q)$ e $d(d'(q))$.

q	$d'(q)$	$d(d'(q))$
0	$\{(q_0^1, q_0^2, q_0^3)\}$	$(\{q_0^1\}, \{q_0^2\}, \{q_0^3\})$
1	$\{(q_1^1, q_0^2, q_0^3), (q_0^1, q_0^2, q_0^3)\}$	$(\{q_1^1, q_0^1\}, \{q_0^2, q_0^2\}, \{q_0^3, q_0^3\})$
2	$\{(q_0^1, q_1^2, q_0^3), (q_0^1, q_0^2, q_0^3)\}$	$(\{q_0^1, q_0^1\}, \{q_1^2, q_0^2\}, \{q_0^3, q_0^3\})$
3	$\{(q_1^1, q_0^2, q_0^3), (q_0^1, q_0^2, q_0^3)\}$	$(\{q_1^1, q_0^1\}, \{q_0^2, q_0^2\}, \{q_0^3, q_0^3\})$
4	$\{(q_0^1, q_0^2, q_1^3)\}$	$(\{q_0^1\}, \{q_0^2\}, \{q_1^3\})$
5	$\{(q_0^1, q_1^2, q_1^3), (q_0^1, q_1^2, q_0^3), (q_0^1, q_0^2, q_0^3)\}$	$(\{q_0^1, q_0^1, q_0^1\}, \{q_1^2, q_1^2, q_0^2\}, \{q_1^3, q_0^3, q_0^3\})$
6	$\{(q_0^1, q_0^2, q_1^3), (q_0^1, q_0^2, q_0^3)\}$	$(\{q_0^1, q_0^1\}, \{q_0^2, q_0^2\}, \{q_1^3, q_0^3\})$

é $\mathcal{F}^\dagger$, e pode ser usada no Teorema 2. A linguagem $\mathcal{F}^\dagger$

Figura 8. Autômato $G'_{s_{ot}}$ com auto-laços do evento $\sigma_{ik}^\dagger$.

deve ser implementada no sistema visando executar a cadeia otimizada $s_6 = \alpha_1\alpha_2\alpha_1\alpha_3\alpha_2\beta_3\alpha_3\beta_3$. Não se pode implementar somente s_6 pois, após cada ocorrência de α_3 , existe a possibilidade de ocorrer β_4 no lugar de β_3 , ambos não-controláveis. Caso ocorra β_3 , segue-se a produção com o restante da cadeia s_6 , mas se ocorrer β_4 , é preciso que o sistema seja capaz de lidar com a situação. A introdução dos eventos Σ_C^{fix} e dos auto-laços com $\sigma_{ik}^\dagger$, garantem o retorno do sistema para a situação de produção e, portanto, garantem a controlabilidade da linguagem resultante A .

6. CONCLUSÃO

Apresenta-se neste trabalho uma extensão do Teorema Vilela and Pena (2016) para o caso em que a linguagem que implementa a projeção para o conjunto de eventos controláveis do comportamento em malha fechada não possui a propriedade do observador. A inexistência da PO demonstra a importância para a produção de alguns eventos não-controláveis. Desta forma, estende-se a abstração para incluir tais eventos. Esta inclusão transforma a cadeia otimizada em uma linguagem a ser implementada no sistema.

Para obtenção de tal linguagem, $\mathcal{F}^\dagger$, parte-se de uma cadeia otimizada, escolhida entre aquelas que produzem um número de produtos, pressupondo que eventos não controláveis específicos vão ocorrer. Em seguida, constrói-se a linguagem que garante que mesmo que os não-controláveis previstos não ocorram, é possível finalizar a produção.

Uma proposta de continuidade consiste na implementação de um algoritmo que busca a cadeia s que será usada para obter $\mathcal{F}^\dagger$ e a integração desta solução com um protótipo de sistema de manufatura disponível no LACSED/UFGM.

REFERÊNCIAS

- Alves, L.V., Bravo, H.J., Pena, P.N., and Takahashi, R.H. (2016). Planning on discrete events systems: A logical approach. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 1055–1060. IEEE.
- Arisha, A., Young, P., and El Baradie, M. (2001). Job shop scheduling problem: an overview. In *International Conference for Flexible Automation and Intelligent Manufacturing*, 682 – 693.
- Cassandras, C.G. and Lafortune, S. (2009). *Introduction to discrete event systems*. Springer Science & Business Media.
- Huang, J. and Kumar, R. (2007). Optimal nonblocking directed control of discrete event systems. In *2007 American Control Conference*, 4285–4290. IEEE.
- Nishi, T. and Wakatake, M. (2014). Decomposition of timed automata for solving scheduling problems. *International Journal of Systems Science*, 45(3), 472–486.
- Pena, P.N., Bravo, H.J., da Cunha, A.E., Malik, R., Lafortune, S., and Cury, J.E. (2014). Verification of the observer property in discrete event systems. *IEEE Transactions on Automatic Control*, 59(8), 2176–2181.
- Pena, P.N., Cury, J.E., Malik, R., and Lafortune, S. (2010). Efficient computation of observer projections using op-verifiers1. *IFAC Proceedings Volumes*, 43(12), 406–411.
- Pinha, D.C., de Queiroz, M.H., and Cury, J.E. (2011). Optimal scheduling of a repair shipyard based on supervisory control theory. In *2011 IEEE International Conference on Automation Science and Engineering*, 39–44. IEEE.
- Rafael, G. and Pena, P. (2018). Using an abstraction of the supervisor to solve a planning problem in manufacturing systems. In *Anais do XXII Congresso Brasileiro de Automática, CBA*.
- Ramadge, P.J. and Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98.
- Su, R. (2012). Abstraction-based synthesis of timed supervisors for time-weighted systems. *IFAC Proceedings Volumes*, 45(29), 128–134.
- Teixeira, M., Cury, J.E., and de Queiroz, M.H. (2013). Local modular control with distinguishers applied to a manufacturing system. *IFAC Proceedings Volumes*, 46(9), 263–268.
- Vilela, J.N. and Pena, P.N. (2016). Supervisor abstraction to deal with planning problems in manufacturing systems. In *13th International Workshop on Discrete Event Systems (WODES'16)*, pp.117–122. IEEE.
- Wong, K. (1998). On the complexity of projections of discrete-event systems. In *International Workshop on Discrete Event Systems, WODES 1998*, pp.201–206.
- Zhang, R., Cai, K., and Wonham, W.M. (2017). Supervisor localization of discrete-event systems under partial observation. *Automatica*, 81, 142–147.