Detecção de falhas descentralizada em uma planta mecatrônica usando modelos identificados

Caio E.O. Lacoste * **Diego A. Libanio** * **Gustavo S. Viana** * **Marcos V. Moreira** *

* *Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, RJ (e-mails; caio.lacoste@poli.ufrj.br; diego.alibanio@poli.ufrj.br; gustavo.viana@poli.ufrj.br; moreira.mv@poli.ufrj.br)*

Abstract: Recently, techniques for Discrete Event Systems (DES) identification with the objective of fault detection have been proposed in the literature. DES identification consists of computing a monolithic model capable of simulating the observed fault-free behavior generated by the system. Thus, a fault is detected when a discrepancy is observed between the evolution of the system and the estimation of the model. However, when working with systems whose information is distributed, it is necessary to implement a decentralized fault diagnosis architecture in which local diagnosers are computed based on their own observations. Therefore, this paper proposes a decentralized fault detection architecture using identified models. As a motivational example, a mechatronic plant system with decentralized control is proposed and used to verify the proposed method. The formalism used to represent the local models of each subsystem is the Deterministic Automaton with Outputs and Conditional Transitions (DAOCT). Finally, intermittent and permanent faults are simulated in the equipment of the system to verify the efficiency of the proposed method in detecting faults.

Resumo: Recentemente, técnicas de identificação de Sistemas a Eventos Discretos (SEDs) com o objetivo de detectar falhas foram propostas na literatura. A identificação de SEDs consiste em computar um modelo monolítico capaz de simular o comportamento observado livre de falhas gerado pelo sistema. Assim, uma falha é detectada quando é observada uma discrepância entre a evolução do sistema e a estimativa do modelo. Entretanto, quando as informações do sistema são distribuídas, é necessário implementar uma arquitetura de diagnóstico de falhas descentralizada, na qual os diagnosticadores locais são construídos com base em suas próprias observações. Dessa forma, neste artigo é proposta uma arquitetura descentralizada de detecção de falhas usando modelos identificados. Como exemplo motivador, um sistema de uma planta mecatrônica com controle descentralizado é estudado e utilizado para testar o método proposto. O formalismo utilizado para representar os modelos locais de cada subsistema da planta é o modelo de *Deterministic Automaton with Outputs and Conditional Transitions* (DAOCT). Por fim, são simuladas falhas intermitentes e permanentes nos equipamentos do sistema para verificar a eficiência do método proposto em detectar falhas.

Keywords: Identification; Fault detection; Discrete-event systems; Finite state automata; Black-box identification.

Palavras-chaves: Identificação; Detecção de falhas; Sistemas a eventos discretos; Autômatos de estado finito; Identificação caixa-preta.

1. INTRODUÇÃO

Indústrias modernas estão em crescente evolução no mercado, expandindo e aumentando sua produção através da automação de seus processos. Dessa maneira, o desenvolvimento de métodos para detecção eficaz de falhas é vital para o funcionamento correto do sistema, o que, caso contrário, pode comprometer toda a linha de produção. No contexto de Sistemas a Eventos Discretos (SEDs), o estudo de diagnóstico de falhas foi introduzido em Sampath et al. (1995), a partir da propriedade de diagnosticabilidade. Assim, uma falha é dita diagnosticável se, e somente se, é possível detectar e isolar sua ocorrência após número limitado de eventos. A partir disso, diversos estudos foram publicados apresentando diferentes estratégias de diagnóstico de falhas (Zaytoon and Lafortune, 2013; Moreira et al., 2011; Viana et al., 2019). Porém, nesses trabalhos, é suposto que o comportamento do sistema, antes e após a ocorrência da falha, é conhecido, o que pode ser uma tarefa difícil para sistemas industriais complexos.

Com o objetivo de superar essa dificuldade, técnicas de identificação de sistemas, voltadas para detecção de falhas, foram propostas na literatura (Klein et al., 2005; Roth et al., 2010; Cabasino et al., 2015; Moreira and Lesage, 2019b,a). A identificação de sistemas com objetivo de diagnóstico de falhas consiste em computar um modelo

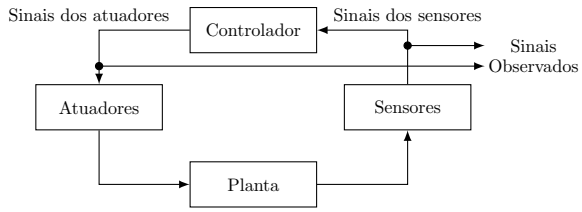

Figura 1. Sistema malha fechada.

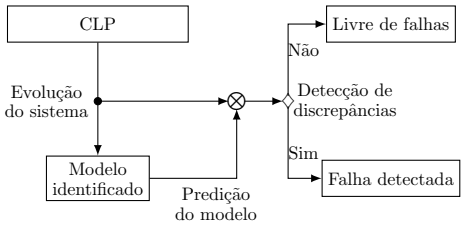

Figura 2. Arquitetura do esquema de detecção de falhas.

livre de falhas do sistema, obtido através da observação dos sinais dos sensores e atuadores do controlador, de acordo com a Figura 1. Após a obtenção do modelo, um detector de falhas observa, simultaneamente, a evolução do sistema e as predições do modelo identificado. Assim, uma falha é detectada quando o sistema realiza algum comportamento que não pertença à linguagem gerada pelo modelo identificado, como ilustrado na Figura 2.

Os trabalhos propostos na literatura, que tratam do problema de detecção de falhas utilizando modelos identificados, utilizam uma arquitetura centralizada. Neste cenário, um modelo monolítico é construído por identificação e utilizado para detecção de falhas, em que todos os sinais de entrada e saída do controlador são obtidos de um único controlador lógico programável (CLP). Caso o sistema a ser identificado possua muitos comportamentos concorrentes, a tendência é que o modelo monolítico se torne muito grande e o processo de identificação fique praticamente impraticável em tempo hábil. Entretanto, ainda não foram abordadas técnicas de identificação com objetivo de detecção de falhas para SEDs, com informações descentralizadas e múltiplos controladores. Isto é, os sinais de sensores e atuadores não são provenientes de um único controlador, mas de vários controladores locais, reduzindo consequentemente o tamanho dos modelos.

Neste artigo, é adaptada a arquitetura descentralizada de detecção de falhas apresentada no protocolo 3 de Debouk et al. (2000), com o objetivo de aplicação em uma planta mecatrônica real, cujo controle é distribuído em 4 controladores separados. Nesse caso, utilizando modelos locais ao invés de apenas um monolítico, podemos detectar falhas a partir um modelo apresentado em Moreira and Lesage (2019a), chamado *Deterministic Automaton with Outputs and Conditional Transitions* (DAOCT). Dessa forma é analisada, através da simulação de falhas no sistema, quão eficiente é tal metodologia.

Este artigo está organizado da seguinte forma. Na Seção 2, são introduzidos conceitos preliminares. Na Seção 3, a planta mecatrônica montadora de cubos é apresentada. Na Seção 4, a arquitetura descentralizada de detecção de falhas é explicada. Na Seção 5, é proposta a solução para o exemplo prático, exibindo os resultados da aplicação da arquitetura descentralizada. Na Seção 6, conclusões são apresentadas.

2. CONCEITOS PRELIMINARES

2.1 Notações

Seja $G = (X, \Sigma, f_{nd}, x_0, X_m)$ um autômato não determinístico, em que X é o conjunto de estados, Σ é o conjunto de eventos, $f_{nd} : X \times \Sigma \to 2^X$ é a função de transição não determinística, x_0 é o estado inicial e X_m é o conjunto de estados marcados. Considere o conjunto $\Sigma^* = \{\varepsilon\} \cup \Sigma \cup \Sigma\Sigma \cup \Sigma\Sigma\Sigma \cup \ldots$, em que Σ^* é denominado como fecho de Kleene.

O conjunto de números inteiros não negativos é denotado por $\mathbb{N}$ e o conjunto formado apenas pelos números 0 e 1, é denotado por $\mathbb{N}_1 = \{0, 1\}$.

O Ou Exclusivo (XOR), denotado como $\oplus$, é uma operação lógica em que $1 \oplus 1 = 0 \oplus 0 = 0$ e $1 \oplus 0 = 1$. A operação XOR pode ser aplicada a dois vetores de números binários x_1 e x_2 se o número de elementos de x_1 e x_2 forem iguais. Nesse caso, o vetor $x = x_1 \oplus x_2$ é definido de forma que a i-ésima entrada de x seja computada como $x_i = x_{1,i} \oplus x_{2,i}$, em que $x_{j,i}$ é o i-ésimo elemento do vetor x_j, para $j = 1, 2$.

2.2 Identificação de Sistemas a Eventos Discretos

Considere o sistema malha fechada ilustrado na Figura 1, em que n_I e n_O denotam o número de entradas (*inputs*) e saídas (*outputs*) binárias do CLP, respectivamente. O vetor formado pelos sinais de entrada e saída do CLP em um instante de tempo $t_k \in \mathbb{R}$, chamado neste artigo de vetor de I/O, é denotado como $u_k = [I_1(k)I_2(k)\ldots I_{n_I}(k)O_1(k)O_2(k)\ldots O_{n_O}(k)]$, em que $I_\beta(k)$ e $O_\delta(k)$, para $\beta \in \{1, 2, \ldots, n_I\}$ e $\delta \in \{1, 2, \ldots, n_O\}$ são, respectivamente, os sinais de entrada e saída do controlador no instante de tempo t_k. Considere que u_k é um vetor binário, *i.e.*, $u_k \in \mathbb{N}_1^n$, em que $n = n_I + n_O$. Conforme o sistema evolui, os sinais do vetor I/O do sistema pode mudar devido a alterações nas leituras dos sensores ou nos comandos dos atuadores.

Se uma sequência de l vetores I/O, u_1, u_2, ..., u_l, for observada, é possível descrever um caminho observado do sistema $p = (u_1, u_2, \ldots, u_l)$. Considere $p_j = (u_{j,1}, u_{j,2}, \ldots, u_{j,l_j})$, para $j = 1, \ldots, |\mathcal{P}|$, caminhos observados que representam tarefas completas executadas pelo sistema livre de falhas, em que $\mathcal{P}$ é o conjunto formado por todos os caminhos observados distintos. Logo, o comportamento observado do sistema livre de falhas pode ser descrito a partir de todas os caminhos observados de vetores de I/O, p_j, executadas pelo sistema. Neste artigo, é feita a seguinte hipótese sobre os caminhos observados.

***Hipótese* 1.** *Todos os caminhos do sistema livre de falhas começam com o mesmo vetor I/O, i.e.,* $\nu_{i,1} = \nu_{j,1}$*, para todos os* $p_i, p_j \in \mathcal{P}$*, e são cíclicos, i.e., para todos os* $p_j \in \mathcal{P}$*,* $\nu_{j,1} = \nu_{j,l_j}$*.*

A identificação de Sistemas a Eventos Discretos com o objetivo de detecção de falhas consiste em computar um modelo que simula a linguagem observada do sistema livre de falhas (Roth et al., 2010; Klein et al., 2005). Após obter o modelo identificado, a estratégia de detecção de

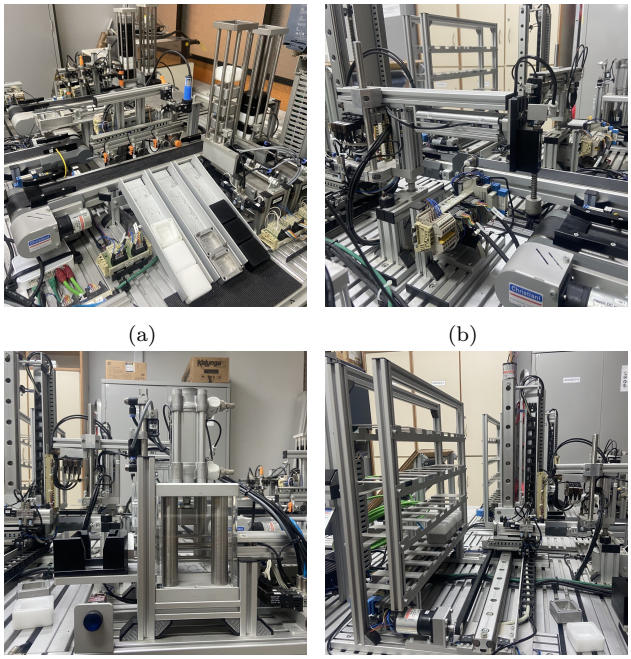

Figura 3. Exemplo prático de planta industrial controlada por 4 CLPs. a) Módulo da esteira, b) Módulo do braço mecânico c) Módulo da prensa, e d) Módulo do armazenador.

falhas consiste em detectar uma falha por meio de uma discrepância entre o comportamento observado do sistema e o comportamento esperado do modelo identificado. Esta arquitetura está descrita na Figura 2. Caso o vetor I/O observado for diferente de todos os possíveis vetores I/Os alcançáveis pelo estado atual do modelo do sistema, uma falha será detectada.

Note que a arquitetura de detecção de falhas ilustrada na Figura 2 representa uma arquitetura centralizada. No entanto, quando o sistema é controlado por vários controladores locais que observam parcialmente os sinais do sistema, e as informações do sistema são descentralizadas, a identificação de um modelo monolítico para representar o sistema torna-se inviável. Neste artigo, será estudada a identificação de uma planta mecatrônica montadora de cubos, cujo controle é descentralizado. Dessa forma, a contribuição deste trabalho consiste em propor uma arquitetura descentralizada para detecção de falhas capaz de ser aplicada ao sistema proposto. Assim, na seção seguinte, o sistema e seus respectivos módulos são apresentados.

3. PLANTA MECATRÔNICA COM CONTROLE DESCENTRALIZADO

Considere a planta industrial cujos módulos são ilustrados na Figura 3. O sistema é composto por quatro módulos, com quatro etapas sequenciais com o objetivo de montar uma caixa, composta de tampa e base, sendo que a tampa pode ser somente branca e a base somente metálica. Os quatro módulos serão explicados em mais detalhes a seguir.

O primeiro módulo é a seleção de peças, que consiste em uma esteira, dois buffers de entrada (um para tampas e outro para bases), sensores e prensas hidráulicas. Seu objetivo é garantir que a ordem e a peça correta continuem para a etapa da montagem. O sistema começa enviando uma base do buffer para a esteira. Assim, a esteira começa a se mover, até chegar no ponto em que os sensores indutivo, capacitivo e de profundidade identificam se a peça é a correta para continuar no processo, ou seja, é uma base de material metálico. No caso da peça não ser a correta, os atuadores hidráulicos ao longo da esteira são utilizados para expelir essas peças indesejadas. Existe um atuador para cada cor de peça, de forma que peças rejeitadas possam ser organizadas em rampas separadas de descarte. No caso da peça ser a correta, a esteira leva ela até o final do percurso, onde o sensor de fim de curso indica que está pronta para ser usada na próxima etapa.

O segundo módulo consiste do braço mecânico, responsável por transportar a peça entre os outros três módulos da planta. Seu funcionamento é baseado em um *encoder* rotativo presente em seu eixo. Portanto, para garantir o posicionamento correto, o braço inicialmente volta para a posição inicial, identificada com um sensor indutivo e depois vai até a peça. Tendo buscado a peça, seu próximo destino é o terceiro módulo, composto pela prensa hidráulica. Depois de enviar uma peça de base e uma peça de tampa para o terceiro módulo, o braço aguarda, na direção da prensa, a montagem da caixa, para logo depois pegá-la e transportá-la para o quarto módulo, o armazenador.

O terceiro módulo consiste em um sistema da prensa. Após receber duas peças corretas, o recipiente da prensa se retrai e fecha suas portas. Ao fechar a comporta, o atuador hidráulico se estende e pressiona as peças, de forma que os pinos da base se fixem na tampa e forme a caixa completa. Depois do atuador retrair, o recipiente sai para a mesma posição de início, preparado para o braço pegar a caixa pronta.

Por fim, o quarto módulo consiste no armazenador de caixas. Nessa seção, o objetivo é guardar as caixas em uma estante, de maneira a preencher da direita para esquerda. Para saber a posição correta do armazenador, são utilizados contadores na esteira horizontal correspondentes a cada um dos espaços para caixas, totalizando sete posições disponíveis. Ao chegar na posição correta, o armazenador sobe um pouco, estende seu braço e desce, deixando a caixa na estante, assim retraindo o braço pela abertura embaixo de cada posição.

O controle da planta industrial é realizado por quatro controladores lógicos programáveis (CLPs), um para cada módulo citado. Além disso, os sinais de sensores e atuadores observados estão distribuídos em quatro sites locais distintos, o que impossibilita a identificação de um modelo monolítico do sistema que observe todos os sinais simultaneamente. Assim, para realizar a identificação do sistema proposto, é introduzida neste trabalho uma arquitetura descentralizada para detecção de falhas, através de modelos locais descrevendo o comportamento livre de falhas de cada módulo descrito nesta seção.

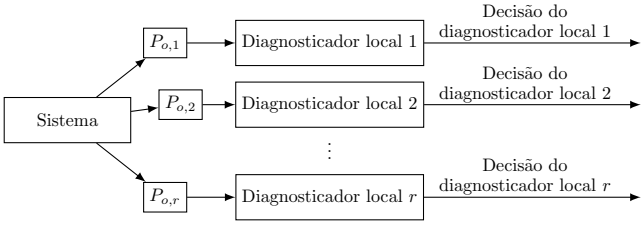

Figura 4. Arquitetura clássica de diagnóstico de falhas descentralizado.

4. IDENTIFICAÇÃO DESCENTRALIZADA COM OBJETIVO DE DETECÇÃO DE FALHAS

Em Debouk et al. (2000), assume-se que as informações do sistema são distribuídas em um determinado número $r \in \mathbb{N}$ de sites locais. Cada site local A_i, para $i = 1, \ldots, r$, tem uma observação parcial do sistema, em que $\Sigma_{o,i}$ e $\Sigma_{uo,i}$ são, respectivamente, os conjuntos de eventos observáveis e não observáveis para o site local A_i. No Protocolo 3 apresentado em Debouk et al. (2000), um diagnosticador é implementado para cada local A_i, sendo computado com base no modelo completo do sistema. Nessa arquitetura, os diagnosticadores locais não se comunicam entre si e uma falha é considerada codiagnosticável quando pelo menos um site local é capaz de detectar e isolar qualquer ocorrência de falha dentro de um número limitado de ocorrências de eventos após a falha. A definição formal de codiagnosticabilidade da linguagem é definida a seguir (Debouk et al., 2000).

***Definição* 1.** *Considere G o autômato que modela o sistema completo, e L_G a linguagem viva gerada pelo autômato G. Então, L_G é codiagnosticável com relação às projeções $P_{o,i} : \Sigma^* \to \Sigma^*_{o,i}$, $i = 1, \ldots, r$ e Σ_f, se e somente se,*

$$(\exists z \in \mathbb{N})(\forall s \in L_G \setminus L_N)(\forall st \in L_G \setminus L_N)(||t|| \geq z \Rightarrow$$
$$(\exists i \in \{1, ..., r\})[P_{o,i}(st) \cap P_{o,i}(\omega) = \emptyset, \forall \omega \in L_N]). \quad (1)$$

Na Figura 4 é ilustrada a arquitetura clássica de diagnóstico de falhas descentralizada. Note que, nessa arquitetura, o conhecimento completo do modelo do sistema, consistindo tanto no comportamento livre de falha, como o comportamento pós-falha, é necessário para computar os diagnosticadores locais. Portanto, se o comportamento completo do sistema não for conhecido, o método clássico de diagnóstico de falhas não poderá ser aplicado.

Nessa seção, é apresentado o problema da detecção de falhas descentralizada utilizando modelos parciais. Partindo do pressuposto que o sistema é regido por diversos controladores locais CLP_i, para $i = 1, \ldots, r$, cada modelo local livre de falhas $\mathcal{G}_i$ é identificado usando os vetores I/O dos controladores CLP_i, como apresentado na Figura 5. Assim, um detector local de falhas, $\mathcal{D}_i$, é computado para cada local A_i. Como em Debouk et al. (2000), assume-se que não existe comunicação entre os controladores, em que uma falha é detectada quando, pelo menos, um dos diagnosticadores locais observa uma sequência de vetores I/O que seja diferente da esperada pelo modelo livre de falhas, como mostrado na Figura 6.

Na detecção de falhas descentralizada proposta nesse artigo, os diagnosticadores observam os vetores I/O com-

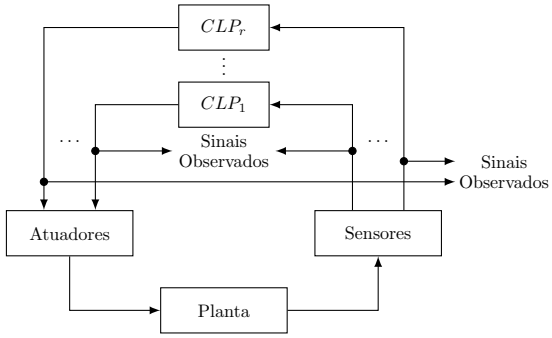

Figura 5. Sistema a eventos discretos malha fechada descentralizado.

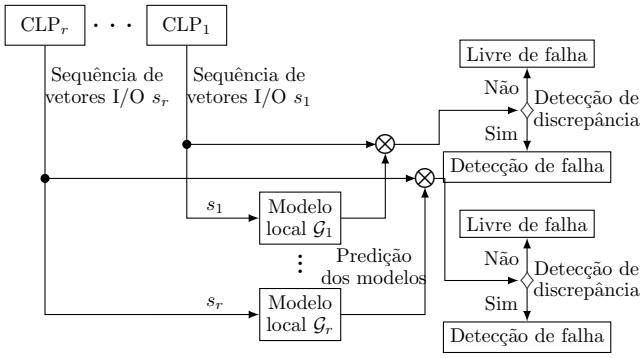

Figura 6. Arquitetura descentralizada de detecção de falhas usando modelos identificados.

postos pelos sinais de sensores e atuadores, enquanto que em Debouk et al. (2000) os eventos são diretamente enviados para os diagnosticadores. Para associar os vetores I/O com os eventos, é intuitivo considerar mudanças em pelo menos um sinal de sensor ou atuador como um evento. Para isso, considere $u^i_k = [I^i_1(k)\ I^i_2(k) \ldots I^i_{n_{I,i}}(k)\ O^i_1(k)\ O^i_2(k) \ldots O^i_{n_{O,i}}(k)]$, o vetor I/O observado pelo site local A_i, em que I^i_β e O^i_δ, para $\beta \in \{1, 2, \ldots, n_{I,i}\}$ e $\delta \in \{1, 2, \ldots, n_{O,i}\}$, são os conjuntos de sinais de entradas e saídas do controlador CLP_i, respectivamente, e $n_{I,i}$ e $n_{O,i}$ são a quantidade de entradas e saídas do CLP_i, respectivamente. Portanto, $u^i_k \in \mathbb{N}^{n_i}_1$, em que $n_i = n_{I,i} + n_{O,i}$. Caso dois vetores I/O consecutivos u^i_1 e u^i_2 sejam diferentes, o evento associado a mudança de sinais pode ser definido como um vetor $e^i = u^i_1 \oplus u^i_2$, denotado neste trabalho como vetor evento. É possível notar que um elemento de e^i é igual a 1 se, e somente se, o sinal correspondente nos vetores u^i_1 e u^i_2 se alterar.

Com o objetivo de relacionar a ocorrência de um evento da planta com sua observação no subsistema local, é introduzida a função de seleção de vetores $S_i : \mathbb{N}^n_1 \to \mathbb{N}^{|\Phi_i|}_1$, que forma um novo vetor e^i_k a partir de e_k, selecionando apenas os elementos com índice pertencente a Φ_i, em que Φ é o conjunto formado por todos os índices do vetor u_k, e $\Phi_i \subset \Phi$ é o conjunto de índices do vetor u_k associados com os sensores e atuadores observados pelo subsistema local A_i. Por exemplo, se $e_k = [e_{k,1}\ e_{k,2}\ e_{k,3}\ e_{k,4}]$ e $\Phi_1 = \{2, 3\}$, $e^1_k = [e_{k,2}\ e_{k,3}]$. É importante ressaltar que não necessariamente os conjuntos Φ_i, $i = 1, \ldots r$, são disjuntos, *i.e.* dois controladores locais podem compartilhar sinais.

Seja E o conjunto de vetores evento do modelo completo da planta, e E_i o conjunto de vetores evento observados por cada subsistema A_i. Caso exista um vetor evento $e_k \in E$ tal que $S_i(e_k) = \underline{0}$, em que $\underline{0}$ é o vetor nulo de tamanho $|\Phi_i|$, é dito que e_k é não observável.

***Definição* 2** (Vetor evento não observável)**.** *Seja E o conjunto de vetores evento da planta completa e S_i a operação de seleção de vetores associada à cada local A_i. O conjunto de vetores evento não observáveis, $E_{uo,i}$, para o subsistema A_i, é formado por todos os vetores evento $e_k \in E$, tal que $S_i(e_k) = \underline{0}$.* □

De acordo com a arquitetura descentralizada mostrada na Figura 6, depois da execução de um evento na planta, $e \in E$, cada subsistema o observa como $S_i(e) \in E_i$, para $i = 1, \ldots, r$, e, caso o evento seja observável, $S_i(e) \neq \underline{0}$, compara com os possíveis eventos disponíveis no modelo local livre de falhas. Caso a ocorrência do evento observável não seja possível, a falha é detectada. Portanto, para que seja possível implementar o diagnóstico de falhas descentralizado, é necessário, primeiramente, obter os modelos locais livre de falhas $\mathcal{G}_i$, $i = 1, \ldots, r$.

Visto que os modelos identificados são, neste artigo, baseados na observação de eventos da planta, o *Deterministic Automaton with Outputs and Conditional Transitions* (DAOCT) proposto em Moreira and Lesage (2019a), é utilizado para a identificação. O DAOCT é definido da seguinte forma:

***Definição* 3.** *(Deterministic Automaton with Outputs and Conditional Transitions) O Deterministic Automaton with Outputs and Conditional Transitions é definido como uma tupla de 9 elementos, $\mathcal{G} = (X, \Sigma, \Omega, f, \lambda, R, \theta, x_0, X_f)$, em que X é o conjunto de estados, Σ é o conjunto de vetores evento, $\Omega \subseteq \mathbb{Z}_2^n$ é o conjunto de vetores I/O, em que n é o número de sinais do sistema, $f : X \times \Sigma^* \to X$ é a função de transição determinística, $\lambda : X \to \Omega$ é a função de rótulo dos estados, $R = \{1, 2, \ldots, r\}$ é o conjunto de índices de caminhos, em que $r \in \mathbb{Z}^+$, $\theta : X \times \Sigma \to 2^R$ é a função de estimador de caminho, x_0 é o estado inicial e $X_f \subseteq X$ é o conjunto de estados finais .* □

Para computar o DAOCT, é necessário obter os caminhos de eventos $\tilde{p}_j$, associados com todos os caminhos $p_j = (u_{j,1}, u_{j,2}, \ldots, u_{j,l_j})$, de forma que $\tilde{p}_j = (u_{j,1}, e_{j,1}, e_{j,2}, \ldots e_{j,l_j-1})$, em que $e_{j,z} = u_{j,z} \oplus u_{j,z+1}$, para $z = 1, \ldots, l_j - 1$, *i.e.*, cada vetor evento de um caminho observado $\tilde{p}_j$ é computado a partir da operação XOR entre dois vetores I/O consecutivos do caminho p_j. Assim, o DAOCT é computado considerando todos os caminhos de eventos $\tilde{p}_j, j = 1, \ldots, |\mathcal{P}|$.

A função λ é responsável por associar cada estado $x \in X$ com um vetor I/O $u \in \Omega$. A função θ é usada para definir a linguagem gerada pelo modelo. Para isso, é necessário expandir a função θ para $\theta_s : X \times \Sigma^* \to 2^R$, definida recursivamente como:

$$\begin{aligned} \theta_s(u, \varepsilon) &= R, \\ \theta_s(u, s\sigma) &= \begin{cases} \theta_s(x, s) \cap \theta(x', \sigma), \text{em que} \\ x' = f(x, s), \text{se } f(x, s\sigma)! \\ \text{indefinido, caso contrário.} \end{cases} \end{aligned} \tag{2}$$

Finalmente, a linguagem gerada pelo DAOCT é dada por:

Tabela 1. Número de sensores e atuadores em cada módulo do exemplo prático

Módulo	Número de sensores	número de atuadores
Seleção	11	6
Braço mecânico	9	5
Prensa	6	6
Armazenamento	6	5

$$L_{\mathcal{G}} = \{s \in \Sigma^* : f(x_0, s)! \wedge \theta_s(x_0, s) \neq \emptyset\}. \tag{3}$$

De acordo com a Equação (3), uma sequência de vetores evento s só é representada no DAOCT se $f(x_0, s)$ for definido e existir pelo menos um caminho $\tilde{p}_j$ na estimativa de caminho após a ocorrência de s, representado pela condição $\theta_s(u, s) \neq \emptyset$.

Voltando para a arquitetura descentralizada da Figura 6, um DAOCT local $\mathcal{G}_i$ deve ser computado para cada módulo da planta. Logo, considere $p_j^i = (u_{j,1}^i, u_{j,2}^i, \ldots, u_{j,l_j^i}^i)$ um caminho observado pelo site local A_i e $\mathcal{P}^i$ o conjunto formado por todos os possíveis caminhos observados distintos do site local A_i. Cada subsistema A_i observa as entradas e saídas de um controlador local CLP_i, em que cada caminho parcialmente observado $p_j^i \in \mathcal{P}^i$ corresponde a uma tarefa local completa controlada por CLP_i. Assim, cada modelo $\mathcal{G}_i$ representa uma sub-tarefa executada pelo sistema.

Na próxima seção, o exemplo explicado na Seção 3 será utilizado para ilustrar a arquitetura descentralizada. Como o sistema é controlado por quatro CLPs, é necessário que sejam computados quatro modelos DAOCT, um para cada módulo.

5. DETECÇÃO DE FALHAS DESCENTRALIZADA NA PLANTA MECATRÔNICA

Retomando o exemplo da Seção 3, a Tabela 1 mostra a quantidade de sensores e atuadores do sistema, resultando em um total de 54 sinais da planta completa. Para a comunicação com o CLP e gravação dos sinais de sensores e atuadores foi utilizada a biblioteca Snap7 do Python, de forma a gerar planilhas onde cada linha representa um sinal de sensor ou atuador (I/O) e as colunas representam o valor em si, de forma que uma nova coluna é computada apenas quando for observada a variação de, pelo menos, um sinal do vetor I/O, como mostrado na Figura 7, para o módulo da prensa.

Assim, dada a base de dados com aquisição de vetores I/O, é necessário obter os caminhos observados $p_j \in \mathcal{P}$. Note que, de acordo com a Definição 3 cada modelo DAOCT, a ser identificado possui apenas um estado inicial x_0. Assim, primeiro define-se o vetor I/O inicial u_0, para cada subsistema, de forma que este represente unicamente seu respectivo estado inicial x_0. Por exemplo, o vetor I/O inicial do braço mecânico é considerado quando o braço se encontra retraído e abaixado, enquanto está na posição "zero" do *encoder*, que é quando o sensor indutivo está ligado. Esta posição é escolhida como estado inicial visto que o vetor I/O gerado por essa posição é único, ou seja, não se repete para nenhum outro estado do modelo. Assim, um caminho observado $p_j \in \mathcal{P}$ é obtido da planilha selecionando cada sequência de vetores I/O que comece

prensa levantada	1	1	1	1	1	1	1	0	1	1	1	1	1	1
prensa abaixada	0	0	0	0	0	0	1	1	1	0	0	0	0	0
porta aberta	1	1	1	1	0	0	0	0	0	0	0	1	1	1
porta fechada	0	0	0	0	0	1	1	1	1	1	0	0	0	0
prensa estendida	1	1	0	0	0	0	0	0	0	0	0	0	0	1
prensa recuada	0	0	0	1	1	1	1	1	1	1	1	1	0	0
abrir porta	0	0	0	0	0	0	0	0	0	1	1	0	0	0
fechar porta	0	0	0	1	1	0	0	0	0	0	0	0	0	0
estender prensa	0	0	0	0	0	0	0	0	0	0	0	1	1	0
recuar prensa	0	1	1	0	0	0	0	0	0	0	0	0	0	0
abaixar prensa	0	0	0	0	0	1	0	0	0	0	0	0	0	0
levantar prensa	0	0	0	0	0	0	1	1	1	0	0	0	0	0

Figura 7. Planilha de aquisição de vetores I/O de sinais do CLP, correspondente ao módulo da prensa.

com o vetor I/O inicial u_0 e termine com a emissão sequente do mesmo.

Dessa forma, os dados da planilha podem ser finalmente usados para a criação dos modelos parciais. Associado a cada controlador local, foram criados os modelos locais livres de falhas. Para o módulo da esteira, 4 caminhos diferentes foram identificados, no qual o DAOCT obtido possui 34 estados e 64 transições. Para o módulo do braço mecânico, foram obtidos 2 caminhos distintos, gerando um DAOCT com 26 estados e 28 transições. Já o módulo da prensa tem apenas 1 caminho, cujo DAOCT identificado possui 12 estados e 13 transições. Por fim, para o módulo do armazenador, foram obtidos 9 caminhos distintos, cujo DAOCT identificado possui 26 estados e 38 transições.

Note que os modelos obtidos são compactos, isto é, não possuem um número muito grande de estados e transições, sendo necessário por volta de 30 minutos de aquisição de dados do maior subsistema. Caso o sistema fosse controlado por um único CLP, seria possível computar um modelo monolítico para representar o sistema completo. Porém, devido aos comportamentos concorrentes dos quatro módulos apresentados, a quantidade de caminhos observáveis da planta poderia ter grande cardinalidade, o que necessitaria um tempo de identificação muito maior que o utilizado para identificar os modelos locais.

Com objetivo de exemplificar um dos modelos livres de falha computados, na Figura 8, é mostrado o DAOCT do módulo da prensa, cujos eventos são descritos na Tabela 2. Após a obtenção dos modelos locais livre de falhas de cada módulo do sistema, é necessário verificar a sua eficiência em detectar falhas usando a arquitetura descentralizada proposta na Figura 6. Para isso, foram criadas variáveis internas auxiliares nos CLPs, capazes de forçar a ativação (ON) e desativação (OFF) de qualquer sensor e atuador da planta, ou seja, forçam o valor dos sinais a serem iguais a 1 ou 0, respectivamente. Cada falha foi acionada individualmente, em momentos aleatórios de operação, de forma a ter 108 comportamentos de falha aplicados ao sistema.

Usando a arquitetura descentralizada, foram detectadas 59 das 108 falhas simuladas, na qual a relação de falhas detectadas por cada módulo do sistema é apresentada na Tabela 3. Todas as falhas detectadas são identificadas apenas por seus respectivos módulos. Isso se dá pelo fato de que cada subsistema trabalha isoladamente, não possuindo sinais em comum, com exceção do sensor de fim de curso, comum entre o braço robótico e a esteira. Porém, nesse

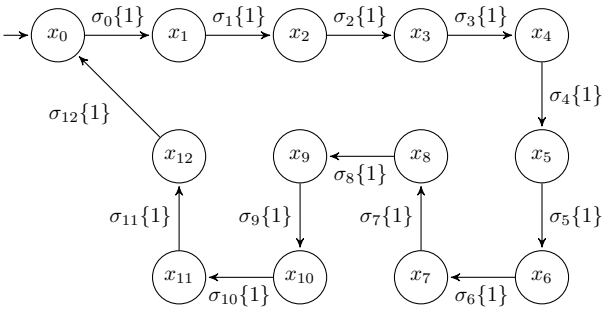

Figura 8. Modelo livre de falhas gerado para o módulo da prensa.

Tabela 2. Descrição dos eventos do modelo livre de falhas da prensa.

Evento	Descrição
σ_0	↑ recuarPrensa
σ_1	↓ prensaEstendida
σ_2	↑ prensaRecuada, ↑ fecharPorta , ↓ recuarPrensa
σ_3	↓ portaAberta
σ_4	↑ portaFechada , ↓ fecharPorta , ↑ abaixarPrensa
σ_5	↑ prensaAbaixada , ↓ abaixarPrensa , ↑ levantarPrensa
σ_6	↓ prensaLevantada
σ_7	↑prensaLevantada
σ_8	↓ prensaAbaixada , ↑ abrirPorta , ↓ levantarPrensa
σ_9	↓ portaFechada
σ_{10}	↑ portaAberta , ↓ abrirPorta , ↑ estenderPrensa
σ_{11}	↓ prensaRecuada
σ_{12}	↑ prensaEstendida

caso específico, o módulo do braço robótico não possui informação suficiente para saber se realmente tem uma caixa sendo transportada no módulo da esteira, fazendo com que as falhas nesse sinal sejam não detectáveis pelo modelo livre de falhas do braço robótico.

Tabela 3. Relação de falhas detectadas em cada módulo do sistema.

Módulo	Falhas detectadas	
	ON	OFF
Seleção	12	7
Braço mecânico	6	6
Prensa	7	8
Armazenamento	7	6

Comparando as falhas detectadas e não detectadas, nota-se que o momento em que a falha ocorre influencia diretamente em sua diagnosticabilidade. Um sinal que seja forçado a manter seu estado atual, como um sensor que estava baixo e é forçado a continuar em nível baixo, vai gerar um *deadlock* (bloqueio), visto que não é criado nenhum vetor I/O inesperado no meio do caminho. Já um sinal que é forçado a alterar seu estado atual é identificado como uma falha, porque será criado um novo estado, não existente no modelo livre de falhas. Para ilustração do problema, considere a simulação da falha do sensor de porta aberta da prensa. Segundo a Tabela 2, esse sensor apresenta uma borda de descida quando o evento σ_3 ocorre, e uma borda de subida quando o evento σ_{10} ocorre. Portanto, caso sua ativação fosse forçada nos estados $x_{11}, x_{12}, x_0, x_1, x_2$ e x_3 do modelo, a falha seria não diagnosticável. Porém, de acordo com a tomada de dados da simulação desse erro, ilustrada na Figura 9, a falha foi inserida quando

prensa levantada	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
prensa abaixada	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
porta aberta	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1
porta fechada	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
prensa estendida	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
prensa recuada	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0
abrir porta	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
fechar porta	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
estender prensa	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
recuar prensa	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
abaixar prensa	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
levantar prensa	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0

Figura 9. Planilha de aquisição de vetores I/O de sinais do CLP, simulando uma falha de ativação do sensor de porta aberta da prensa.

o subsistema se encontrava no estado x_6, tornando assim a falha diagnosticável.

Nesse contexto, é importante destacar que todas as 49 falhas não detectáveis geram uma linguagem não viva, ou seja, causam um *deadlock* logo após sua ocorrência no sistema, de forma que as 49 falhas são não diagnosticáveis. Ou seja, a eficiência do método proposto, na prática, é 100%. Dessa forma, o objetivo de adaptar a arquitetura descentralizada não comprometeu a detecção de falhas, visto que os modelos DAOCTs conseguiram detectar todas as falhas diagnosticáveis, sem a necessidade de um modelo monolítico da planta industrial.

6. CONCLUSÃO

Nesse artigo, foi apresentada a arquitetura descentralizada para detecção de falhas para sistemas controlados por mais de um CLP local. Para descrever o comportamento de cada subsistema da planta, foi escolhido o *Deterministic Automaton with Outputs and Conditional Transitions* (DAOCT), em que cada estado representa um vetor I/O do sistema e os eventos são a variação do sinal dos sensores e atuadores. A metodologia proposta foi aplicada a uma planta mecatrônica controlada por quatro controladores locais. A arquitetura se mostrou eficiente, sem a necessidade de construir um modelo monolítico para o diagnóstico.

Para trabalhos futuros, é de interesse utilizar um modelo que inclua a informação de tempo para detectar falhas que causem *deadlocks* (de Souza et al., 2020) além de desenvolver um coordenador que permita a comunicação dos modelos locais identificados, melhorando a eficiência do sistema de detecção de falhas.

AGRADECIMENTOS

Este trabalho foi apoiado pelo Programa de Recursos Humanos da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis – PRH-ANP, pelo CNPq sob os subsídios 436672/2018-9 e 431307/2018-0, FAPERJ sob os subsídios E-26/211.136/2019 e E-26/010/002620/2019, e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Código Financeiro 001.

REFERÊNCIAS

Cabasino, M.P., Daroundeau, P., Fanti, M.P., and Seatzu, C. (2015). Model identification and synthesis of discrete-event systems. *Contemporary Issues in System Science and Engineering*, 343–366.

de Souza, R., Moreira, M., and Lesage, J. (2020). Fault detection of discrete-event systems based on an identified timed model. *Control Engineering Practice*, 105, 104638.

Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, 10(1), 33–86.

Klein, S., Litz, L., and Lesage, J. (2005). Fault detection of discrete event systems using an identification approach. *IFAC Proceedings*, 38(1), 92–97.

Moreira, M., Jesus, T., and Basilio, J. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 56(7), 1679–1684.

Moreira, M. and Lesage, J. (2019a). Discrete event system identification with the aim of fault detection. *Discrete Event Dynamic Systems*, 29(2), 191–209.

Moreira, M. and Lesage, J. (2019b). Fault diagnosis based on identified discrete-event models. *Control Engineering Practice*, 91, 104101.

Roth, M., Lesage, J., and Litz, L. (2010). Black-box identification of discrete event systems with optimal partitioning of concurrent subsystems. *Proceedings of the 2010 American Control Conference*, 2601–2606.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.

Viana, G.S., Moreira, M.V., and Basilio, J.C. (2019). Codiagnosability analysis of discrete-event systems modeled by weighted automata. *IEEE Transactions on Automatic Control*, 64, 4361–4368.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 308–320.