Um modelo monolítico para detecção de falhas em Sistemas a Eventos Discretos obtido a partir da identificação de modelos parciais

Diego A. Libanio * **Gustavo S. Viana** * **Marcos V. Moreira** *

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

Abstract: In this paper, we propose a method for the computation of a composed model from the identified partial models for cyclic systems, where the composed model is used for fault detection. In order to do so, we propose a modification in one of the models presented in the literature that is suitable for composition, such that the modified model is capable of representing correctly cyclic systems. Then, we define a synchronous composition of the identified partial models, called modular synchronous composition. We also analyse the reduction of the exceeding language by implementing the composed model, and by varying a free parameter used to identify the partial models. A virtual plant, simulated using a 3D simulation software, and controlled by a programmable logic controller, is used to illustrate the proposed method.

Resumo: Neste artigo, é proposto um método para o cálculo de um modelo composto a partir de modelos parciais identificados para sistemas cíclicos, em que o modelo composto é usado para detecção de falhas. Para isso, uma modificação em um dos modelos apresentados na literatura, e adequado para a composição, é proposta, de modo que o modelo modificado seja capaz de representar corretamente sistemas cíclicos. Logo, é definida uma composição síncrona dos modelos parciais identificados, denominada composição modular síncrona. Por fim, é analisada a redução da linguagem em excesso, por meio da implementação do modelo composto e da variação de um parâmetro livre usado para identificar os modelos parciais. Uma planta virtual, simulada com um software de simulação 3D e controlada por um controlador lógico programável, é usada para ilustrar o método proposto.

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

Atualmente, com a evolução das fábricas inteligentes, tornou-se cada vez mais importante usar métodos eficazes de detecção e isolamento de falhas. O problema de diagnóstico de falhas para Sistemas a Eventos Discretos foi introduzido em Sampath et al. (1995). Desde então, diversos trabalhos desenvolveram diferentes técnicas para verificar a capacidade de um sistema em diagnosticar falhas, *i.e.*, a capacidade de detectar e isolar falhas em um número limitado de ocorrência de eventos (Debouk et al., 2000; Zaytoon and Lafortune, 2013; Cabral and Moreira, 2019; Viana and Basilio, 2019; Viana et al., 2019). Porém, em todos esses trabalhos, é considerado que o comportamento completo do sistema é conhecido, *i.e.,* o comportamento do sistema antes, e após a ocorrência de falhas, é conhecido, o que pode ser uma tarefa difícil, ou até impossível, para sistemas complexos. Assim, técnicas de identificação de sistemas, com objetivo de diagnóstico de falhas, foram propostas a partir do uso de autômatos e Redes de Petri (Klein et al., 2005; Roth et al., 2010; Dotoli et al., 2008; Cabasino et al., 2015; Estrada-Vargas et al., 2015; Moreira and Lesage, 2019b,a; Machado et al., 2023). Uma identificação usando Redes de Petri é, em geral, mais adequada quando se tem um conhecimento completo ou parcial do sistema. Já modelagens usando autômatos são mais adequadas para a identificação de sistemas caixa-preta, *i.e.*, sistemas cujas informações são limitadas às suas entradas e saídas (de Souza et al., 2020).

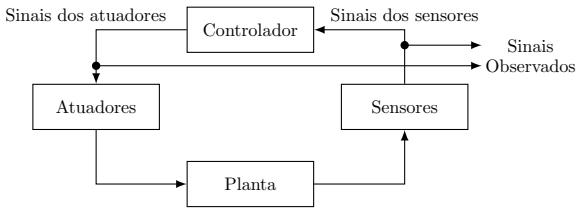

Figura 1. Sistema a eventos discretos malha fechada.

Em Klein et al. (2005), um algoritmo de identificação de sistemas caixa-preta é proposto usando uma classe de autômatos, denominada *Nondeterministic Autonomous Automata with Outputs* (NDAAO), além de introduzir um parâmetro k, utilizado para ajustar a eficácia do

modelo identificado. O NDAAO é obtido a partir da observação das sequências binárias das entradas e saídas do controlador, de acordo com a Figura 1, em que uma falha é detectada quando o sistema realiza algum comportamento que não pertença ao modelo identificado. A desvantagem da estratégia proposta em Klein et al. (2005), é que um modelo monolítico é identificado, o que pode exigir um longo tempo de observação dos sinais do controlador, de forma a obter todos possíveis comportamentos realizados pelo sistema. Esse problema ocorre quando o sistema é composto de diversos subsistemas com comportamentos concorrentes, o que leva a um grande número de diferentes comportamentos possíveis do sistema completo.

Para contornar o problema da identificação monolítica, Roth et al. (2010) apresenta uma metodologia de identificação distribuída, em que modelos parciais são obtidos a partir de uma observação parcial das entradas e saídas do controlador. Esses modelos parciais são identificados como NDAAOs e, em seguida, compostos para representar o comportamento completo do sistema, o qual é utilizado para detecção de falhas. A principal desvantagem do método proposto por Roth et al. (2010), é que, para sistemas cíclicos, os modelos parciais identificados podem não reinicializar corretamente, de forma que a linguagem original do sistema possa não pertencer à linguagem do modelo identificado *i.e.,* o modelo identificado não simula o comportamento do sistema sem falhas. Outro problema importante presente na estratégia proposta por Roth et al. (2010), é que a composição dos modelos parciais proposta, em geral, gera uma linguagem em excesso, *i.e.,* a linguagem aceita como livre de falhas pelo modelo identificado é maior que a linguagem original do sistema livre de falhas, de forma que os comportamentos de falhas, que pertençam a linguagem em excesso, não serão detectados.

Neste artigo é proposta uma modificação no modelo NDAAO apresentado em Roth et al. (2010), denominado NDAAO modificado (M-NDAAO), uma vez que, devido à simplicidade do NDAAO, este modelo é adequado para a composição de modelos parciais. Assim, o M-NDAAO tem como objetivo garantir a representação correta da linguagem original do sistema. Além disso, é proposta uma nova composição paralela entre modelos parciais. Esta composição, chamada de composição modular síncrona, permite a criação de um modelo monolítico a partir de M-NDAAOs parciais identificados, para qualquer valor do parâmetro livre k. O modelo monolítico computado deve representar todos os comportamentos livre de falhas que o sistema possa realizar, de forma que este será sincronizado com o sistema original para diagnosticar falhas, conforme ilustra a Figura 2. Portanto, baseado no modelo monolítico, é possível calcular a linguagem gerada, de forma a verificar sua eficiência em diagnosticar falhas. Também é analisada a redução da linguagem em excesso, a partir da variação do parâmetro livre k. Para demonstrar os resultados, um exemplo é desenvolvido a partir de uma planta virtual, modelada com o software de simulação 3D *Factory I/O* e controlada por um controlador lógico programável (CLP).

Este artigo está organizado da seguinte forma. Na Seção 2, são apresentados conceitos preliminares. Na Seção 3, o algoritmo de construção do M-NDAAO é apresentado, assim como suas propriedades. Na Seção 4, é apresentada a técnica de identificação distribuída, em que é definido os

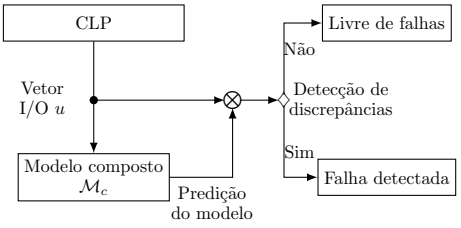

Figura 2. Arquitetura do esquema de detecção de falhas.

modelos parciais. Na Seção 5, a definição da composição modular síncrona, para diferentes valores do parâmetro livre k, é apresentada. Na Seção 6, um exemplo é utilizado para demonstrar os resultados do método proposto e, por fim, na Seção 7, conclusões são apresentadas.

2. CONCEITOS PRELIMINARES

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 fechamento de Kleene. A parte acessível de G é definida como $Ac(G) := (X_{ac}, \Sigma, f_{ac}, x_0, X_{ac,m})$, em que $X_{ac} = \{x \in X : (\exists s \in \Sigma^*)[x \in f_{nd}(x_0, s)]\}$, $X_{ac,m} = X_m \cap X_{ac}$ e $f_{ac} : X_{ac} \times \Sigma \to 2^{X_{ac}}$ é calculado a partir de f_{nd}, restringindo seu domínio aos estados acessíveis de G (Cassandras and Lafortune, 2008). A parte coacessível de G é definida como $CoAc(G) := (X_{coac}, \Sigma, f_{coac}, x_{0,coac}, X_m)$, em que $X_{coac} = \{x \in X : (\exists s \in \Sigma^*)(\exists x_m \in X_m)[x_m \in f_{nd}(x, s)]\}$, $f_{coac} : X_{coac} \times \Sigma \to 2^{X_{coac}}$ é obtido de f_{nd}, restringindo seu domínio aos estados coacessíveis de G, e $x_{0,coac} = x_0$, se $x_0 \in X_{coac}$, ou indefinido caso contrário (Cassandras and Lafortune, 2008). A operação $Trim$ de G é dada por $Trim(G) = Ac(CoAc(G)) = CoAc(Ac(G))$.

Um autômato é considerado autônomo quando as transições de um estado para o outro não dependem da ocorrência de um evento de entrada (Klein et al., 2005). Assim, um autômato autônomo não determinístico é definido como $G_a = (X, f_{nd}, x_0, X_m)$, em que $f_{nd} : X \to 2^X$ é uma função de transição não determinística em que qualquer estado em $f_{nd}(x)$ pode ser igualmente alcançado a partir de x após uma única transição.

Considere o sistema descrito na Figura 1, tal que n_I e n_O denotam o número de entradas e saídas binárias do CLP, respectivamente. Dessa forma, o vetor formado pelos sinais de entrada e saída do CLP em um instante de tempo $t_i \in \mathbb{R}^+$, chamado neste trabalho de vetor I/O, é denotado como $u_i = [I_1(i)\ I_2(i) \ldots I_{n_I}(i)\ O_1(i)\ O_2(i) \ldots O_{n_O}(i)]$, em que $I_\beta(i)$ e $O_\delta(i)$, 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_i. Consideramos que u_i é um vetor binário, *i.e.*, $u_i \in \mathbb{Z}_2^n$, em que $n = n_I + n_O$ e $\mathbb{Z}_2 = \{0, 1\}$. O comprimento de uma sequência de vetores I/O $s = u_1 u_2 \ldots u_l$ é denotado por $\|s\| = l$.

Considere $p_j = (\nu_{j,1}, \nu_{j,2}, \ldots, \nu_{j,l_j})$ um caminho observado em que $\nu_{j,z}$, para $z = 1 \ldots, l_j$, é um vetor I/O observado. Além disso, é possível definir $s_j = \nu_{j,1}\nu_{j,2} \ldots \nu_{j,l_j}$ como uma sequência de vetores I/O associados a p_j e $\mathcal{P} =$

$\{p_1, p_2, \ldots, p_{|\mathcal{P}|}\}$ sendo o conjunto de caminhos distintos observados. Neste artigo, é feita a seguinte hipótese.

***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 $p_j \in \mathcal{P}$, $\nu_{j,1} = \nu_{j,l_j}$.*

Após obter os caminhos cíclicos, é possível computar um modelo que represente o comportamento do sistema livre de falhas. Em Roth et al. (2010), um *Nondeterministic Autonomous Automaton with Outputs* (NDAAO) é proposto para modelar o comportamento do sistema livre de falhas, tal que $\mathcal{N} = (X, \Omega, f_{nd}, \lambda, x_0)$, em que X é o conjunto de estados, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ é o conjunto de sinais de saída, $f_{nd} : X \to 2^X$ é a função de transição autônoma não determinística, $\lambda : X \to \Omega$ é a função de saída e x_0 é o estado inicial.

O NDAAO tem um único estado inicial $x_0 \in X$, associado ao vetor I/O inicial observado em todos os caminhos p_j. A função de saída, λ, é responsável pela associação de cada estado $x \in X$ a um sinal de saída $\omega \in \Omega$, em que Ω é formado por todos os vetores I/O observados. A fim de satisfazer a relação de simulação, deve haver uma associação de cada $\omega \in \Omega$ com pelo menos um estado $x \in X$.

Assim, cada estado do NDAAO está associado a uma sequência distinta de vetores I/O de comprimento k, observados nos caminhos $p_j \in \mathcal{P}$. Para compreender a construção das sequências de vetores I/O de tamanho k, considere a função $Remove : \Omega^k \to \Omega^{k-1}$, que remove o primeiro vetor de uma sequência de vetores I/O de comprimento k, ou seja, se $\sigma^k = u_1 u_2 \ldots u_k$, então $Remove(\sigma^k) = u_2 \ldots u_k$. Para calcular o modelo NDAAO, primeiro é necessário gerar os caminhos modificados, p_j^k, a partir de p_j, de acordo com o parâmetro livre k, de forma que $p_j^k = (\sigma_{j,1}^k, \sigma_{j,2}^k, \ldots, \sigma_{j,l_j}^k)$, em que $\sigma_{j,1}^k$ é formado pela concatenação de k vetores I/O, todos iguais a $\nu_{j,1}$, e $\sigma_{j,z}^k = Remove(\sigma_{j,z-1}^k)\nu_{j,z}$, para $1 < z \leq l_j$. Assim, o modelo NDAAO pode ser calculado de modo que cada estado seja associado a uma sequência distinta de vetores I/O $\sigma_{j,z}^k$, e a transição entre dois estados existe no modelo NDAAO se, e somente se, as sequências de vetores I/O associadas aos estados, foram observadas em pelo menos um p_j^k. O valor de $\lambda(x)$, onde $x \in X$, é definido como o último vetor I/O da sequência de vetores I/O associados a x. De acordo com Klein et al. (2005), o parâmetro livre k permite uma troca entre o tamanho do modelo e sua precisão na representação dos caminhos observados.

Para definir a linguagem formada por todas as sequências de vetores I/O de comprimento um até um determinado valor $q \in \mathbb{N}$ do modelo NDAAO identificado, denotado como L_{Iden}^q, primeiro apresentamos as seguintes linguagens geradas a partir do estado inicial x_0:

$$W^q = \{s_\omega \in \Omega^q : s_\omega = \lambda(x_0)\lambda(x_1)...\lambda(x_{q-1}) \wedge (x_{\eta+1} \in f_{nd}(x_\eta), 0 \leq \eta < q-1)\}.$$

Logo, $L_{Iden}^q = \bigcup_{h=1}^q W^h$.

A linguagem formada por todas as sequências observadas de vetores I/O de comprimento q iniciando no primeiro vetor I/O, a partir dos caminhos observados p_j, $j = 1, \ldots, |\mathcal{P}|$, é dada por $W_{Obs}^q = \bigcup_{j=1}^{|\mathcal{P}|} \{\nu_{j,1}\nu_{j,2} \ldots \nu_{j,q}\}$. Assim, a linguagem formada por todas as sequências observadas de vetores I/O de comprimento um até um determinado valor q é $L_{Obs}^q = \bigcup_{h=1}^q W_{Obs}^h$.

Para definir a linguagem original do sistema, considere $p'_j = (\nu_{j,1}, \nu_{j,2}, \ldots, \nu_{j,l_j-1})$ o caminho obtido de p_j eliminando seu último vetor I/O, tal que $\mathcal{P}' = \{p'_j : j \in \{1, 2, \ldots, |\mathcal{P}|\}\}$. Como, de acordo com a Hipótese 1, todos os caminhos do sistema são cíclicos, após a execução completa de um caminho p'_j, qualquer outro caminho em $\mathcal{P}'$ pode ser executado. Assim, considere $\mathcal{P}'^\infty$ como sendo o conjunto formado pela concatenação de todos os caminhos de $\mathcal{P}'$ com todos os caminhos em $\mathcal{P}'$, um número arbitrariamente longo de vezes, de forma que os caminhos de $\mathcal{P}'^\infty$ têm comprimento ilimitado. Agora, considere um caminho $(\mu_{j,1}, \mu_{j,2}, \ldots) \in \mathcal{P}'^\infty$, em que $\mu_{j,z}$ é um vetor I/O, para $z = 1, \ldots, \infty$. Assim, considerando que todos os caminhos distintos possíveis p_j foram observados, a linguagem original formada por todas as sequências de vetores I/O de comprimento q é dada por $W_{Orig}^q = \bigcup_{j=1}^\infty \{\mu_{j,1}\mu_{j,2} \ldots \mu_{j,q}\}$. Note que W_{Orig}^q é finito. Portanto, a linguagem original formada por todas as sequências de vetores I/O, de comprimento um até q , é dada por $L_{Orig}^q = \bigcup_{i=1}^q W_{Orig}^q$. Por fim, caso $L_{Orig}^q \subset L_{Iden}^q$, significa que o modelo identificado contém sequências que não pertencem ao comportamento original da planta, caracterizando uma linguagem em excesso, $L_{Exc}^q = L_{Iden}^q \setminus L_{Orig}^q$.

3. O NDAAO MODIFICADO

Como discutido na Seção 1, o NDAAO proposto por Roth et al. (2010) pode não reinicializar corretamente um modelo que represente sistemas cíclicos. Isto ocorre quando a última sequência de vetores I/O σ_{j,l_j}^k, de um caminho modificado p_j^k, é igual à sequência de vetores I/O $\sigma_{i,z}^k$, em que i não é necessariamente diferente de j, e $1 < z < l_j$. Assim, o estado associado a essas sequências de vetores representa, simultaneamente, o término e a continuação de um caminho. Esta ambiguidade, impede o NDAAO de reinicializar, *i.e.*, retornar ao estado inicial, de forma que o modelo é incapaz de representar a finalização de tarefas, cujos caminhos modificados, apresentem esta dúvida. Como consequência, o modelo identificado por Roth et al. (2010) pode não representar corretamente a linguagem original para sistemas cíclicos. É importante ressaltar que essa ambiguidade, chamada de problema da reinicialização, só ocorre para parâmetros $k > 1$, visto que, para modelos identificados com $k = 1$, a reinicialização ocorre naturalmente, *i.e.*, o modelo constrói uma transição para o estado inicial, sempre que um caminho é finalizado.

Na sequência, um NDAAO modificado, que leva em conta a reinicialização do modelo após a execução de um caminho $p_j \in \mathcal{P}$, é apresentado. A estrutura do M-NDAAO é semelhante à estrutura do NDAAO sendo dada por $\mathcal{M} = (X, \Omega, f_\mathcal{M}, \lambda, x_0, X_m)$. A diferença se baseia em uma construção distinta do modelo a partir dos caminhos modificados p_j^k para $k > 1$, levando a uma função de transição diferente $f_\mathcal{M} : X \to 2^X$, e à definição de um conjunto de estados marcados X_m. No M-NDAAO, o único estado marcado é o estado inicial x_0, *i.e.*, $X_m = \{x_0\}$, para representar que o modelo foi reinicializado. É importante

observar que, para $k = 1$, o M-NDAAO é igual ao NDAAO, exceto pelo estado marcado no M-NDAAO.

No Algoritmo 1, o modelo M-NDAAO é construído a partir dos caminhos modificados p_j^k. Na linha 1, é criada a variável t, responsável por enumerar os estados $x_t \in X$, e j, que indica o índice do caminho. Nas linhas 2-4, é feita a inicialização do M-NDAAO, criando o estado inicial x_0 e sua saída $\lambda(x_0)$. Além disso, cria-se o conjunto O, que armazena todas as sequências de vetores I/O de comprimento k dos caminhos p_j^k, e a função bijetiva $\Lambda : X \to O$, que associa cada estado de X a uma sequência distinta $\sigma_{j,z}^k$. Na linha 5, o ciclo é iniciado e executado para todos os caminhos observados p_j^k, para $j = 1, 2, \ldots, |\mathcal{P}|$. Na linha 6, são criadas as variáveis x_c e z, em que x_c armazena o último estado visitado durante a construção do M-NDAAO e z é um contador. Nas linhas 8-20, o M-NDAAO é construído a partir de cada p_j^k, visitando cada sequência de vetores I/O $\sigma_{j,z}^k$. Se $\sigma_{j,z}^k$ ainda não foi visitado, um estado x_t é criado na linha 10, em que, na linha 11, sua saída é definida como o último vetor I/O de $\sigma_{j,z}^k$. Na linha 13, $\sigma_{j,z}^k$ é adicionada ao conjunto O. Na linha 14, uma transição de x_c para x_t é criada. Agora, caso $\sigma_{j,z}^k$ já tiver sido visitada, será feita uma pesquisa na linha 18 para encontrar o estado $x_p \in X$, que satisfaça $\Lambda(x_p) = \sigma_{j,z}^k$. Em seguida, na linha 19, x_p é adicionado à função de transição do último estado visitado x_c. De acordo com a Hipótese 1, o último elemento de um caminho p_j deve ser igual ao elemento inicial. Assim, na linha 22, a reinicialização do modelo é forçada, criando uma transição do último estado visitado x_c para o estado inicial x_0.

Algoritmo 1 Construção do M-NDAAO

Entradas: $p_1^k, p_2^k, ..., p_{|\mathcal{P}|}^k$
Saídas: $\mathcal{M} = (X, \Omega, f_\mathcal{M}, \lambda, x_0, X_m)$

1: $j \leftarrow 1, t \leftarrow 1$
2: Construa o estado inicial x_0, e defina $\lambda(x_0) = \nu_{j,1}$
3: $X \leftarrow \{x_0\}$, $X_m \leftarrow \{x_0\}$, $\Omega \leftarrow \{\nu_{j,1}\}$
4: $O \leftarrow \{\sigma_{j,1}^k\}$, $\Lambda(x_0) = \sigma_{j,1}^k$
5: **while** $j \leq |\mathcal{P}|$ **do**
6: $\quad x_c \leftarrow x_0$, $z \leftarrow 2$
7: $\quad f_\mathcal{M}(x_c) \leftarrow \emptyset$
8: $\quad$ **while** $z \leq l_j - 1$ **do**
9: $\quad\quad$ **if** $\sigma_{j,z}^k \notin O$ **then**
10: $\quad\quad\quad$ Crie o estado x_t, em que $f_\mathcal{M}(x_t) \leftarrow \emptyset$
11: $\quad\quad\quad$ Defina $\lambda(x_t) = \nu_{j,z}$ e $\Lambda(x_t) = \sigma_{j,z}^k$
12: $\quad\quad\quad$ $X \leftarrow X \cup \{x_t\}$, $\Omega \leftarrow \Omega \cup \{\nu_{j,z}\}$
13: $\quad\quad\quad$ $O \leftarrow O \cup \{\sigma_{j,z}^k\}$
14: $\quad\quad\quad$ $f_\mathcal{M}(x_c) \leftarrow f_\mathcal{M}(x_c) \cup \{x_t\}$
15: $\quad\quad\quad$ $x_c \leftarrow x_t$
16: $\quad\quad\quad$ $t \leftarrow t + 1$
17: $\quad\quad$ **else**
18: $\quad\quad\quad$ Encontre x_p tal que $\Lambda(x_p) = \sigma_{j,z}^k$
19: $\quad\quad\quad$ $f_\mathcal{M}(x_c) \leftarrow f_\mathcal{M}(x_c) \cup \{x_p\}$
20: $\quad\quad\quad$ $x_c \leftarrow x_p$
21: $\quad\quad$ $z \leftarrow z + 1$
22: $\quad f_\mathcal{M}(x_c) \leftarrow f_\mathcal{M}(x_c) \cup \{x_0\}$
23: $\quad j \leftarrow j + 1$

Assim, é possível definir a linguagem formada por todas as sequências de vetores I/O de comprimento um até

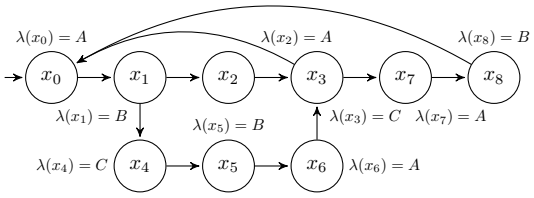

Figura 3. Modelo M-NDAAO do Exemplo 1.

q, geradas a partir do estado inicial x_0 do M-NDAAO, $L_{Iden,\mathcal{M}}^q$, de forma semelhante à linguagem identificada do NDAAO, L_{Iden}^q. Para isso, considere

$$W_\mathcal{M}^q = \{s_\omega \in \Omega^q : s_\omega = \lambda(x_0)\lambda(x_1)...\lambda(x_{q-1}) \wedge (x_{\eta+1} \in f_\mathcal{M}(x_\eta), 0 \leq \eta < q-1)\}.$$

Logo, $L_{Iden,\mathcal{M}}^q = \bigcup_{h=1}^q W_\mathcal{M}^h$.

Para mostrar que a linguagem gerada pelo M-NDAAO simula a linguagem do sistema original, considere a seguinte definição.

***Definição* 1.** *Considere que todos os caminhos distintos possíveis p_j gerados pelo sistema tenham sido observados e que $\mathcal{M}$ seja o M-NDAAO identificado, obtido a partir dos caminhos p_j, $j = 1, \ldots, |\mathcal{P}|$. Então, $\mathcal{M}$ é considerado um modelo completo no sentido de que a observação de qualquer outro caminho do sistema, no procedimento de identificação, não adiciona estados ou transições a $\mathcal{M}$.* □

***Teorema* 1.** *Considere que $\mathcal{M}$ seja um modelo completo. Logo, $L_{Orig}^q \subseteq L_{Iden,\mathcal{M}}^q$, para todo $q \in \mathbb{N}$.*

Prova. As provas deste teorema foram omitidas devido à limitação de páginas.

***Exemplo* 1.** *Considere um sistema cíclico que possa realizar dois caminhos: $p_1 = (A, B, C, B, A, C, A, B, A)$ e $p_2 = (A, B, A, C, A)$. De acordo com o Algoritmo 1, obtém-se o M-NDAAO para $k = 3$ representado na Figura 3. Considere agora que o caminho $p_1' p_2$ é gerado pelo sistema. Após a execução do caminho p_1, o estado x_0 é alcançado no modelo, permitindo uma nova execução do caminho p_1 ou a execução de um caminho diferente. Assim, o caminho p_2 pode ser executado no modelo, de forma que $L_{Orig}^q \subseteq L_{Iden,\mathcal{M}}^q$ para qualquer valor de $q \in \mathbb{N}$.*

Note que, em alguns casos, pode haver uma ambiguidade após a execução de um caminho completo. A ambiguidade ocorre quando o modelo pode estar em dois estados ao mesmo tempo. Neste exemplo, se p_2 for observado e executado pelo modelo, dois estados são possíveis no M-NDAAO, após p_2, sendo eles x_0 e x_7. Assim, após p_2, dois diferentes caminhos são possíveis no modelo M-NDAAO.

O M-NDAAO resultante do Algoritmo 1 é um modelo cíclico, que simula os caminhos observados e todas possíveis concatenações deles. Além disso, para valores de $k > 1$, é possivel ter uma ambiguidade quando o modelo é reinicializado, o que pode criar uma linguagem em excesso. No entanto, de acordo com a construção M-NDAAO, é possível resolver a ambiguidade, conforme dita o Teorema 2.

***Teorema* 2.** *Considere $\mathcal{M}$, um M-NDAAO obtido de acordo com o Algoritmo 1, para $k > 1$. Então, todas as ambiguidades são resolvidas em $k - 1$ passos, desde que nenhuma transição de reinicialização seja executada durante esses passos, i.e., o M-NDAAO não retorna ao estado inicial x_0 em nenhum dos $k - 1$ passos.*

Prova. As provas deste teorema foram omitidas devido à limitação de páginas.

O Teorema 2 estabelece que a linguagem em excesso gerada pela ambiguidade é eliminada após $k-1$ observações de vetores I/O, contanto que essas observações não levem a uma nova transição de reinicialização. É importante notar que, mesmo quando novas transições de reinicialização são executadas antes das $k-1$ observações, é possível, em alguns casos, eliminar a ambiguidade em um número limitado de observações.

4. IDENTIFICAÇÃO DISTRIBUÍDA

Neste artigo, um procedimento de identificação distribuída é realizado, usando o M-NDAAO para identificação dos modelos parciais. É importante ressaltar que o cálculo das entradas e saídas utilizadas para descrever o comportamento de cada modelo parcial está fora do escopo deste artigo, supondo que estes são obtidos por meio de qualquer técnica proposta na literatura, como os métodos apresentados em Roth et al. (2010) e Castro et al. (2022).

Considere que $\mathcal{M}_\ell = (X_\ell, \Omega_\ell, f_{\mathcal{M}_\ell}, \lambda_\ell, x_{0,\ell}, \{x_{0,\ell}\})$ para $\ell = 1, \ldots, r$, sejam os modelos parciais em que $\Phi = \{1, 2, \ldots, n\}$ é o conjunto formado pelos índices das entradas e saídas dos vetores I/O observados. Logo, $\Phi_\ell \subset \Phi$ é o conjunto dos índices das entradas e saídas associadas ao modelo parcial $\mathcal{M}_\ell$, de forma que $\mathcal{M}_\ell$ é calculado a partir dos caminhos observados p_j, considerando apenas as entradas e saídas do sistema que pertencem a Φ_ℓ. Para isso, as entradas e saídas de cada vetor I/O de p_j, que não pertencem a Φ_ℓ, devem ser substituídas pelo símbolo '$-$', que representa 'don't care', em cada vetor $\nu_{j,z} \in p_j$, resultando em um novo vetor $\nu^\ell_{j,z}$. É possível definir a operação de projeção $P_\ell : \mathbb{Z}_2^n \to \mathbb{Z}_{2,-}^n$, em que $\mathbb{Z}_{2,-} = \mathbb{Z}_2 \cup \{-\}$, tal que $P_\ell(u) = u^\ell$, onde $u^\ell[m] = u[m]$, se $m \in \Phi_\ell$, e $u^\ell[m] = -$, se $m \notin \Phi_\ell$, sendo que $u[m]$ denota o m-ésimo elemento de u. Então, $\nu^\ell_{j,z} = P_\ell(\nu_{j,z})$. A operação de projeção P_ℓ pode ser definida para uma sequência de vetores I/O, de maneira recursiva, $P_\ell(su) = P_\ell(s)P_\ell(u)$, em que $s \in \mathbb{Z}_2^{n^*}$, é uma sequência de vetores I/O, e $u \in \mathbb{Z}_2^n$, é um vetor I/O.

Após projetar cada vetor I/O de todos caminhos $p_j \in \mathcal{P}$, é possível que os novos caminhos tenham dois vetores I/O iguais consecutivos. Assim, para obter o caminho π_ℓ, que corresponde à observação parcial do caminho p_j pelo modelo parcial $\mathcal{M}_\ell$, é necessário combinar todos os vetores I/O iguais consecutivos em um único. Assim, o comprimento do caminho π_ℓ pode ser menor do que o comprimento de p_j. Além disso, dois caminhos diferentes em $\mathcal{P}$ podem levar ao mesmo caminho correspondente π_ℓ, o que implica que o número de caminhos distintos π_ℓ pode ser menor que $|\mathcal{P}|$. Assim, cada modelo parcial tem seu próprio conjunto de caminhos parciais, denotado como $\mathcal{P}_\ell$. Depois de obter o conjunto de caminhos parciais $\mathcal{P}_\ell$ para cada observação parcial, os modelos $\mathcal{M}_\ell$ podem ser computados usando o Algoritmo 1.

5. A COMPOSIÇÃO MODULAR SÍNCRONA

Nesta seção, é proposta uma composição síncrona para modelos parciais M-NDAAO identificados. A composição modular síncrona utiliza as transições de reinicialização para sincronizar o fim dos caminhos nos modelos parciais. Assim, quando um caminho completo p_j é executado no sistema, gerando uma sequência de vetores I/O s_j, suas observações parciais $P_\ell(s_j)$ são todas sincronizadas, levando aos estados iniciais de todos os modelos parciais. Isso reduz a linguagem em excesso que pode ser gerada pelos modelos parciais, quando executados em paralelo.

Antes de definir a composição modular síncrona, é necessário definir a operação união $J : \mathbb{Z}_{2,-}^n \times \mathbb{Z}_{2,-}^n \to (\mathbb{Z}_{2,-} \cup \{c\})^n$, em que c é um símbolo usado na operação para representar uma contradição (Roth et al., 2010).

Definição 2. *Considere que $u_i, u_j \in \mathbb{Z}_2^n$ são vetores I/O, em que $u_i^{\ell_1} = P_{\ell_1}(u_i)$ e $u_j^{\ell_2} = P_{\ell_2}(u_j)$, onde $\ell_1, \ell_2 \in \{1, \ldots, r\}$ e $\ell_1 \neq \ell_2$. A união de $u_i^{\ell_1}$ com $u_j^{\ell_2}$ é um vetor com n elementos, de modo que o m-ésimo elemento de $J(u_i^{\ell_1}, u_j^{\ell_2})$, denotado por $J(u_i^{\ell_1}, u_j^{\ell_2})[m]$, é definido como:*

$$J(u_i^{\ell_1}, u_j^{\ell_2})[m] = \begin{cases} u_i^{\ell_1}[m], & \text{se } u_i^{\ell_1}[m] = u_j^{\ell_2}[m] \\ u_i^{\ell_1}[m], & \text{se } u_i^{\ell_1}[m] \neq - \wedge u_j^{\ell_2}[m] = - \\ u_j^{\ell_2}[m], & \text{se } u_i^{\ell_1}[m] = - \wedge u_j^{\ell_2}[m] \neq - \\ c, & \text{se } u_i^{\ell_1}[m] \neq u_j^{\ell_2}[m] \neq - \end{cases}$$

para $m = 1, \ldots, n$. □

Note que, de acordo com a Definição 2, o símbolo c é usado para representar que o m-ésimo elemento dos vetores I/O não pode ser sincronizado, pois $u_i^{\ell_1}[m] \neq u_j^{\ell_2}[m]$, e ambos são diferentes de '$-$', o que significa que ambos os valores são observados por seus modelos parciais correspondentes. Observe também que quando o símbolo '$-$' aparece em $u_i^{\ell_1}[m]$ ou $u_j^{\ell_2}[m]$, e o outro é diferente de '$-$', esse valor é atribuído a $J(u_i^{\ell_1}, u_j^{\ell_2})[m]$.

Definida a operação união, é possível definir a composição modular síncrona, para dois casos: (i) $k = 1$; e (ii) $k > 1$. A diferença para os dois casos se refere ao fato de que, para $k = 1$, o modelo composto é reinicializado naturalmente, enquanto para $k > 1$ é necessário forçar a reinicialização de todos os modelos parciais quando uma tarefa executada pelo sistema é completada.

Definição 3 (Composição modular síncrona para $k = 1$)**.** *Considere que $\mathcal{M}_\ell = (X_\ell, \Omega_\ell, f_{\mathcal{M}_\ell}, \lambda_\ell, x_{0,\ell}, \{x_{0,\ell}\})$, para $\ell = 1, 2$ sejam dois modelos parciais M-NDAAO obtidos para $k = 1$. A composição síncrona modular de $\mathcal{M}_1$ e $\mathcal{M}_2$ é definida como:*

$$\begin{aligned} \mathcal{M}_c &= \mathcal{M}_1 || \mathcal{M}_2 \\ &= Trim(X, \Omega, f_{\mathcal{M}_c}, \lambda, (x_{0,1}, x_{0,2}), X_m), \end{aligned}$$

em que,

- $X = \{(x_1, x_2) \in X_1 \times X_2 : c \neq J(\lambda_1(x_1), \lambda_2(x_2))[m], \forall m \in \{1, 2, \ldots, n\}\}$;
- $\Omega = \{J(\lambda_1(x_1), \lambda_2(x_2)) : (x_1, x_2) \in X\}$;
- $f_{\mathcal{M}_c}(x_1, x_2) = \{(x_1', x_2') \in X : (x_1', x_2') \in [(\{x_1\} \cup f_{\mathcal{M}_1}(x_1)) \times (\{x_2\} \cup f_{\mathcal{M}_2}(x_2))] \wedge (x_1', x_2') \neq (x_1, x_2)\}$;
- $\lambda(x_1, x_2) = J(\lambda_1(x_1), \lambda_2(x_2)), \forall (x_1, x_2) \in X$;
- $X_m = \{(x_{0,1}, x_{0,2})\}$. □

De acordo com a Definição 3, é possível notar que o estado (x_1, x_2) pertence a X somente se os vetores I/O associados, $\lambda_1(x_1)$ e $\lambda_2(x_2)$, não gerarem uma contradição, indicada pelo símbolo c, em nenhum dos elementos do vetor de união

$J(\lambda_1(x_1), \lambda_2(x_2))$. Assim, o conjunto Ω é formado por todos os vetores de união possíveis obtidos dos elementos de X.

***Definição* 4** (Composição modular síncrona para $k > 1$)**.** *Considere que $\mathcal{M}_\ell = (X_\ell, \Omega_\ell, f_{\mathcal{M}_\ell}, \lambda_\ell, x_{0,\ell}, \{x_{0,\ell}\})$, para $\ell = 1, 2$ sejam dois modelos parciais M-NDAAO obtidos para $k > 1$. A composição síncrona modular de $\mathcal{M}_1$ e $\mathcal{M}_2$ é definida como:*

$$\begin{aligned}\mathcal{M}_c &= \mathcal{M}_1 || \mathcal{M}_2 \\ &= Trim(X, \Omega, f_{\mathcal{M}_c}, \lambda, (x_{0,1}, x_{0,2}), X_m),\end{aligned}$$

em que,

- $X = \{(x_1, x_2) \in X_1 \times X_2 : c \neq J(\lambda_1(x_1), \lambda_2(x_2))[m], \forall m \in \{1, 2, \ldots, n\}\}$;
- $\Omega = \{J(\lambda_1(x_1), \lambda_2(x_2)) : (x_1, x_2) \in X\}$;
- $f_{\mathcal{M}_c}(x_1, x_2) = \begin{cases} \{(x_1', x_2') \in X : (x_1', x_2') \in (\{x_1\} \cup \\ \quad f_{\mathcal{M}_1}(x_1)) \times (\{x_2\} \cup f_{\mathcal{M}_2}(x_2)) \wedge \\ \quad (x_1', x_2') \neq (x_1, x_2)\}, \\ \quad se\ (x_1 = x_{0,1}) \vee (x_2 = x_{0,2}) \\ \{(x_1', x_2') \in X : (x_1', x_2') \in [(\{x_1\} \cup \\ \quad f_{\mathcal{M}_1}(x_1)) \times (\{x_2\} \cup f_{\mathcal{M}_2}(x_2))] \setminus \\ \quad [\{x_{0,1}\} \times (f_{\mathcal{M}_2}(x_2) \setminus \{x_{0,2}\}) \cup \\ \quad (f_{\mathcal{M}_1}(x_1) \setminus \{x_{0,1}\}) \times \{x_{0,2}\}] \wedge \\ \quad (x_1', x_2') \neq (x_1, x_2)\}, \\ \quad caso\ contrário. \end{cases}$
- $\lambda(x_1, x_2) = J(\lambda_1(x_1), \lambda_2(x_2)),\ \forall (x_1, x_2) \in X$;
- $X_m = \{(x_{0,1}, x_{0,2})\}$. □

De acordo com as Definições 3 e 4, o conjunto $f_{\mathcal{M}_c}(x_1, x_2)$ é formado por todos os pares possíveis de estados que não levam a uma contradição, alcançados em ambos os modelos após uma transição. Porém, na Definição 4, caso uma das transições seja uma transição de reinicialização, e a outra não, a transição não é realizada na composição, de forma que os modelos só podem ser reinicializados ao mesmo tempo, indicando que o caminho associado do sistema foi concluído. Note que, a abordagem de sincronizar a reinicialização dos modelos parciais, identificados para $k > 1$, só é possível usando o modelo M-NDAAO, visto que, no M-NDAAO, uma transição de reinicialização é forçada após a observação completa de um caminho. Também é importante ressaltar que a estratégia de sincronizar as transições de reinicialização de todos os modelos parciais funciona somente quando, pelo menos, uma entrada ou saída de cada modelo parcial é alterada quando o caminho observado completo é reinicializado. Isso garante que a sincronização entre as transições de reinicialização represente corretamente o comportamento do sistema.

Como apenas o estado inicial de $\mathcal{M}_c$, nas Definições 3 e 4, é marcado, após a execução da operação *trim*, apenas os estados acessíveis e os estados coacessíveis são mantidos no modelo. A operação *trim* reduz a linguagem em excesso e evita o alcance de estados sem transições de saída que não pertencem ao comportamento do sistema sem falhas, pois, de acordo com a Hipótese 1, todas as tarefas executadas pelo sistema são cíclicas.

Considere que $L^q_{Iden,\mathcal{M}_c}$ denota a linguagem formada por todas as sequências de vetores I/O, de comprimento um até q, geradas por $\mathcal{M}_c$. Como o objetivo da composição é computar um modelo adequado para o detecção de falhas, a linguagem identificada $L^q_{Iden,\mathcal{M}_c}$ deve conter a linguagem original L^q_{Orig}, o que leva ao Teorema 3.

***Teorema* 3.** *Sejam $\mathcal{M}_\ell$, $\ell = 1, 2, \ldots, r$, os modelos parciais M-NDAAO completos e $\Phi = \bigcup_{\ell=1}^{r} \Phi_\ell$, de forma que $\mathcal{M}_c = \|_{\ell=1}^{r} \mathcal{M}_\ell$. Portanto, $L^q_{Orig} \subseteq L^q_{Iden,\mathcal{M}_c}$.*

Prova. As provas deste teorema foram omitidas devido à limitação de páginas.

De acordo com o Teorema 3, se os modelos parciais $\mathcal{M}_\ell, \ell = 1, 2, \ldots, r$, são completos, então a composição $\mathcal{M}_c = \|_{\ell=1}^{r} \mathcal{M}_\ell$ representa a linguagem original do sistema. Porém, para afirmar que um modelo é completo, é necessário uma observação infinita do sistema, algo inalcançável em um processo de identificação prático. Portanto, é necessário especificar um critério, que defina quando um modelo está suficientemente próximo de ser completo. Assim, o parâmetro η-convergência é definido.

***Definição* 5.** *Um modelo converge quando seu número de transições não aumenta após observar η caminhos executados pelo sistema, seguindo as etapas do Algoritmo 1, em que η é um parâmetro livre.*

Neste artigo, presume-se que o modelo está suficientemente próximo de ser completo quando o modelo converge, no qual o parâmetro η é definido como uma porcentagem do número total de caminhos observados.

Por fim, em Klein et al. (2005), é demonstrado que a linguagem em excesso de um modelo NDAAO monolítico é reduzida com o aumento do valor do parâmetro livre k. Assim, o Teorema 4 mostra a influência do parâmetro livre k na redução da linguagem em excesso do modelo composto $\mathcal{M}_c$, obtido da composição modular síncrona dos modelos M-NDAAO parciais.

***Teorema* 4.** *Sejam k e k' parâmetros livres, tal que $k > k'$, e considere $\mathcal{M}_\ell$ e $\mathcal{M}'_\ell$, para $\ell = 1, \ldots, r$, os modelos parciais identificados para k e k', respectivamente. Logo, $L^q_{Iden,\mathcal{M}_c} \subseteq L^q_{Iden,\mathcal{M}'_c}$, em que $\mathcal{M}_c = \|_{\ell}^{r} \mathcal{M}_\ell$ e $\mathcal{M}'_c = \|_{\ell}^{r} \mathcal{M}'_\ell$.*

Prova. As provas deste teorema foram omitidas devido à limitação de páginas.

De acordo com o Teorema 4, um aumento no valor do parâmetro livre k pode reduzir a linguagem gerada pelo modelo composto $\mathcal{M}_c$, o que também reduz sua linguagem em excesso. No entanto, é importante observar que o aumento do valor de k também leva ao aumento do número de estados e transições nos modelos parciais, de forma que a convergência dos modelos parciais pode precisar de mais observações para ser alcançada. Assim, há um *trade-off* entre o valor do parâmetro k e a convergência dos modelos parciais.

6. EXEMPLO PRÁTICO

Considere o sistema de separação de caixas representado na Figura 4, apresentado inicialmente em Moreira and Lesage (2019a). O sistema é composto por uma esteira de alimentação (FC), uma esteira de distribuição (DC), dois pistões, 1 e 2 ($P1$ e $P2$, respectivamente), e sensores $k_i, i = 1, \ldots, 8$. Assim, o sistema completo tem 4 atuadores e 8 sensores, sendo que o vetor I/O u dado por: $u = [k_7\ k_8\ k_5\ k_6\ k_2\ k_1\ k_4\ k_3\ P2\ P1\ FC\ DC]^T$.

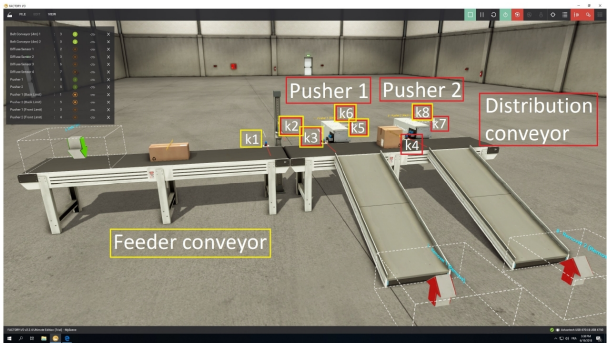

Figura 4. Sistema de separação de caixas.

O objetivo do sistema é empurrar as caixas altas na rampa da direita e as caixas pequenas na rampa à esquerda. Somente uma caixa pode estar na esteira de distribuição por vez. Assim, se houver uma caixa na esteira de distribuição e outra caixa chegar ao sensor k_1, a esteira de alimentação é desligada e só é ligada novamente depois de observar a ativação dos sensores k_6 ou k_8, indicando que os pistões $P1$ ou $P2$, respectivamente, foram retraídos e a caixa já foi empurrada. O sensor k_2 é usado para indicar se a caixa é alta, e os sensores k_3 e k_4 são usados para indicar que a caixa está na frente dos pistões $P1$ e $P2$, respectivamente. Após observar a desativação do sensor k_3 (ou k_4), a caixa está na posição para ser empurrada pelo pistão $P1$ (ou $P2$), e a esteira de distribuição para. A ativação dos sensores k_5 e k_7 indica que os pistões $P1$ e $P2$ estão completamente estendidos, respectivamente.

O sistema foi simulado usando o software de simulação 3D *Factory I/O* e controlado por meio de um CLP virtual. Para separar o sistema em modelos parciais, são usados os algoritmos apresentados em Castro et al. (2022). De acordo com o método proposto, foram obtidos dois modelos parciais, em que $\Phi_1 = \{2, 3, 4, 5, 6, 8, 11\}$ e $\Phi_2 = \{1, 2, 3, 4, 5, 7, 8, 9, 10, 12\}$. Assim, para identificar os modelos do sistema, foram observados continuamente 2577 vetores I/O gerados pelo sistema, que correspondem a 197 caminhos cíclicos, sendo 13 caminhos distintos. Com esses dados, é possível identificar, por meio do Algoritmo 1, dois modelos parciais $\mathcal{M}_1$ e $\mathcal{M}_2$, além do modelo monolítico $\mathcal{M}$, para valores de $k = 1, 2, 3$, e 4.

Nas Figuras 5 e 6, são mostrados os números de transições de $\mathcal{M}_1$ e $\mathcal{M}_2$, respectivamente, em relação ao número de caminhos observados. É possível observar que, para $k = 1$, o número de transições de $\mathcal{M}_1$ e $\mathcal{M}_2$ atinge um valor máximo após a observação de 76 caminhos cíclicos e 2 caminhos cíclicos, respectivamente. Para $k > 1$, o número de transições de $\mathcal{M}_1$ ainda atinge o máximo após 76 caminhos cíclicos, mas o número de transições de $\mathcal{M}_2$ atinge o máximo após 73 caminhos cíclicos. Isso mostra que, ao aumentar o valor de k, pode ser necessário observar mais caminhos para atingir o número máximo de transições dos modelos parciais. Na Figura 7, é apresentado o número de transições do modelo monolítico $\mathcal{M}$ versus o número de caminhos observados. Observe que o número de transições do modelo monolítico atinge seu máximo após 146 caminhos cíclicos, o que mostra, como esperado, que a identificação monolítica precisa de muito mais observações do que a identificação distribuída para convergir. Nesse caso, ao escolher um η para a convergência, igual à metade

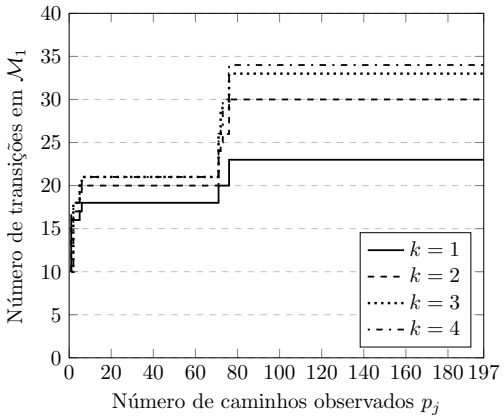

Figura 5. Convergência do modelo parcial $\mathcal{M}_1$.

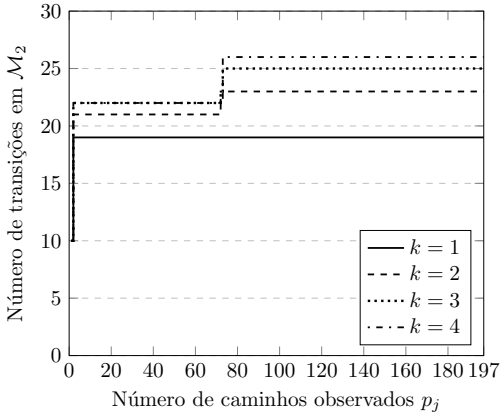

Figura 6. Convergência do modelo parcial $\mathcal{M}_2$.

do total de caminhos observados, ou seja, 99 caminhos cíclicos, conclui-se que os modelos parciais convergem após 175 caminhos cíclicos e o modelo monolítico não converge após os 197 caminhos cíclicos observados.

Uma vez que os modelos parciais convergiram, é possível computar o modelo composto $\mathcal{M}_c$ para os diferentes valores de k simulados. Na Tabela 1, o número de sequências da linguagem identificada de $\mathcal{M}_c$, $L^q_{Iden,\mathcal{M}_c}$, é apresentada para diferentes valores de q. Note que, para alguns valores de k ocorre uma redução na cardinalidade da linguagem identificada. De acordo com o Teorema 3 e com a Definição 5, a linguagem original sempre está contida na linguagem identificada da composição. Assim, a redução da linguagem identificada representa a eliminação de sequências que representam uma linguagem em excesso. Para $k = 2$ em comparação com $k = 1$, é possível observar uma grande redução da linguagem em excesso, para $k = 3$ em comparação com $k = 2$, não há redução, e para $k = 4$ em comparação com $k = 3$, há uma pequena redução.

Por fim, é verificada a eficiência dos modelos identificados para a detecção de falhas. Para isso, foram simuladas 45 falhas intermitentes e permanentes em todos os sensores e atuadores do sistema. As falhas simuladas estão forçando apenas um dos sinais dos sensores ou atuadores a ser igual a um ou zero. A Tabela 2 apresenta o número de falhas detectadas pelo modelo composto $\mathcal{M}_c$ para valores de $k = 1, 2, 3$, e 4. Observe que para $k = 2$ ou mais, o número de falhas detectadas é igual a 37, o que corresponde

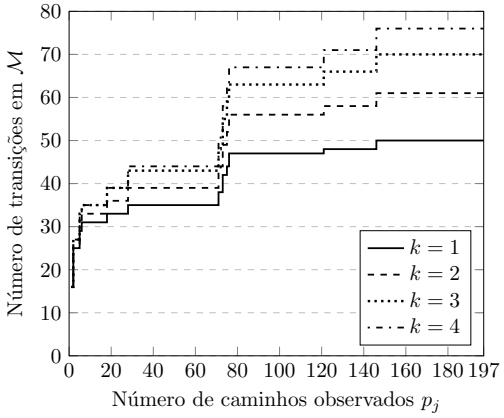

Figura 7. Convergência do modelo monolítico $\mathcal{M}$.

Tabela 1. Linguagem identificada do modelo $\mathcal{M}_c$

$L^q_{Iden,\mathcal{M}_c}$	$k=1$	$k=2$	$k=3$	$k=4$
$q=1$	12	5	5	5
$q=2$	50	12	12	12
$q=3$	209	21	21	21
$q=4$	822	35	35	30
$q=5$	3.218	61	61	39
$q=6$	12.501	112	112	52
$q=7$	48.521	189	189	77
$q=8$	188.244	288	288	122
$q=9$	730.446	414	414	188
$q=10$	2.834.416	592	592	296

Tabela 2. Detecção de falhas para diferentes valores de k.

	$k=1$	$k=2$	$k=3$	$k=4$
Falhas detectadas	27	37	37	37

a aproximadamente 82% do total de falhas simuladas. Portanto, nesse exemplo, o método proposto tem alta eficiência usando o modelo composto para $k = 2$.

7. CONCLUSÃO

Neste artigo, é apresentado um método de identificação distribuída para fins de detecção de falhas. O modelo de identificação proposto é um autômato autônomo não determinístico com saídas modificado (M-NDAAO), que é adequado para modelar sistemas cíclicos. Além disso, é apresentada uma composição modular síncrona que permite obter o modelo do sistema monolítico, usado no esquema de detecção de falhas. Para demonstrar a eficiência da metodologia proposta, esta foi aplicada a uma planta virtual controlada por um CLP virtual, em que foi demonstrado que o esquema proposto leva a uma detecção de falhas de alta eficiência usando modelos parciais que convergem mais rapidamente do que o modelo monolítico.

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.

Cabral, F.G. and Moreira, M.V. (2019). Synchronous diagnosis of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 17(2), 921–932.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, New York, 2nd edition.

Castro, J.G.V., Viana, G.S., and Moreira, M.V. (2022). Distributed identification of discrete-event systems with the aim of fault detection. *IFAC-PapersOnLine*.

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.

Dotoli, M., Fanti, M.P., and Mangini, A.M. (2008). Real time identification of discrete event systems using petri nets. *Automatica*, 44(5), 1209–1219.

Estrada-Vargas, A., López-Mellado, E., and Lesage, J. (2015). A black-box identification method for automated discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 14(3), 1321–1336.

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

Machado, T.H.M., Viana, G.S., and Moreira, M.V. (2023). Event-based automaton model for identification of discrete-event systems for fault detection. *Control Engineering Practice*, 134, 105474.

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. and Basilio, J.C. (2019). Codiagnosability of discrete event systems revisited: A new necessary and sufficient condition and its applications. *Automatica*, 101, 354–364.

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.