BUSCA GENERALIZADA DA PROPRIEDADE DE OBSERVADOR EM SISTEMAS A EVENTOS DISCRETOS

Hugo J. Bravo* Antonio E.C. Cunha† Patrícia N. Pena‡ José E.R. Cury§

*Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil

†SE/3 - Instituto Militar de Engenharia
Rio de Janeiro, RJ, Brasil

‡LACSED - DELT - Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brasil

§DAS - CTC - Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil

Email: hugobravoc@ufmg.br, carrilho@ime.eb.br, ppena@ufmg.br, cury@das.ufsc.br

Abstract— His paper adresses the search of the observer property in abstractions of discrete event systems. In this context, we generalize the OP-Search approach to deal with systems that have cycles of nonrelevant events.

Keywords— Discrete event systems, Natural projection, Observer property.

Resumo— Este artigo aborda o problema da busca da propriedade do observador em abstrações de sistemas a eventos discretos. Neste contexto, generaliza-se a abordagem PO-busca a fim de lidar com sistemas que apresentam ciclos de eventos não relevantes.

Palavras-chave— Sistemas a eventos discretos, Projeção natural, Propriedade de observador.

1 Introdução

O emprego de abstrações de Sistemas a Eventos Discretos (SED), que verifiquem a Propriedade de Observador (PO), tem-se mostrado eficaz no tratamentos de diversos problemas relacionados à Teoria de Controle Supervisório (TCS) (Ramadge e Wonham, 1989), como o controle hierárquico (Wong e Wonham, 1996), a verificação de não conflito do controle modular local (Pena et al., 2010a), entre outras. Abstrações, obtidas por *projeção natural*, que verificam a PO são chamadas de PO-abstrações (Pena et al., 2008).

No âmbito do acima citado, surge a seguinte pergunta: caso a abstração não verifique a PO, como refinar o sistema, de modo que a abstração da mesma a verifique? Assim, o *problema da busca da PO* consiste em refinar o sistema cuja abstração não verifica a PO, a fim de que a abstração do sistema refinado a verifique. Várias abordagens que lidam com este problema, como Wong e Wonham (2004), Schmidt e Moor (2006) e Feng e Wonham (2010).

Dentre essas abordagens, destaca-se por sua eficiência a proposta por Pena et al. (2010), em que propõe um algoritmo, denominado PO-busca. Este algoritmo apoia-se em outro, chamado PO-verificador (Pena et al., 2008), que determina se uma abstração é ou não PO-abstração. Caso não o seja, para um primeiro conjunto de eventos relevantes, denotado por $\Sigma_r \subseteq \Sigma$, então o PO-busca refinará o sistema por meio da renomeação de elementos do conjunto $\Sigma - \Sigma_r$, e a incorporação de novos eventos em Σ_r, até que a propriedade seja atendida. Contudo, o PO-verificador restringe-se a sistemas que não possuam ciclos de eventos não relevantes. Consequentemente, a PO-busca herdará, também, essa restrição.

Recentemente, a fim de contornar a restrição sobre o sistema original, Bravo et al. (2012) apresentaram uma versão modificada do PO-verificador, que permite tratar sistemas com ciclos de eventos não relevantes. Em vista disso, a obtenção de uma versão modificada da PO-busca, torna-se, também, possível.

Neste artigo, apresenta-se uma versão modificada da PO-busca, apoiada na versão modificada proposta por Bravo et al. (2012). O artigo segue com a Seção 2, em que apresentam-se alguns conceitos preliminares seguido da Seção 3, em que descreve-se a versão modificada do PO-verificador. Além disso serão propostas algumas propriedades. Na Seção 4 apresenta-se a versão modificada da PO-busca, com a proposta de algoritmo. Os resultados são discutidos na Seção 5.

2 Preliminares

Nesta seção, revisam-se alguns conceitos da TCS. Sob este paradigma, o comportamento dos SED é descrito por sequências de *eventos* obtidas desde um conjunto finito de eventos Σ. Σ^* é o conjunto de todas as sequências de eventos em Σ, incluindo a sequência vazia ϵ. A *concatenação* das sequências $s, u \in \Sigma^*$ formam a sequência $t = su$, em que $t \in \Sigma^*$. A sequência s é *prefixo* de t e denota-se por $s \sqsubseteq t$. O subconjunto $L \subseteq \Sigma^*$ é denominado *linguagem*. O *prefixo-fechamento* de L é definido por $\overline{L} = \{s \in \Sigma^* | s \sqsubseteq t \text{ para } t \in L\}$.

Uma linguagem regular $L \subseteq \Sigma^*$ pode ser reconhecida por *autômato* (não determinístico) de estados finitos $G = (\Sigma, Q, \rightarrow, Q^\circ)$, em que Σ é um conjunto finito de eventos, Q é o conjunto dos estados, $\rightarrow \subseteq Q \times \Sigma \times Q$ é relação de transição entre os estados, e $Q^\circ \subseteq Q$ é o conjunto dos estados iniciais. G é determinístico, se $|Q^\circ| \leq 1$, e $x \xrightarrow{\sigma} y_1$ e $x \xrightarrow{\sigma} y_2$ implicam sempre $y_1 = y_2$.

A relação de transição pode ser estendida para reconhecer sequências sobre Σ^* facultando $x \xrightarrow{\varepsilon} x$ para todo $x \in Q$, e $x \xrightarrow{s\sigma} z$ se $x \xrightarrow{s} y$ e $y \xrightarrow{\sigma} z$ para algum $y \in Q$. Além disso, $x \xrightarrow{s}$ significa $x \xrightarrow{s} y$ para algum $y \in Q$ e $x \rightarrow y$ significa $x \xrightarrow{s} y$ para algum $s \in \Sigma^*$. Essa notação pode ser aplicada também em conjuntos de estados e autômatos: $X \xrightarrow{s} Y$ para $X, Y \subseteq Q$ significa $x \xrightarrow{s} y$ para algum $x \in X$ e $y \in Y$, e $G \xrightarrow{s}$ significa $Q^\circ \xrightarrow{s}$.

A *linguagem gerada* de G é definida como $\mathcal{L}(G) = \{s \in \Sigma^* \mid Q^\circ \xrightarrow{s}\}$. Os estados marcados de G são indicados por autolaços rotulados com o evento $w \in \Sigma$, assim, a *linguagem marcada* de G é definida como $\mathcal{L}_m(G) = \{s \in (\Sigma \setminus \{w\})^* \mid sw \in \mathcal{L}(G)\}$. Um estado $x \in Q$ é *acessível* se $G \rightarrow x$, e *coacessível* se $x \xrightarrow{tw}$ para algum $t \in \Sigma^*$. Um autômato G é *acessível* se todos seus estados forem acessíveis, e *não bloqueante* se $\mathcal{L}(G) = \overline{\mathcal{L}_m(G)}$.

Sejam $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato e $\sim \subseteq Q \times Q$ uma relação de equivalência sobre Q. O *autômato quociente* de G é

$$G/\sim \; = (\Sigma, Q/\sim, \rightarrow / \sim, \widetilde{Q}^\circ), \tag{1}$$

em que, $\rightarrow/\sim = \{([x], \sigma, [y]) \mid x' \xrightarrow{\sigma} y'$ para algum $x' \in [x]$ e $y' \in [y]\}$ e $\widetilde{Q}^\circ = \{[x^\circ] \mid x^\circ \in Q^\circ\}$. Aqui, $[x] = \{x \in Q \mid x \sim x'\}$ denota uma classe de equivalência de $x \in Q$ e $Q/\sim \; = \{[x] \mid x \in Q\}$ é o conjunto que agrupa as classes de equivalência.

Neste artigo, Σ é particionado como $\Sigma = \Sigma_r \cup \Sigma_{nr}$, em que Σ_r é o conjunto de eventos *relevantes* e Σ_{nr} é o conjunto de eventos *não relevantes*. Para $\Sigma_r \subseteq \Sigma$, a *projeção natural* $\theta : \Sigma^* \rightarrow \Sigma_r^*$, mapeia sequências de eventos em Σ^* em sequências em Σ_r^* apagando os eventos que não pertencem a Σ_r. Este conceito pode ser estendido para linguagens como: $\theta(L) = \{t \in \Sigma_r^* \mid t = \theta(s)$ para algum $s \in L\}$. Uma propriedade que caracteriza a projeção natural é apresentada em seguida.

Definição 2.1 *(Wong et al., 2000) Sejam uma linguagem $L \subseteq \Sigma^*$ e $\theta : \Sigma^* \rightarrow \Sigma_r^*$ a projeção natural de Σ^* para Σ_r^*. Diz-se que $\theta(L)$ possui a propriedade de observador quando para todo $s \in \overline{L}$ e todo $t \in \Sigma_r^*$, se $\theta(s)t \in \theta(L)$ então existe $t' \in \Sigma^*$ tal que $\theta(st') = \theta(s)t$ e $st' \in L$.*

Em palavras, se uma abstração verifica a PO, então as tarefas realizadas no sistema abstraído correspondam sempre a abstrações de tarefas do sistema original. A projeção natural pode ser aplicada sobre autômatos, assim um autômato projetado $\theta(G)$, denomina-se PO-abstração se $\theta(\mathcal{L}_m(G))$ verifica a PO (Pena et al., 2008).

3 Verificação da Propriedade de Observador

3.1 Autômato G_{nr}

Com o propósito de tratar sistemas que possuam ciclos de eventos não relevantes introduz-se o autômato G_{nr}. Sejam $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato e $\Sigma_{nr} \subseteq \Sigma$. Uma relação sobre o conjunto Q é definida da seguinte forma:

$$x \xrightarrow{nr} y \Longleftrightarrow x \xrightarrow{s} y \text{ para algum } s \in \Sigma_{nr}^*; \tag{2}$$

$$x \xleftrightarrow{nr} y \Longleftrightarrow x \xrightarrow{nr} y \text{ e } y \xrightarrow{nr} x. \tag{3}$$

Se $x \xleftrightarrow{nr} y$, então os estados x e y, denominam-se *fortemente $\Sigma_{nr}-$conectados* ($\Sigma_{nr}-$FC). Se G não possui dois estados $\Sigma_{nr}-$FC distintos, então diz-se que é *$\Sigma_{nr}-$acíclico*. O conjunto de estados $\Sigma_{nr}-$FC é chamado de *componente fortemente $\Sigma_{nr}-$conectado* ($\Sigma_{nr}-$CFC). Se cada $\Sigma_{nr}-$CFC é contraído em um único estado, então o autômato resultante é *$\Sigma_{nr}-$acíclico*. O autômato resultante da contração denomina-se *autômato componente fortemente $\Sigma_{nr}-$conectado* (autômato $\Sigma_{nr}-$CFC) de G. Formalmente, a $\Sigma_{nr}-$CFC de $x \in Q$ é

$$[x] = \{y \in Q \mid x \xleftrightarrow{nr} y\}, \tag{4}$$

e o autômato $\Sigma_{nr}-$CFC de G é o autômato quociente

$$G_{nr} = G/\xleftrightarrow{nr}. \tag{5}$$

Por construção, G_{nr} não possui ciclos de eventos não relevantes com excepção de autolaços, isto é, se $[x] \xleftrightarrow{nr} [y]$, então $[x] = [y]$. além disso, para $y \in Q$, $[y]$ é definida como *componente terminal* se, para todo $\sigma \in \Sigma_{nr}$ e todo $z \in Q$ tal que $[y] \xrightarrow{\sigma} [z]$, constata-se que $[y] = [z]$.

Exemplo 3.1 *O autômato G_{nr} construído a partir de G (representado na Figura 1) é mostrado na Figura 2. Os estado 1,2 e 3 formam a $\Sigma_{nr}-$CFC $[1] = \{1, 2, 3\}$ em G_{nr}. Além disso, a $\Sigma_{nr}-$CFC $[0] = \{0\}$ e $\Sigma_{nr}-$CFC $[4] = \{4\}$.*

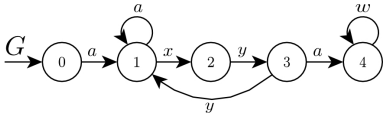

Figura 1: Autômato G.

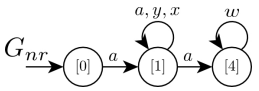

Figura 2: Autômato G_{nr}.

3.2 PO-verificador V_G

Nesta seção, apresenta-se um autômato não determinístico V_G, denominado *PO-verificador*, a fim de determinar se uma abstração é ou não PO-abstração. Sejam $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato determinístico e $G_{nr} = (\Sigma, Q_{nr}, \rightarrow_{nr}, \widetilde{Q}^\circ)$

um autômato quociente de G. O *PO-verificador* V_G para G é

$$V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ) \tag{6}$$

em que,

- $Q_V = \{P \subseteq Q/ \overset{nr}{\leftrightarrow} | \ 1 \leq |P| \leq 2\} \cup \{\perp\}$, ou seja, o conjunto de estado de Q_V que agrupa conjuntos de $\Sigma_{nr}-$CFC de cardinalidade 1 e 2, ou um estado especial $\perp$.
- $\rightarrow_V$ define-se da seguinte forma:

$$\{[x],[y]\} \overset{\sigma}{\rightarrow} \{[x'],[y']\} \text{ se } \sigma \in \Sigma_r, \ [x] \overset{\sigma}{\rightarrow}_{nr} [x'], \text{ e } [y] \overset{\sigma}{\rightarrow}_{nr} [y']; \tag{7}$$

$$\{[x],[y]\} \overset{\sigma}{\rightarrow} \{[x'],[y]\} \text{ se } \sigma \in \Sigma_{nr} \text{ e } [x] \overset{\sigma}{\rightarrow}_{nr} [x']; \tag{8}$$

$$\{[x],[y]\} \overset{\sigma}{\rightarrow} \perp \text{ se } \sigma \in \Sigma_r, \ [x] \overset{\sigma}{\rightarrow}_{nr}, \ [y] \text{ é terminal e } [x] \overset{\sigma}{\nrightarrow}_{nr} . \tag{9}$$

- $Q_V^\circ = \{\widetilde{Q}^\circ\}$ é o conjunto de estado inicial de V_G, o qual agrupa o estado inicial de G_{nr}.

Exemplo 3.2 *Com o propósito de ilustrar a obtenção de V_G, considere G_{nr}, como mostrado na Figura 2. O conjunto de eventos relevante de G_{nr} é $\Sigma_r = \{a, w\}$. O PO-verificador é o resultado das seguintes transições: por **(7)**, $\{[0]\} \overset{a}{\rightarrow} \{[1]\}$, $\{[1]\} \overset{a}{\rightarrow} \{[1]\}$, $\{[1]\} \overset{a}{\rightarrow} \{[4]\}$, $\{[1]\} \overset{a}{\rightarrow} \{[1],[4]\}$ e $\{[4]\} \overset{w}{\rightarrow} \{[4]\}$; por **(8)**, $\{[1]\} \overset{x}{\rightarrow} \{[1]\}$, $\{[1]\} \overset{y}{\rightarrow} \{[1]\}$, $\{[1],[4]\} \overset{x}{\rightarrow} \{[1],[4]\}$ e $\{[1],[4]\} \overset{y}{\rightarrow} \{[1],[4]\}$; e por **(9)**, $\{[1],[4]\} \overset{a}{\rightarrow} \perp$ e $\{[1],[4]\} \overset{w}{\rightarrow} \perp$. O verificador V_G é mostrado na Figura 3.*

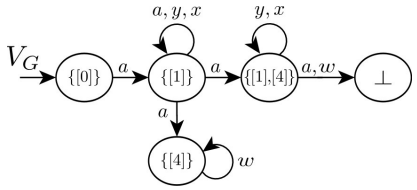

Figura 3: PO-verificador V_G.

O teorema a seguir demonstra a corretude do algoritmo PO-Verificador.

Teorema 3.1 *(Bravo et al., 2012) Seja $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato determinístico não bloqueante. O estado $\perp$ é acessível em V_G, se somente, se $\theta(G)$ não é PO-abstração.*

3.3 Propriedades

Propõem-se nesta subseção algumas propriedades concernentes aos estados de V_G. Nesse contexto, o conjunto Q_V é dividido em Q_V^{es} e Q_V^{ens}, em que Q_V^{es} é o conjunto de estados *seguros* e Q_V^{ens} é o conjunto de estados *não seguros*. Formalmente:

$$Q_V^{es} = \{\{[x],[y]\} \in Q_V | \{[x],[y]\} \nrightarrow \perp\}; \tag{10}$$

$$Q_V^{ens} = \{\{[x],[y]\} \in Q_V \mid \{[x],[y]\} \rightarrow \perp\}. \tag{11}$$

Proposição 3.1 *Sejam $V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ)$ o verificador para $G = (\Sigma, Q, \rightarrow, Q^\circ)$. Portanto, se $x \rightarrow y$, então $\{[x]\} \rightarrow \{[y]\}$, e se $G \rightarrow y$, então $V_G \rightarrow \{[y]\}$.*

Proposição 3.2 *Seja $V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ)$ o verificador. Para todo $\{[x],[y]\} \overset{\sigma}{\rightarrow} \perp$, então $\sigma \in \Sigma_r$ e $|\{[x],[y]\}| = 2$.*

4 Busca da Propriedade de Observador

4.1 Análise

Visto que $\perp$ é acessível no verificador V_G, obtido no Exemplo 3.2, então, pelo Teorema 3.1, determina-se que $\theta(G)$ não é PO-abstração. No contexto da PO-busca, um primeiro passo para tratar esse problema é identificar as causas que levam à violação da PO. Resultados anteriores mostram que os caminhos que ligam estados não seguros ao estado $\perp$ no verificador são compostos por estados de cardinalidade 2 e o primeiro evento no caminho é não relevante. Aplicando este resultado sobre V_G, observa-se que não existe nenhum caminho que conduz ao estado $\perp$ que cumpra com as condições estabelecidas. Não obstante, examinando os caminhos que ligam $\{[1]\}$ ao $\perp$ é possível deduzir que esses estão vinculados à violação da PO. Assim, surge a pergunta: será que a informação proporcionada por V_G é insuficiente para identificar as causas que levam à violação da PO?

A fim de contornar a restrição sobre G, na Seção 3 introduziu-se o autômato G_{nr}. Portanto, é evidente que o impacto de obter G_{nr} a partir de G se refletirá na construção de V_G, originando, assim, a falta de informação. Nesse âmbito, deduz-se que existe informação no verificador $\mathbb{V}$, proposto em Pena et al. (2008), que deveria ser incorporada no verificador V_G, com o intuito de obter uma estrutura que permita identificar as causas que originaram a violação da PO (Bravo, 2012).

4.2 Transformação

Nesta subseção, propõe-se um método que permite localizar a informação de $\mathbb{V}$ omitida por V_G. Nesse âmbito, apresenta-se o seguinte resultado.

Proposição 4.1 *Todo estado de $Q_V \setminus \{\perp\}$, encerra uma $\Sigma_{nr}-$CFC do verificador $\mathbb{V}$.*

Sejam $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato determinístico não bloqueante e $V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ)$ o verificador para G. A $\Sigma_{nr}-$CFC de $\mathbb{V}$, obtida a partir da *transformação* de $Q_V \setminus \{\perp\}$, é

$$(\Omega, \rightarrow^{\mathbb{V}}) \tag{12}$$

em que,

- Ω é o conjunto de estados, tal que cada elemento de Ω agrupa um ou dois elementos do conjunto Q, ou seja, $\Omega = \{\langle x, y\rangle \mid x, y \in Q\}$.

- $\rightarrow^{\mathbb{V}}$ define-se da seguinte forma:

$$\langle x,y\rangle \xrightarrow{\sigma} \langle x',y'\rangle \text{ se } \sigma \in \Sigma_r,\ x \xrightarrow{\sigma} x' \text{ tal que } x' \in [x] \text{ e } y \xrightarrow{\sigma} y' \text{ tal que } y' \in [y]; \quad (13)$$

$$\langle x,y\rangle \xrightarrow{\sigma} \langle x',y\rangle \text{ se } \sigma \in \Sigma_{nr},\ \text{e } x \xrightarrow{\sigma} x' \text{ tal que } x' \in [x]. \quad (14)$$

Algoritmo 4.1 (*Transformação*)

```
   Entrada: {[x],[y]}
 1 escolher x ∈ [x] e y ∈ [y], e construa ⟨x,y⟩;
 2 faça Ω ← ∅ e Λ ← ⟨x,y⟩;
 3 enquanto Λ − Ω ≠ ∅ faça
 4 |   selecione ⟨x,y⟩ ∈ Λ − Ω,
 5 |   excluir ⟨x,y⟩ de Λ e inserir em Ω
 6 |   expandir (⟨x,y⟩)
 7 fim
   Saída: (Ω, →^V)
 8 rotina expandir(⟨x,y⟩);
 9 para todo σ ∈ Σ_r faça
10 |   se x -σ-> e y -σ-> então
11 |   |   para todo x -σ-> x' e y -σ-> y' tal que
   |   |   x' ∈ [x] e y' ∈ [y] faça
12 |   |   |   ⟨x,y⟩ -σ-> ⟨x',y'⟩
13 |   |   |   Λ ← Λ ∪ ⟨x',y'⟩
14 |   |   fim
15 |   fim
16 fim
17 para todo σ ∈ Σ_nr faça
18 |   para todo x -σ-> x' tal que x' ∈ [x] faça
19 |   |   ⟨x,y⟩ -σ-> ⟨x',y⟩
20 |   |   Λ ← Λ ∪ ⟨x',y⟩
21 |   fim
22 |   para todo y -σ-> y' tal que y' ∈ [y] faça
23 |   |   ⟨x,y⟩ -σ-> ⟨x,y'⟩
24 |   |   Λ ← Λ ∪ ⟨x,y'⟩
25 |   fim
26 fim
```

A fim de auxiliar a identificação das causas que levam à violação da PO, a transformação será aplicada apenas sobre estados de $Q_V \setminus \{\bot\}$, que forem totais e não seguros (Bravo, 2012). Classifica-se como *estado total* a um estado de $Q_V \setminus \{\bot\}$ com cardinalidade 1, isto é, $\{[x]\}$, e $|[x]| > 1$.

O conjunto de estados totais e não seguros de V_G é $Q_V^{tns} = \{\{[x]_1\},\ldots,\{[x]_n\}\}$, tal que $Q_V^{tns} \subseteq Q_V$. Agora, se aplicarmos a transformação sobre cada elemento de Q_V^{tns}, obtemos o conjunto $T_{total} = \{(\Omega_1,\rightarrow_1^{\mathbb{V}}),\ldots,(\Omega_n,\rightarrow_n^{\mathbb{V}})\}$. A união da sequência de conjuntos Ω_n e a união da sequência de conjuntos $\rightarrow_n^{\mathbb{V}}$ são:

$$\Omega_{total} = \bigcup_{i=1}^{n} \Omega_i; \quad (15)$$

$$\rightarrow_{total}^{\mathbb{V}} = \bigcup_{i=1}^{n} \rightarrow_i^{\mathbb{V}}. \quad (16)$$

Visto que $(\Omega,\rightarrow^{\mathbb{V}})$ é uma $\Sigma_{nr}-$CFC de $\mathbb{V}$, então Ω_{total} e $\rightarrow_{total}^{\mathbb{V}}$ são famílias de conjuntos disjuntos.

4.3 Vizinhança

Dado um conjunto total e não seguro $\{[x]\} \in Q_V^{tns}$, um estado $\{[i],[j]\} \in Q_V$ é *vizinho antecessor* de $\{[x]\}$ se existe $\alpha \in \Sigma$, tal que $(\{[i],[j]\},\alpha,\{[x]\})$ e um estado $\{[t],[k]\} \in Q_V$ é *vizinho sucessor* de $\{[x]\}$ se existe $\beta \in \Sigma$, tal que $(\{[x]\},\beta,\{[t],[k]\})$. O conjunto de vizinhos antecessores e sucessores de $\{[x]\}$ é $\Gamma(\{[x]\})$. A vizinhança de Q_V^{tns} é

$$\Gamma(Q_V^{tns}) = \bigcup_{i=1}^{n} \Gamma(\{[x]_n\}). \quad (17)$$

Proposição 4.2 *Sejam $G = (\Sigma,Q,\rightarrow,Q^\circ)$ um autômato determinístico não bloqueante e o conjunto $\Gamma(\{[x]\})$, tal que $\{[x]\} \in Q_V^{tns}$, Então:*

(i) Para todo $\{[i],[j]\} \xrightarrow{\alpha} \{[x]\}$, tal que $\{[i],[j]\} \in \Gamma(\{[x]\})$ e $\alpha \in \Sigma_r$, então se $i \xrightarrow{\sigma} x$ tal que $i \in [i]$ e $x \in [x]$, e $j \xrightarrow{\sigma} y$ tal que $j \in [j]$ e $y \in [x]$, tem-se $\{[i],[j]\} \xrightarrow{\alpha} \langle x,y\rangle$;

(ii) Para todo $\{[i],[j]\} \xrightarrow{\alpha} \{[x]\}$, tal que $\{[i],[j]\} \in \Gamma(\{[x]\})$ e $\alpha \in \Sigma_{nr}$, então se $i \xrightarrow{\sigma} x$ tal que $i \in [i]$ e $x \in [x]$, tem-se $\{[i],[j]\} \xrightarrow{\alpha} \langle x,j\rangle$;

(iii) Para todo $\{[x]\} \xrightarrow{\beta} \{[i],[j]\}$, tal que $\{[i],[j]\} \in \Gamma(\{[x]\})$ e $\beta \in \Sigma_r$, então se $x \xrightarrow{\beta} i$ tal que $x \in [x]$ e $i \in [i]$, e $y \xrightarrow{\beta} j$ tal que $y \in [x]$ e $j \in [j]$, tem-se $\langle x,y\rangle \xrightarrow{\beta} \{[i],[j]\}$;

(vi) Para todo $\{[x]\} \xrightarrow{\beta} \{[i],[j]\}$, tal que $\{[i],[j]\} \in \Gamma(\{[x]\})$ e $\beta \in \Sigma_{nr}$, então se $x \xrightarrow{\beta} i$ tal que $x \in [x]$ e $i \in [i]$, tem-se $\langle x,j\rangle \xrightarrow{\beta} \{[i],[j]\}$.

Para $\Gamma(\{[x]\})$, transições obtidas por (i) e (ii), denominam-se *entradas* de $\{[x]\}$ e as obtidas por (iii) e (vi), denominam-se *saídas* de $\{[x]\}$. As entradas e saídas de $\{[x]\}$ é $\rightarrow_{\{[x]\}}^{in/out}$. O conjunto de estadas e saídas para Q_V^{tns} é

$$\rightarrow_{total}^{in/out} = \bigcup_{i=1}^{n} \rightarrow_{\{[x]_i\}}^{in/out}. \quad (18)$$

Para $\{[x]\} \in Q_V^{tns}$, a transição $\{[x]\} \overset{\sigma}{\leftrightarrow} \{[y]\}$ é definida como *terminal* se, para todo $\sigma \in \Sigma$ e todo $\{[y]\} \in Q_V$, constata-se que $\{[x]\} = \{[y]\}$. As transições terminais de $\{[x]\}$ é $\rightarrow_{\{[x]\}}^{ter}$. O conjunto de transições terminais para Q_V^{tns} é

$$\rightarrow_{V}^{ter} = \bigcup_{i=1}^{n} \rightarrow_{\{[x]_i\}}^{ter}. \quad (19)$$

4.4 Verificador Híbrido

Nesta subseção, apresenta-se um autômato não determinístico V_H, denominado *verificador híbrido*, como uma estrutura suplementar capaz de permitir a correta identificação das causas que originaram a violação da PO. Esta estrutura unifica

parte das informações proporcionadas pelos verificadores $\mathbb{V}$ e V_G. Seja $V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ)$ o verificador de G. O verificador híbrido V_H de V_G é

$$V_H = (\Sigma, Q_H, \rightarrow_H, Q_H^\circ) \tag{20}$$

em que,

- $Q_H = (Q_V \setminus Q_V^{tns}) \cup \Omega_{total}$
- $\rightarrow_H = (\rightarrow_V \setminus \rightarrow_V^{ter}) \cup \rightarrow_{total}^{\mathbb{V}} \cup \rightarrow_{total}^{in/out}$
- $Q_H^\circ = \{\widetilde{Q}^\circ\}$

A fim de padronizar os rótulos dos estados de $\mathbb{V}$ e V_G, sem perdida de generalidade, os estados $\langle x, y\rangle$, obtidos nas Subseções 4.2 e 4.3, serão apresentados como $\{[x], [y]\}$. A seguir apresentam-se os seguintes resultados.

Lema 4.1 *Se $Q_V^{tns} = \emptyset$, então $V_H = V_G$.*

Lema 4.2 *Sejam $G = (\Sigma, Q, \rightarrow, Q^\circ)$ um autômato determinístico e $V_H = (\Sigma, Q_H, \rightarrow_H, Q_H^\circ)$ o verificador híbrido, obtido a partir de V_G. Se $\perp$ é acessível em V_H, então existe um estado $a \in Q$, um evento $\eta \in \Sigma_{nr}$, e uma sequência de eventos $t \in \Sigma^*$ tal que $\{[a]\} \stackrel{\eta t}{\rightarrow} \perp$, e, para todo prefixo $t' < t$, $\{[a]\} \stackrel{\eta t'}{\rightarrow} \{[x], [y]\}$, tal que $|\{[x], [y]\}| = 2$.*

O Lema 4.2, aplicado sobre o verificador V_H, identifica de uma forma geral as causas que levam à violação da PO.

4.5 PO-busca

Motivados pela eficiência e simplicidade oferecida pela PO-busca, proposta em Pena et al. (2010), propõe-se, nesta subseção, uma versão modificada da PO-busca, a fim de obter PO-abstrações. Esta nova versão apóia-se na versão modificada do PO-verificador proposta por Bravo et al. (2012).

A diferença entre a versão modificada e versão original da PO-busca reside no fato de que, além de garantir os mesmo resultados e usufruir das vantagens computacionais, contornam-se as restrições sobre o sistema original. Assim, por exemplo, torna-se possível aplicar este método em sistemas que possuam ciclos de eventos não relevantes.

O algoritmo PO-busca (Algoritmo 4.2), segue a idéia de funcionamento do algoritmo PO-busca, proposto em Pena et al. (2010). No entanto, ante a existência de estados totais e não seguros em V_G, a fim de identificar as causas que levam à violação da PO, deve-se construir o verificador V_H.

O Teorema a seguir demonstra a corretude do algoritmo proposto.

Teorema 4.1 *Sejam $\widetilde{G}$ e $\widetilde{\Sigma}_r$ o resultado da execução do algoritmo PO-busca, e $\widetilde{\theta} : \widetilde{\Sigma}^* \rightarrow \widetilde{\Sigma}_r^*$ a projeção natural de $\widetilde{\Sigma}$ em $\widetilde{\Sigma}_r$. Então $\widetilde{\theta}(\widetilde{G})$ é sempre PO-abstração.*

Algoritmo 4.2 (*PO-busca*)

Entrada: $G = (\Sigma, Q, \rightarrow, Q^\circ)$
Entrada: Um primeiro conjunto $\Sigma_r \subseteq \Sigma$
1 Construir $V_G = (\Sigma, Q_V, \rightarrow_V, Q_V^\circ)$ a partir de G ;
2 **enquanto** $\perp$ *for acessível em* V_G **faça**
3 **se** $Q_V^{tns} \neq \emptyset$ **então**
4 Construir $V_H = (\Sigma, Q_H, \rightarrow_H, Q_H^\circ)$
5 **senão**
6 $V_H = V_G$
7 **fim**
8 Encontre um estado $a \in Q$, um evento $\eta \in \Sigma_{nr}$, e uma sequência $t \in \Sigma^*$ tal que $\{[a]\} \stackrel{\eta t}{\rightarrow} \perp$, e, para todo prefixo $t' < t$, existe $\{[a]\} \stackrel{\eta t'}{\rightarrow} \{[x], [y]\}$, tal que $|\{[x], [y]\}| = 2$;
9 Crie um novo evento $\eta' \in \Sigma_{nr}$ por renomeação de η;
10 Construa o autômato renomeação $\widehat{G}$ a partir de G por substituição dos rótulos com η da transição que parte de $a \in Q$ por η';
11 $\widehat{\Sigma}_r \leftarrow \Sigma_r \cup \{\eta'\}$;
12 Construir o verificador $\widehat{V}_{\widehat{G}}$ a partir de $\widehat{G}$ e $\widehat{\Sigma}_r$;
13 **faça** $V_G \leftarrow \widehat{V}_{\widehat{G}}$, $G \leftarrow \widehat{G}$ e $\Sigma_r \leftarrow \widehat{\Sigma}_r$
14 **fim**
Saída: $\widetilde{G} \leftarrow G$ e $\widetilde{\Sigma}_r \leftarrow \Sigma_r$

Exemplo 4.1 *Considere G, G_{nr} e V_G, mostrados nas figuras 1, 2 e 3, aqui retomados dos exemplos 3.1 e 3.2. O primeiro conjunto de eventos relevantes é $\Sigma_r = \{a, w\}$. Note que $\perp$ é acessível em V_G, então, pelo Teorema 3.1, $\theta(G)$ não é PO-abstração. Assim, G deve ser refinado conforme as linhas 4 a 11, do algoritmo PO-busca. Em V_G, identifica-se o estado $\{[1]\}$ como o único elemento de Q_V^{tns}. Portanto, pela Linha 4, constrói-se o verificador V_H, como mostrado na Figura 4. Após de obter V_H, seleciona-se um caminho que conduza a $\perp$ conforme a linha 8. O caminho selecionado é $(\{[3]\}, yaw, \perp)$. Pela linha 9, escolhe-se o evento $y1$ para substituir y, originando $\widehat{\Sigma}_r = \{a, w, y1\}$. Logo, pela linha 10, refina-se G substituído $(3, y, 1)$ por $(3, y1, 1)$. O autômato refinado $\widehat{G}$ (primeira renomeação) de G é mostrado na Figura 5.*

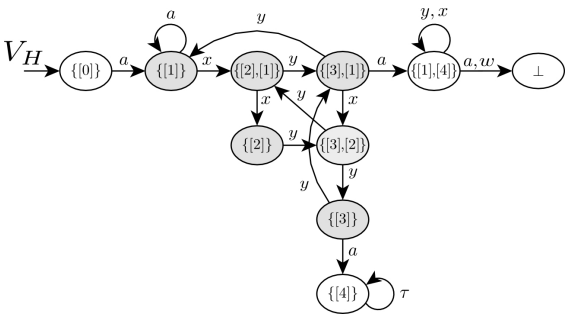

Figura 4: Verificador V_H.

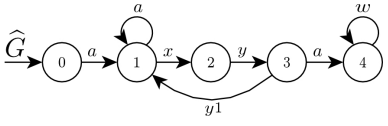

Figura 5: Autômato renomeado $\widehat{G}$.

De acordo com a linha 14, deve-se construir $\widehat{V}_{\widehat{G}}$ a partir de $\widehat{G}$ e $\widehat{\Sigma}_r$ a fim de verificar se $\perp$ ainda é acessível em $\widehat{V}_{\widehat{G}}$. Após da segunda renomeação de G, observa-se que $\perp$ já não é acessível em $\widehat{V}_{\widehat{H}}$, como mostrado na Figura 7. Portanto, a execução do algoritmo conclui e retorna o autômato $\widetilde{G}$, mostrado na Figura 6 e o conjunto de eventos relevantes é $\widetilde{\Sigma}_r = \{a, w, y1, x1\}$.

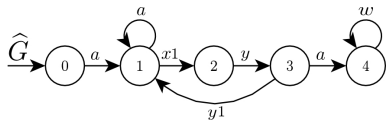

Figura 6: Autômato renomeado $\widehat{G}$.

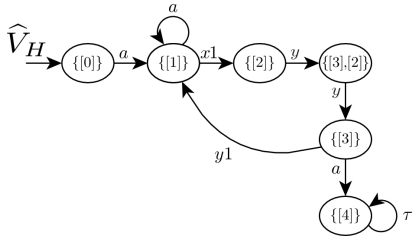

Figura 7: Verificador $\widehat{V}_{\widehat{H}}$.

5 Conclusões

Neste artigo foi proposta uma versão modificada da PO-busca, a qual torna possível o tratamento de sistemas que possuam ciclos de eventos não relevantes. Nesta versão, realiza-se a transformação alguns estados totais não seguros do verificador a fim de evidenciar as informações omitidas devido aos ciclos de eventos não relevantes, mas que são necessárias para se conseguir que um sistema abstraído apresente a propriedade de observador. Demonstra-se que a inclusão dessa informação pode ser realizada de forma parcial, isto é apenas nos estados envolvidos nos caminhos que levam ao não atendimento da propriedade de observador.

6 Agradecimentos

O presente trabalho foi realizado com o apoio financeiro da CAPES - Brasil. O terceiro autor é apoiado parcialmente pela FAPEMIG. O quarto autor é apoiado parcialmente pelo CNPq (PQ300953/93-3). Todos os autores são apoiados parcialmente pela CAPES (Procad 102/2007).

Referências

Bravo, H. (2012). *Verificação e Busca da Propriedade de Observador em Sistemas a Eventos Discretos*, Master's thesis, Instituto Militar de Engenharia, Rio de janeiro - Brasil.

Bravo, H., da Cunha, A., Pena, P., Malik, R. e Cury, J. (2012). Generalised Verification of the Observer Property in discrete event systems, *Proceedings of the 11th International Workshop of Discrete Event Systems, WODES'12*, Guadalajara, Mexico, pp. 337–342.

Feng, L. e Wonham, W. M. (2010). On the Computation of Natural Observers in Discrete-Event Systems, *Discrete Event Dynamic Systems* **20**(1): 63–102.

Pena, P., Cury, J., Cunha, A. e Lafortune, S. (2010a). Metodologia e Ferramenta de Apoio ao Teste de Não-Conflito no Controle Modular de Sistemas a Eventos Discretos, *Sba Controle & Automação* **21**(1): 58–68.

Pena, P., Cury, J. e Lafortune, S. (2008). Polynomial-Time Verification of the Observer Property in Abstractions, *Proceedings of the 2008 American Control Conference, ACC'08*, Seattle, USA, pp. 465–470.

Pena, P. N., Cury, J., Malik, R. e Lafortune, S. (2010). Efficient Computation of Observer Projections using OP-Verifiers, *Proceedings of the 10th International Workshop on Discrete Event Systems, WODES'10*, Berlin, Germany, pp. 416–421.

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

Schmidt, K. e Moor, T. (2006). Marked-String Accepting Observers for the Hierarchical and Decentralized Control of Discrete Event Systems, *8th International Workshop on Discrete Event Systems, 2006*, pp. 413–418.

Wong, K. C., Thistle, J. G., Malhamé, R. P. e Hoang, H. H. (2000). Supervisory Control of Distributed Systems: Conflict Resolution, *Discrete Event Dynamic Systems* **10**: 131–186.

Wong, K. e Wonham, W. (2004). On the Computation of Observers in Discrete-Event Systems, *Discrete Event Dynamical Systems* **14**(1): 55–107.

Wong, K. e Wonham, W. M. (1996). Hierarchical Control of Discrete-Event Systems, *Discrete Event Dynamic Systems: Theory and Applications* **6**(3): 241–273.