Modelagem e Controle de Ambientes Dinâmicos Dependentes de Contexto

André Lucas Silva Mainara Cristina Lorencena Richardson Ribeiro Marcelo Teixeira

***Resumo*—A *Teoria de Controle Supervisório* provê um mecanismo formal para o cálculo de controladores para *Sistemas a Eventos Discretos*, inspirado na Teoria dos *Autômatos Finitos* e *Linguagens*. Mesmo em face à vantagens, a aplicação dessa teoria em escala industrial esbarra em algumas limitações, como por exemplo a adaptabilidade de uma solução de controle para ambientes dinâmicos. Neste trabalho, tal limitação é ilustrada por meio de um exemplo prático de um processo industrial, que também é usado para a proposta de uma solução para o problema, baseada em modelagem por *Autômatos Finitos Estendidos*. Resultados apontam para a possibilidade de aplicar a teoria sobre ambientes que exigem que a ação de controle seja dinâmica, se reconfigurando em tempo de execução. No artigo são abordadas as etapas de modelagem do sistema e de síntese de um controlador que resolve o problema de controle de maneira minimamente restritiva e não bloqueante.**

***Palavras-chave*—Coordenação de Robôs, Leitura de Contexto, Especificação Dinâmica, Controle Supervisório.**

I. Introdução

A Teoria de Controle Supervisório (TCS) é um método que define formalmente a síntese de controladores para Sistemas a Eventos Discretos (SEDs). Estruturada fundamentalmente sobre a teoria dos de Autômatos Finitos (AFs) e Linguagens Regulares, a TCS provê um mecanismo formal para o cálculo de controladores que, por construção, garantem certas propriedades qualitativas, como a máxima permissividade e o não bloqueio, características importantes no meio industrial.

Na TCS, o sistema a ser controlado é modelado por um AF que é denominado planta. As ações de controle a serem exercidas sobre o sistema também são modelados por AFs e são denominadas especificações. A aplicação das especificações na planta é dada por uma entidade denominada *supervisor*, o qual é calculado a partir de uma operação matemática executada sobre o modelo que representa a composição de plantas e especificações. O resultado dessa operação é um AF, que sintetiza de maneira ótima as ações de controle desejadas [1].

Mesmo em face às vantagens da TCS, amplamente reconhecidas, a aplicação da TCS em escala industrial esbarra em algumas limitações, como a explosão do espaço de estados que ocorre no processamento de AFs extensos, problema clássico que é amplamente investigado em controle supervisório [2], [3], [4], [5], [6]. Além disso, a modelagem de especificações de controle pode ser complexa para ser expressa por AFs, particularmente quando envolve a dependência de dados.

Os autores pertencem ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná, Pato Branco, Brasil ({andsil,mainara}@alunos.utfpr.edu.br, {richardsonr,marceloteixeira}@utfpr.edu.br).

Uma variação da TCS que incorpora o uso de variáveis por meio de modelos de Autômatos Finitos Estendidos (AFEs) tem sido investigada na literatura como forma de tratar das limitações mencionadas [7], [8], [9]. Essa abordagem tem a finalidade de monitorar, por meio do conjunto de variáveis, os elementos do sistema que contribuem para a sua evolução. A atualização do conjunto de variáveis se dá por meio da inserção de fórmulas lógicas junto às transições que modelam a dinâmica do sistema. Da mesma forma, as ações de controle também são expressas levando-se em conta os valores das variáveis. Nesse caso, fórmulas de guarda sintetizam a semântica dos requisitos de controle por meio da checagem de condições lógicas antes da ocorrência de cada transição.

Grande parte dos estudos relacionado ao uso de AFEs se dedica a tratar do processo de simplificação da modelagem de sistemas. Poucos trabalhos têm se concentrado em investigar o impacto disso no processo cálculo do controlador. Ademais, não é de conhecimento nenhum trabalho que explore a estrutura de variáveis para fins de gerenciar o chaveamento de contexto do controlador em relação ao status da planta.

Dessa forma, o presente trabalho se concentra nessa lacuna, ou seja, ele explora uma estrutura de variáveis associada ao modelo de um SED para coordenar a dinâmica das ações de controle em tempo real. Em tese, se uma estrutura de variáveis estiver associada ao sistema de controle e o modelo do sistema, assim como os seus requisitos, reconheça essa estrutura, então seria possível permitir a ocorrência ou não de um determinado evento com base no valor atual da variável.

Um reflexo imediato dessa hipótese é que a dinamicidade do sistema passa a estar diretamente atrelada à atualização das variáveis. Do mesmo modo, é esperado que as ações de controle sejam personalizadas em tempo de execução, sem requerer uma completa reestruturação na modelagem e nem o cálculo de um novo controlador para cada contexto de controle. Será mostrado o processo de modelagem e cálculo de um controlador cuja ação sobre um SED é auto adaptável conforme um conjunto de variáveis. A abordagem é ilustrada por meio de um exemplo de controle de um SED cuja evolução integra diferentes contextos, reconhecidos e adaptados dinamicamente pelo controlador, em tempo de execução.

Este manuscrito é organizado da seguinte forma: a Seção II apresenta uma breve revisão do estado da arte; a Seção III apresenta os conceitos que fundamentam o estado da arte em que o trabalho está inserido, com o intuito de oferecer um suporte teórico para o entendimento dos resultados; a Seção IV descreve a abordagem convencional de síntese de supervisores para SEDs, a qual sustenta a extensão proposta na Seção V,

exemplificada por um estudo de caso, na Subseção V-F; por fim, a Seção VI apresenta algumas conclusões e perspectivas para a continuidade da pesquisa.

II. Revisão do estado da arte

Alguns trabalhos demonstram o emprego AFEs utilizando tanto a conceituação matemática quanto exemplos de aplicações, com diversas finalidades na ação de controle. Por exemplo, [10] propõem a implementação de um controle supervisório para um sistema modelado por máquinas de estados finitos utilizando variáveis booleanas e funções guardas, que são responsáveis pela ação de controle, permitindo ou não a ocorrência de um evento analisando o valor booleano das variáveis, dentro de um conjunto válido, no momento da ocorrência do evento. [11] demonstra uma aplicação prática da modelagem de um SED utilizando AFE aplicado na logística de uma rede de distribuição de energia elétrica.

Recentemente, muitos trabalhos focam na complementação da TCS aplicada a AFE, como [12] que faz o uso de abstração de variáveis visando diminuir o custo da síntese do controlador e aperfeiçoar a ação de controle. Já os benefícios da abordagem de AFE para a redução da complexidade de modelagem de problemas de controle discreto são explorados em [5] juntamente com a introdução de algoritmos de síntese de controladores que suportam técnicas de abstração de variáveis.

[13] mostram a abstração de projeção de eventos de um AFE, com resultados significativos na redução do tamanho do controlador, aumentando assim o desempenho computacional em termos de processamento e uso de memória. Em [14] é proposto um algoritmo para sintetizar o controlador em plantas e especificações modeladas por um AFE, fazendo o uso das funções de guardas associadas aos eventos, visando manter a segurança na ação de controle do supervisor modelado por um AFE. Em [15], é apresentado um modelo de verificação das propriedades da TCS para AFE de maneira eficiente em tempo polinomial. [2] tratam o problema da explosão do espaço de estados em AFE, entretanto utilizado diagramas binários de decisão [16] para representar a transição do supervisor modelado por um AFE.

Em relação à literatura relacionada ao trabalho, não é de conhecimento dos autores nenhuma obra que explore a estrutura de variáveis para fins de gerenciar o chaveamento de contexto do controlador em relação ao status da planta, meta esta que compõe o objetivo desse trabalho.

III. Conceitos introdutórios

A. Sistemas a Eventos Discretos

Os SEDs compõem uma classe de sistemas na qual que as transições entre estados são regidas pela ocorrência de estímulos denominados *eventos*, que podem ocorrer de maneira assíncrona e normalmente em intervalos irregulares de tempo [1]. Na ocorrência de um evento, o sistema muda instantaneamente para outro estado, onde permanece até a ocorrência de um próximo evento, que possa ocasionar outra transição. Assim, o sistema não depende da passagem de tempo para mudar de estado.

B. Modelagem de SEDs

A etapa de modelagem de um SED se justifica, principalmente, pelo fato de que a realização de análises e experimentos sobre a estrutura real do sistema pode não ser trivial ou nem mesmo possível. É comum que um sistema seja extenso, complexo, ou mesmo insalubre para a ação humana. Ainda, em geral algumas partes do comportamento de um sistema podem ser menos relevantes para determinados contextos, podendo ser abstraídas para fins de análise [17]. Essa representação de um SED, em níveis de abstração, pode ser definida como um *modelo do sistema*, e a ação de representar o sistema por um modelo é denominada de *modelagem*.

Dentre os formalismos existentes para a modelagem de SEDs citam-se as *Redes de Petri*[18], a *Teoria das Filas* [19] e a *Teoria dos Autômatos e Linguagens* [20]. Cada formalismo, no entanto, possui suas próprias características, vantagens e desvantagens, de modo que nenhum é até hoje aceito de maneira unânime ou como um padrão para a modelagem de SEDs.

Este trabalho adota a modelagem por *Autômatos e Linguagens*, que representa o sistema por uma estrutura de transição de estados que facilita especificar quais eventos podem ocorrer ou não em determinado estado. Além disso, esse formalismo fundamenta a abordagem de sínteses de controladores utilizada no trabalho, o que o torna uma escolha natural. Alguns conceitos relacionados a autômatos e linguagens são apresentados a seguir como forma de preparar a estrutura teórica do restante do trabalho [21].

C. Caracteres e linguagens

A estrutura mais básica para a construção de um modelo de um SED usado por Autômatos e Linguagens é o *alfabeto*, o que nesse trabalho é denotado por Σ. Σ contempla o conjunto de eventos que compõem o SED e em geral é assumido como um conjunto finito e não-vazio. O conjunto de todas as possíveis cadeias finitas compostas por eventos de Σ é denotado por Σ^*, incluindo a *cadeia vazia*, denotada por ϵ, que corresponde a uma sequência sem eventos [17].

Uma *linguagem* L definida sobre um alfabeto Σ é um subconjunto de cadeias em Σ^*. Por exemplo, seja o alfabeto $\Sigma = \{\alpha, \beta, \gamma, \eta\}$, uma possível linguagem sobre Σ seria $L_1 = \{\epsilon, \beta, \eta\alpha, \alpha\alpha\beta\eta\gamma\}$, sendo que β, $\eta\alpha$ e $\alpha\alpha\beta\eta\gamma$ são cadeias formadas por eventos em Σ. Uma característica importante de uma linguagem é que mesmo que o alfabeto seja finito, o conjunto de cadeias da linguagem pode ser infinito [20].

Por ser uma linguagem, um conjunto, todas as operações convencionais sobre conjuntos, como por exemplo a união e a intersecção, estão automaticamente definidas. A somar com essas operações, também podem ser aplicadas a linguagens operações para lidar com tipos especiais de conjuntos, cujos elementos têm a característica de serem cadeias de eventos.

1) Operações Morfológicas: Seja um alfabeto Σ e uma cadeia $s = \alpha\beta\gamma$ com $\alpha, \beta, \gamma \in \Sigma^*$, então:

- α é o prefixo de s;
- β é uma subcadeia de s;
- γ é o sufixo de s;

2) Concatenação: Sejam duas linguagens $L_1, L_2 \subseteq \Sigma^*$, então a concatenação L_1L_2 das duas linguagens é definida como:

$$L_1L_2 = \{s \in \Sigma^* : (s = s_1s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2)\} \quad (1)$$

Assim, uma cadeia de eventos em L_1L_2 pode ser escrita como a concatenação de uma cadeia em L_1 com uma cadeia em L_2.

3) Prefixo-Fechamento: Seja uma linguagem $L \subseteq \Sigma^*$ então o seu prefixo-fechamento $\overline{L}$ é definido por:

$$\overline{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*) \text{ tal que } st \in L\} \quad (2)$$

Em palavras, $\overline{L}$ consiste de todos prefixos de todas as cadeias em L. Em geral, $L \subseteq \overline{L}$, e L é dita prefixo-fechada se $L = \overline{L}$. Assim, L é prefixo-fechada se qualquer prefixo de qualquer cadeia de eventos em L também seja uma cadeia de L [21]. Por exemplo, seja $\Sigma = \{\alpha, \beta, \gamma\}$ e as linguagens $L_1 = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\beta\}$ e $L_2 = \{\epsilon, \gamma, \gamma\beta\beta\}$. L_1 é prefixo-fechada pois o prefixo de qualquer uma de suas cadeias, também é uma cadeia de L_1. Entretanto, L_2 não é prefixo-fechada pois a cadeia $\gamma\beta$ que é prefixo da cadeia $\gamma\beta\beta$, não pertence a L_2.

D. Autômatos Finitos Determinísticos

Um autômato finito (AF) é um diagrama de transição de estados, que permite modelar, de maneira intuitiva, diversos problemas computacionais [17]. Um AF contém um conjunto de estados, no qual a transição entre um estado para um outro é realizada na ocorrência de um evento [22]. Formalmente um AFD pode se descrito como 5-tupla $\langle \Sigma_G, Q_G, q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$ em que:

- Σ_G é o alfabeto finito;
- Q_G é o conjunto finito de estados;
- $q_G^\circ \in Q_G$ é o estado inicial;
- $Q_G^\omega \subseteq Q_G$ é o subconjunto de estados finais;
- $\rightarrow Q_G \times \Sigma_G \times Q_G$ é a relação de transição de estados.

Uma transição é denotada formalmente por $q_1 \xrightarrow{\sigma} q_2$, representando que a ocorrência de um evento $\sigma \in \Sigma$ evoluiu o autômato do estado q_1 para o estado q_2 [17].

O modelo formal descrito acima pode ser ilustrado por um modelo gráfico no formato de grafos dirigidos [20], com os vértices representando os estados e as arestas representando as transições entre estados na ocorrência de um determinado evento. Essa representação visa simplificar a visualização e a analise do autômato [21].

1) Autômatos como reconhecedores de linguagens: Por implementar a noção de estados, eventos e transições, o comportamento de um autômato pode ser descrito pelas diferentes cadeias de eventos produzidas pela transição entre estados. Assim, o comportamento de um autômato pode ser expresso em termos de linguagens, classificadas em:

- *Linguagem gerada* $L(G)$: envolve todas as possíveis cadeias que podem ocorrer em G, a partir de um estado inicial, independentemente de levar para um estado marcado. Formalmente é definida como:

$$L(G) = \{s \in \Sigma^* : G \xrightarrow{s} q \in Q\}$$

- *Linguagem marcada* L^ω: contempla todas as cadeias que transitam o autômato do estado inicial para algum estado pertencente ao conjunto de estados marcados. Formalmente é definida como:

$$L^\omega(G) = \{s \in \Sigma^* : G \xrightarrow{s} q \in Q^\omega\}$$

A partir da linguagem obtida do autômato, pode-se analisar a equivalência entre autômatos. Sejam dois autômatos G_1 e G_2, se $L(G_1) = L(G_2)$ e $L^\omega(G_1) = L^\omega(G_2)$, então G_1 e G_2 são ditos autômatos equivalentes [23]. Outra associação em AFD e linguagens, é que toda linguagem definida por um AFD, pode ser definida por uma linguagem formada por uma expressão regular [20]. [23] reforça essa definição a partir do teorema de Kleene: "Se a linguagem L é regular, existe um autômato G com número finito de estados tal que $L^\omega(G) = L$. Se G tem um número finito de estados, então $L^\omega(G)$ é regular."

2) Composição síncrona de autômatos: Para dois autômatos $A = \langle \Sigma_A, Q_A, q_A^\circ, Q_A^\omega, \rightarrow_A \rangle$ e $B = \langle \Sigma_B, Q_B, q_B^\circ, Q_B^\omega, \rightarrow_B \rangle$, a composição síncrona entre A e B, denotada por $A \parallel B$, é definida pelo autômato

$$A \parallel B = \{\Sigma_A \cup \Sigma_B, Q_A \times Q_B, (q_A^\circ, q_B^\circ), Q_A^\omega \times Q_B^\omega, \rightarrow\},$$

tal que a relação de transição no modelo composto é definida como segue:

- $(q_A, q_B) \xrightarrow{\sigma} (q_A^\iota, q_B^\iota)$, se $\sigma \in \Sigma_A \cap \Sigma_B$. Ou seja, se o evento σ pertence aos alfabetos de ambos autômatos, então ele levará a uma transição em ambos autômatos;
- $(q_A, q_B) \xrightarrow{\sigma} (q_A^\iota, q_B)$, se $\sigma \in \Sigma_A \setminus \Sigma_B$. Se o evento σ pertence ao alfabeto de A mas não de B, então ele levará a uma transição apenas no estado gerado a partir de A.
- $(q_A, q_B) \xrightarrow{\sigma} (q_A, q_B^\iota)$, se $\sigma \in \Sigma_B \setminus \Sigma_A$. Se o evento σ pertence ao alfabeto de B mas não de A, então ele levará a uma transição apenas no estado gerado a partir de B.

Se duas transições forem rotuladas com um mesmo evento σ, comum a Σ_A e Σ_B, então o resultado da composição leva as transições a serem sincronizadas. Do contrário, se σ pertencer a apenas a um dos alfabetos, a sua ocorrência será assíncrona no modelo composto, ou seja, as transições são executadas de modo independente, em qualquer ordem, fenômeno também conhecido por *interleaving* [24].

Embora o produto síncrono esteja sendo apresentado sobre apenas dois autômatos, ele é naturalmente extensível para um número arbitrário de autômatos. Assim, sejam os autômatos $G^i = \langle \Sigma_i, Q_i, q_i^\circ, Q_i^\omega, \rightarrow_i \rangle$, para $i = 1, \cdots, n$, a composição síncrona global pode ser obtida a partir de:

$$G = \|_{i=1}^n G^i, \text{ tal que } \Sigma = \bigcup_{i=1}^n \Sigma_i$$

Uma propriedade a ser levada em conta da composição síncrona é que o modelo composto deve preservar os comportamentos individuais gerados e marcados de cada autômato. Assim, a composição síncrona é útil para representar sistemas complexos modularmente, podendo ser modelados por subsistemas, em geral mais simples, e compostos posteriormente para que se obtenha o modelo global do sistema.

3) Acessibilidade e bloqueio em autômatos: Uma característica importante de um AFD é a acessibilidade de seus estados, ou seja, a partir do estado inicial, é importante saber quais dos estados posteriores podem ser alcançados [23]. Dado um AFD $G = \langle \Sigma_G, Q_G, q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$, podem ser definidos os seguintes tipos de acessibilidade [21]:

- *Estado acessível*: um estado $q \in Q_G$ é acessível se $q^\circ \xrightarrow{s} q$ para algum $s \in \Sigma^*$. Em palavras, um estado q é dito acessível se existe uma cadeia que transite o estado inicial q° até q.
- *Autômato acessível*: se $\forall q \in Q_G$, q for acessível, então G é dito acessível. Ou seja, se todos os estados forem acessíveis, então o autômato é acessível como um todo.
- *Autômato co-acessível*: se cada cadeia $s \in L(G)$ pode ser completada por algum $r \in \Sigma^*$ tal que $sr \in L^\omega(G)$. Em outras palavras, um autômato é co-acessível se a partir de qualquer estado, exista ao menos um caminho que leve a um estado final. Formalmente, a co-acessibilidade também pode ser descrita por $L(G) = \overline{L^\omega(G)}$.
- *Autômato Trim*: G é dito *trim* se ele é acessível e também co-acessível.
- *Autômato não-bloqueante*: Se $L(G) = \overline{L^\omega(G)}$ então G é não bloqueante. Caso exista ao menos uma cadeia $s \in L(G)$ mas $s \notin L^\omega(G)$, então G é dito *bloqueante*.

A partir de G, pode ser obtida a sua componente acessível G_{ac} e co-acessível G_{co}. G_{ac} pode ser obtida a partir da remoção dos estados não-acessíveis de G, bem como as transições associadas a esse estado. De maneira semelhante, eliminando os estados não co-acessíveis de G, bem como suas transições associadas, se obtém G_{co}[23].

IV. Controle Supervisório de SEDs

Estruturada na teoria dos autômatos e linguagens, a *Teoria de Controle Supervisório* (TCS) considera que um conjunto de autômatos é usado para descrever o comportamento de um SED (planta) e suas especificações de controle [1]. Então, operações matemáticas são processadas sobre a composição desses autômatos, resultando em uma lógica de controle que possa supervisionar o sistema de maneira tal que a sua interferência no sistema seja minimamente restritiva e não-bloqueante. Portanto, entende-se que o principal ponto da TCS é permitir a síntese automática de controladores (supervisores) ótimos para SEDs.

A. Planta do sistema

Para aplicar a TCS em um SED, primeiramente deve ser levado em conta os componentes que compões o sistema. Esses componentes são, em geral, elementos individualmente organizados dentro de um arranjo lógico, de modo que a execução conjunta desses elementos determina o comportamento global do sistema, conhecido como *planta*. Sem nenhuma interferência externa ou ação de controle, é dito que a planta representa o comportamento do sistema em *malha aberta*.

Uma maneira de se obter o modelo da planta de um sistema é modelando os seus componentes individuais na forma de subsistemas, de modo tal que a composição desses subsistemas resulte no comportamento global desejado para a planta. Operacionalmente, utilizando a modelagem por autômatos, os subsistemas podem ser representados por AFDs, e a planta global ser obtida através de uma operação automatizada de composição síncrona entre esses autômatos [23].

B. Especificações de controle

O comportamento em malha aberta em alguns casos pode não ser satisfatório para determinada ação de controle pois a linguagem $L(G)$ de uma planta G pode conter cadeias indesejáveis que podem violar condições de acessibilidade e de não-bloqueio, desejadas para o sistema, ou até mesmo apresentar um comportamento inadequado fisicamente, como por exemplo a colisão de dois robôs em uma linha de automação [21].

Assim, para se obter o comportamento que de fato se espera sob controle, são implementadas *especificações* de controle sobre a planta, para que ajustem a planta a um determinado comportamento esperado para quando estiver em funcionamento. Ao ser associada ao modelo de uma planta, uma especificação interfere em determinados eventos possíveis em malha aberta, adequando o comportamento da planta aos requisitos de controle.

No contexto da TCS, uma especificação representa uma ação proibitiva que observa os eventos da planta e desabilita pontualmente eventos considerados proibidos, quando esses puderem ocorrer [1]. Um passo-a-passo que sistematiza a modelagem de uma especificação é descrito em [25]. O comportamento do sistema após a ação de uma especificação sobre uma planta é denominado de *malha fechada*.

C. Supervisor

A ação de uma especificação E sobre uma planta G resulta na proibição de determinados eventos a partir da desabilitação desses eventos em G. Entretanto, a planta pode conter alguns eventos cuja ação de controle é ineficaz e assim não podem ser desabilitados de maneira direta.

Por exemplo, a quebra de uma máquina em uma linha de produção não depende de qualquer ação de controle, tal que uma possível desabilitação desse evento por parte de E poderia ocasionar em uma inconsistência semântica entre o sistema e sua ação de controle [6]. Portanto, é necessário realizar uma distinção entre os eventos, a fim de que a ação de controle tenha conhecimento dos eventos passíveis de desabilitação e daqueles que não podem ser desabilitados.

Nesse sentido, a TCS prevê o particionamento do conjunto de eventos $\Sigma = \Sigma_\mathrm{c} \cup \Sigma_\mathrm{u}$, tal que:

- *Eventos controláveis* denotado por Σ_c é o conjunto de eventos cuja ocorrência pode ser desabilitada na planta.
- *Eventos não-controláveis* denotado por Σ_u é o conjunto de eventos que não podem ser desabilitados diretamente.

A partir da partição do conjunto de eventos, o comportamento do sistema em malha fechada também se altera, pois a ação de controle agora depende de uma estrutura extra, capaz de sintetizar a ideia de controlabilidade. Na TCS, essa

estrutura é representada por um agente chamado de *supervisor*, denotado por S. Basicamente, um supervisor sintetiza e integra a controlabilidade de eventos e as leis de controle impostas pelas especificações sobre a planta [21].

Formalmente, um supervisor S é um mapa $S : L(G) \to 2^{\Sigma}$, associado a uma linguagem $L_S \subseteq L^{\omega}(G)$ que, após qualquer cadeia $s \in L(G)$, observa eventos em G e informa, dentre os eventos possíveis de ocorrer no estado atual, quais devem ser habilitados. Portanto, a ação de controle de S sobre G, denotada por S/G consiste na habilitação de eventos do conjunto de eventos habilitados $S(s) \subseteq \Sigma$ [23], [1]. A Figura 1 demonstra a ação de controle do supervisor sobre a planta em malha fechada.

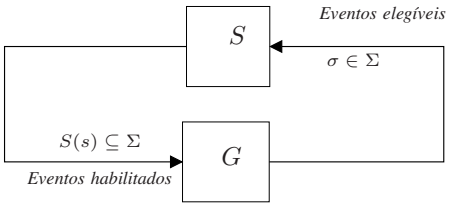

Fig. 1. Ação de controle de um supervisor S sobre uma planta G

Se um evento $\sigma \in \Sigma$ for observado após uma cadeia qualquer $s \in L(G)$, S atualiza o conjunto de eventos habilitados $S(s)$. Caso um evento não faça parte do conjunto $S(s)$ mas seja possível de ocorrer na planta G, então ele está sendo inibido pela ação de controle do supervisor.

O ação de controle S/G apresenta um comportamento gerado em malha fechada, dado por uma linguagem denotada por $L(S/G)$, sendo que $L(S/G)$ é um subconjunto de $L(G)$. O comportamento marcado em malha fechada é dado por $L^{\omega}(S/G) = L(S/G) \cap L^{\omega}(G)$. Assim, $L^{\omega}(S/G)$ consiste de todas as cadeias de $L^{\omega}(G)$ que sobrevive sob a ação de controle de S [21]. Caso $L(S/G) = \overline{L^{\omega}(S/G)}$, então o supervisor S é dito ser não-bloqueante [23]. Dessa maneira, sempre que uma cadeia sobrevive à ação de controle, essa cadeia é um prefixo de uma cadeia marcada, garantindo que a evolução do sistema nunca bloqueie [21] .

D. O Problema de Controle Supervisório

O *Problema de controle supervisório* (PCS), conforme [1], pode ser definido como:

"Seja uma planta G, com eventos em Σ, e uma especificação $E \subseteq \Sigma^*$, definindo um comportamento desejado $K = E \cap L^{\omega}(G)$, encontre um supervisor não-bloqueante S tal que $L^{\omega}(G/S) \subseteq K$".

Em palavras, deseja-se encontrar um supervisor S que controle a planta G, tal que S seja livre de bloqueio, respeitando um conjunto de especificações modeladas por E, e que, ao mesmo tempo, exerça uma ação de controle minimamente restritiva sobre G.

E. Exemplo

Este exemplo se inspira no processo apresentado em [1], o qual é reproduzido na Figura 2.

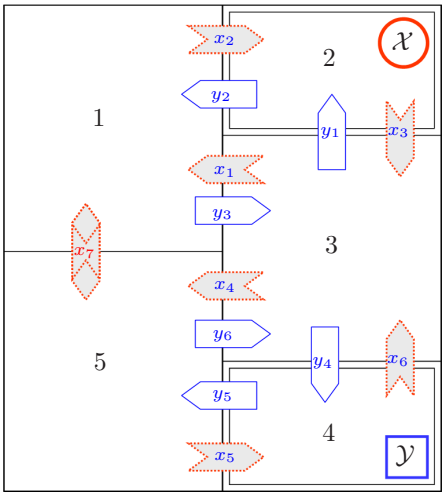

Fig. 2. Processo avaliado como exemplo

O processo é composto por cinco setores através dos quais operam dois robôs, $\mathcal{X}$ e $\mathcal{Y}$. Os robôs são inicialmente posicionados nos setores 2 e 4, respectivamente, os quais também são os setores aceitos para que os robôs finalizem as atividades. Os caminhos permitidos para $\mathcal{X}$ são identificados por setas cinza-pontilhadas e as transições entre os setores são rotuladas pelos eventos x_j, para $j = 1, \cdots, 7$. Já os caminhos permitidos para $\mathcal{Y}$ são identificados por setas branca-contínuas e as transições entre os setores são rotuladas pelos eventos y_k, para $k = 1, \cdots, 6$. O caminho que conecta os setores 1 e 5 é assumida ser bidirecional para o robô $\mathcal{X}$. Assim, para fins de modelagem, assume-se que x_7 é um evento não-controlável enquanto todos os outros são controláveis.

As Figuras 3 e 4 apresentam respectivamente as versões modulares dos autômatos representando o comportamento dos robôs $\mathcal{X}$ e $\mathcal{Y}$ em malha aberta. O comportamento modular de cada planta é semelhante. Por exemplo, a planta $G^3_{\mathcal{X}}$ representa a movimentação do robô $\mathcal{X}$ pelo setor 2. Na ocorrência do evento x_2, $\mathcal{X}$ transita para o setor 2, representado pelo estado marcado em $G^3_{\mathcal{X}}$. A saída desse setor é dada pelo evento x_3, retornando $G^3_{\mathcal{X}}$ para o estado inicial que representa que $\mathcal{X}$ se encontra fora do setor 2.

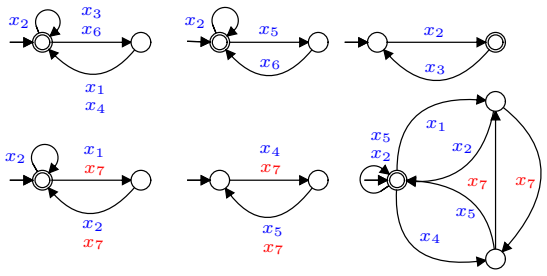

Fig. 3. Autômatos modulares $G_{\mathcal{X}} = \|_{i=1,\cdots,6}\, G^i_{\mathcal{X}}$ modelando $\mathcal{X}$

A planta do sistema, formada pela composição das plantas modulares, é modelada pelo autômato $G = G_{\mathcal{X}} \,\|\, G_{\mathcal{Y}}$ com 36

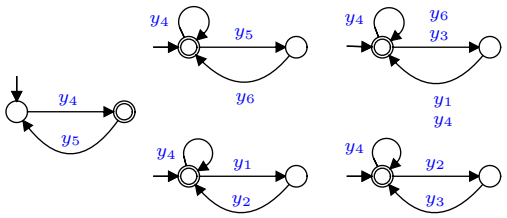

Fig. 4. Autômatos modulares $G_{\mathcal{Y}} = \|_{j=1,\cdots,5}\ G^j_{\mathcal{Y}}$ modelando $\mathcal{Y}$

estados e 96 transições.

Assume-se que o objetivo de controle para o ambiente é realizar a exclusão mútua dos robôs em cada setor, ou seja, $\mathcal{X}$ e $\mathcal{Y}$ não podem ocupar o mesmo setor, ao mesmo tempo. Para isso, é realizada a modelagem individual de cada especificação para cada setor, obtendo-se assim as especificações E_i para $i = 1, \cdots, 5$ representando o respectivo setor, como apresenta a Figura 5.

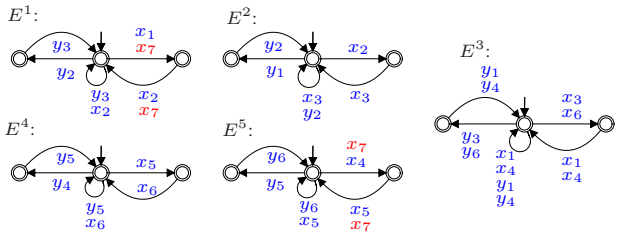

Fig. 5. Especificações para exclusão mútua

A composição das especificações $E = \|_{i=1,\cdots,5}\ E^i$ é um autômato contendo 174 estados e 632 transições. A composição $K = E \parallel G$ contém 29 estados e 60 transições, que pode conter um supervisor $\sup\mathcal{C}(K, G)$ modelado por um autômato S contendo 17 estados e 31 transições.

F. Um problema motivador

Para o exemplo anterior, suponha que $\mathcal{X}$ e $\mathcal{Y}$ são robôs coletores que visitam cada setor e carregando e descarregando material. Assume-se que apenas os setores 2,3 e 4 contém material para carga e descarga, enquanto os setores 1 e 5 são apenas para transição e podem ser compartilhados pelos robôs a qualquer instante. A figura 6 apresenta esse novo ambiente.

Para esse novo cenário, suponha que a planta do sistema possa assumir dois contextos distintos. Quando ambos os robôs $\mathcal{X}$ e $\mathcal{Y}$ estão vazios, eles podem compartilhar os setores 2, 3 e 4 até que ocorra algum carregamento, o que é modelado pelos eventos $l^{\mathcal{X}}_i, i = 2, 3, 4$ e $l^{\mathcal{X}}_j, j = 2, 3, 4$. Ao ocorrer o carregamento, é aplicada a exclusão mútua em todos os setores até que ocorra um descarregamento, o que é modelado pelos eventos $u^{\mathcal{X}}$ e $u^{\mathcal{Y}}$, quando os robôs podem voltar a compartilhar o mesmo setor.

Observe que, na ocorrência de um carregamento, o comportamento da ação de controle é similar a apresentada anteriormente, com o intuito de realizar a exclusão mútua. Entretanto, ao ocorrer a descarga, o compartilhamento dos setores é

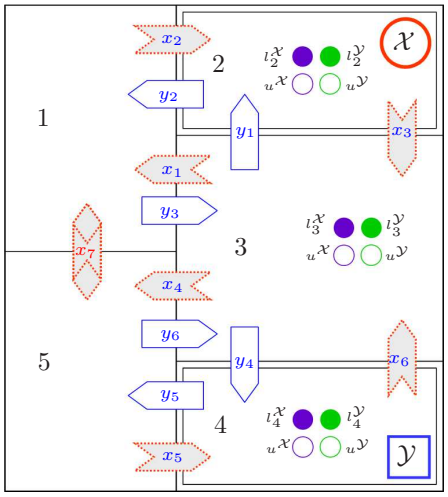

Fig. 6. Ilustração do ambiente considerando $\mathcal{X}$ e $\mathcal{Y}$ como robôs coletores

permitido aos robôs, e a ação de controle deve suspender a exclusão mútua.

A solução de controle esperada, para esse caso, deve ser flexível e auto-configurável, de tal maneira que inicie a exclusão mútua dos robôs quando um evento de carregamento é observado, e desabilite a exclusão mútua quando um evento de descarregamento é observado, representado uma troca de contexto de controle.

Na prática, a implementação de um controle supervisório com troca de contexto pode ser dispendioso em termos de esforço de modelagem e de custo computacional, pois a ação de controle pode envolver diversas trocas de contextos e, para cada contexto, as especificações de controle se alterem de uma maneira que exija a síntese de um supervisor diferente. Assim, torna-se desejável a obtenção de um supervisor que detenha características de dinamismo e flexibilidade, o que não é o caso do supervisor S obtido anteriormente, que se limita à prática de apenas uma única ação de controle.

Diante desse novo contexto assumido pelo problema de controle, observa-se que a solução obtida conforme a TCS convencional se torna degradada pela necessidade de dinamismo do controlador em adaptar-se ao contexto da planta. Esse tipo de ambiente é mais naturalmente representável utilizando variáveis associadas ao autômato que modela a planta, com a finalidade de processar o contexto de determinada informação que interfira diretamente na ação de controle. A abordagem que incorpora variáveis na TCS é então sugerida como forma de tratar do problema, e é apresentada a seguir.

V. Autômatos Finitos Estendidos

Um *Autômato Finito Estendido* (AFE) é uma estrutura de estados, similar aos AFs, mas estendida com fórmulas sobre as transições do modelo, responsáveis por atualizar valores de variáveis nos estados alcançados após a transição [26].

A. Conjuntos de variáveis

Uma *variável* v é uma entidade associada a um domínio $\mathrm{dom}(v)$ e um valor inicial $v^\circ \in \mathrm{dom}(v)$. Um conjunto de variáveis $V = \{v_0, \ldots, v_n\}$ contém o domínio $\mathrm{dom}(V) = \mathrm{dom}(v_0) \times \cdots \times \mathrm{dom}(v_n)$, e um elemento de $\mathrm{dom}(v)$ é denotado por $\bar{v} = (\bar{v}_0, \ldots, \bar{v}_n) \in \mathrm{dom}(V)$ com $\bar{v}_i \in \mathrm{dom}(v_i)$, denotado da *valoração*, sendo que uma valoração é uma estrutura que atribui para cada variável $v \in \mathrm{dom}(V)$ um valor correspondente ao domínio.

Por exemplo, seja $V = a, b, c$ um conjunto de variáveis com $\mathrm{dom}(a) = \mathrm{dom}(b) = \mathrm{dom}(c) = 0, \ldots, 10$. Uma possível valoração (em um caso determinístico) para $\mathrm{dom}(V)$ pode ser, por exemplo $(0, 0, 0)$, que também é a valoração inicial, ou também $(2, 6, 4)$. Caso uma variável seja booleana, uma valoração atribui apenas valores *verdadeiro* ou *falso*.

Um segundo conjunto de variáveis, denominado *variáveis de próximo estado* e denotado por $V' = \{\, v' \mid v \in V \,\}$ com $\mathrm{dom}(V') = \mathrm{dom}(V)$, é usado para descrever como as variáveis são atualizadas pelas transições [26]. Assim, em um AFE, quando uma transição é disparada e uma fórmula implementada sobre essa transição atualiza alguma variável, a transição na verdade está alterando o valor de uma variável associada ao estado corrente para um novo valor a ser associado ao próximo estado [27].

B. Estrutura lógica de atualização de variáveis

Seja V um conjunto de variáveis associadas a um AFE, a implementação de fórmulas lógicas para mapear o valor atual da variável definido em V com um valor de próximo estado definido em V' [9], na ocorrência das transições, leva em conta a semântica que se deseja expressar. Tal implementação dispõe dos recursos de constantes, operados lógicos, matemáticos e relacionais, além de dispor da própria combinação de variáveis [27].

Por exemplo, seja:

- α uma variável que representa números inteiros;
- $\mathrm{dom}(\alpha) = \{0, \cdots, 9\}$;
- $\alpha^\circ = 0$.

Uma transição com a fórmula de atualização $\alpha' = \alpha + 1$, atualiza no próximo estado o valor atual da variável acrescentada de 1, caso α seja menor que 9. Caso $\alpha = 9$, a adição de uma unidade ao valor 9 está fora de $\mathrm{dom}(\alpha)$, assim a atualização não pode ser efetivada e a transição é desabilitada.

Já uma transição com a fórmula de atualização $\alpha' = 5$ atualiza no próximo estado o valor da variável para 5, independente do seu valor atual. Nesse caso, como $5 \in \mathrm{dom}(\alpha)$, esta transição sempre é habilitada. Em contraste, uma fórmula como $\alpha = 5$ compara o valor atual de α com 5 no estado atual, sem realizar nenhuma atualização. Nesse caso, se a comparação for verdadeira, a transição é habilitada, caso o contrário, não.

C. Definição formal e explícita de um AFE

Formalmente, um AFE pode ser descrito por uma 6-upla $G_\mathrm{v} = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$, sendo:

- Σ o alfabeto de eventos;
- $V = \{v_1, \ldots, v_n\}$ o conjunto de variáveis;
- Q o conjunto finito de estados;
- $Q^\circ \subseteq Q$ o conjunto de estados iniciais;
- $Q^\omega \subseteq Q$ o conjunto de estados marcados;
- $\rightarrow \subseteq Q \times \Sigma \times \mathcal{F} \times Q$ a relação de transição entre estados, onde $\mathcal{F}$ é o conjunto de fórmulas sobre $V \cup V'$.

Denota-se por $x \overset{\sigma:p}{\rightarrow} y$ uma transição em G_v que parte do estado x para o estado y, com o evento $\sigma \in \Sigma$ e a fórmula $p \in \mathcal{F}$.

Um AFE G_v também pode ser interpretado de maneira explícita como um AF, em que os valores das variáveis são descritos explicitamente como parte de cada estado. Assim, seja um AF $G = \langle \Sigma, Q_G, Q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$ em que:

- $Q_G = Q \times \mathrm{dom}(V)$;
- $Q_G^\circ = Q^\circ \times \{(v_1^\circ, \ldots, v_n^\circ)\}$;
- $Q_G^\omega = Q^\omega \times \mathrm{dom}(V)$;
- $\rightarrow$ é tal que $(x, \bar{v}) \overset{\sigma}{\rightarrow} (y, \bar{v}')$ para $\bar{v}, \bar{v}' \in \mathrm{dom}(V)$, se há $x \overset{\sigma:p}{\rightarrow} y$ tal que $p(\bar{v}, \bar{v}') = verdade$.

A relação de transição é definida com base na relação em G_v, porém levando em conta os efeitos das fórmulas lógicas implementadas sobre as variáveis [26], [27].

A relação de transição também pode ser estendida para cadeias em Σ^*, assim para um AFE G_v:

- $(x, \bar{v}) \overset{\varepsilon}{\rightarrow} (x, \bar{v})$ para todo $(x, \bar{v}) \in Q_G$;
- $(x, \bar{v}) \overset{s\sigma}{\rightarrow} (x'', \bar{v}'')$ se $(x, \bar{v}) \overset{s}{\rightarrow} (x', \bar{v}') \overset{\sigma}{\rightarrow} (x'', \bar{v}'')$ para algum $(x', \bar{v}') \in Q_G$.

Denota-se por $G_\mathrm{v} \overset{s}{\rightarrow} (x, \bar{v})$ o estado $(x, \bar{v})$ que pode ser alcançado a partir do estado inicial de G_v. Assim o AFE G_v pode ter linguagens representando o seu comportamento gerado e marcado, definidas respectivamente por [26]:

$$L(G_\mathrm{v}) = \{\, s \in \Sigma^* \mid G_\mathrm{v} \overset{s}{\rightarrow} (x, \bar{v}) \in Q_G \}; \text{ e}$$

$$L^\omega(G_\mathrm{v}) = \{\, s \in \Sigma^* \mid G_\mathrm{v} \overset{s}{\rightarrow} (x, \bar{v}) \in Q_G^\omega \}$$

D. Controle Supervisório de SEDs com AFEs

Caso a planta e a especificação de um sistema sejam modelados por um AFE, a condição de controlabilidade também passa a ser estendida para considerar as variáveis, denotada por *V-controlabilidade* [26]. Assim, seja uma planta $G_\mathrm{v} = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$ e uma especificação $E_\mathrm{v} = \langle \Sigma, V, Q, Q^\circ, Q^\omega, \rightarrow \rangle$, então E_v é V-controlável em relação a G_v se a seguinte implicação for verdadeira:

Para todo $s \in \Sigma^*, \mu \in \Sigma_\mathrm{u}, x_e \in Q_E, x_g, x_g' \in Q_G$ e $\bar{v}$, $\bar{v'} \in \mathrm{dom}(V)$:

$$\text{se } E_\mathrm{v} \overset{s}{\rightarrow} (x_e, \bar{v}) \wedge G_\mathrm{v} \overset{s}{\rightarrow} (x_g, \bar{v}) \overset{\mu}{\rightarrow} (x_g', \bar{v'})$$

$$\text{então existe um } x_e' \in Q_E \text{ tal que } E_\mathrm{v} \overset{s}{\rightarrow} (x_e, \bar{v}) \overset{\mu}{\rightarrow} (x_e', \bar{v'}).$$

Em palavras, uma especificação modelada por E_v é V-controlável se não proíbe nenhum evento não controlável μ na planta G_v. A V-controlabilidade se difere da controlabilidade convencional, pelo fato que a especificação não deve apenas habilitar todos os eventos não controláveis que são possíveis na planta, mas também mantém o comportamento de atualização

das variáveis do mesmo modo que a planta o faz. Entretanto, para eventos controláveis que podem ser desabilitados pela especificação, ela pode desabilitar a atualização das variáveis.

Seja um comportamento esperado sob controle $E_{\mathrm{v}} \parallel G_{\mathrm{v}}$, define-se o conjunto:

$$\mathcal{C}_V = \{K_{\mathrm{v}} \subseteq E_{\mathrm{v}} \parallel G_{\mathrm{v}} | K_{\mathrm{v}} \text{ é V-controlável em relação a } G_{\mathrm{v}}\}$$

$\mathcal{C}_V$ contém um elemento supremo (AFE) denotado por $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$, que representa o comportamento mais permissivo possível de ser implementado em G_{v} atendendo as especificações de controle E_{v}.

Em [26], a condição de controlabilidade de AFEs é apresentada em mais detalhes. Em [27] propõe-se um algoritmo para a síntese automática de supervisores a partir de AFEs, bem como apresenta-se detalhes sobre as propriedades desse supervisor.

Além da controlabilidade, é desejado que o comportamento $\sup\mathcal{C}_V$ seja não bloqueante. De maneira formal, o não-bloqueio de um autômato $A_{\mathrm{v}} = \langle \Sigma, V, Q, Q^{\circ}, Q^{\omega}, \rightarrow \rangle$ pode ser assim definido:

Para todo $s \in \Sigma^*, t \in \Sigma, x, y \in Q_A, y \in Q_A^{\omega}, \bar{v}, \bar{w} \in V$,

$$A_{\mathrm{v}} \xrightarrow{s} (x, \bar{v})$$

implica em

$$(x, \bar{v}) \xrightarrow{t} (y, \bar{w}),$$

para algum $(y, \bar{w}) \in Q_A^{\omega}$.

Se A_{v} for não-bloqueante, então sua linguagem $L(A_{\mathrm{v}})$ também é não-bloqueante [27]. A característica de não-bloqueio de um AFE é semelhante a um de AFD, assim uma caso uma cadeia $s \in L(A_{\mathrm{v}})$ e também $s \in L^{\omega}(A_{\mathrm{v}})$, então o AFE é não-bloqueante.

E. Problema de Controle Supervisório com Variáveis

O PCS pode ser reformulado para o contexto do *Controle Supervisório com Variáveis* (PCS-V) [26] como:

Dados os AFEs Q-determinísticos E_{v} e G_{v} que modelam respectivamente a especificação e a planta de um SED, encontre um subautômato não-bloqueante $K_v \subseteq E_{\mathrm{v}} \parallel G_{\mathrm{v}}$ que seja V-controlável em relação a G_{v}. .

Assim, se $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ é não-bloqueante, então ele é a solução ótima para o PCS-V e pode ser usado para implementar um supervisor que desabilita todos os eventos controláveis possíveis em G_{v} que não são elegíveis em $\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})$ [26]. A solução do PCS-V também pode ser associada a PCS desde que seja garantida a equivalência da representação do SED e suas especificações.

Seja G e E AFDs que modelam respectivamente a planta e a especificação de um SED, definindo um comportamento desejado K. Seja G_{v} a versão explícita de um AFE modelando a planta de um SED e E_{v} um AFE que apresenta os mesmos requisitos de controle que E, tal que

$$L(G) = L(G_{\mathrm{v}}),$$

$$L(E \parallel G) = L(E_{\mathrm{v}} \parallel G_{\mathrm{v}}) \text{ e}$$

$$L^{\omega}(E \parallel G) = L^{\omega}(E_{\mathrm{v}} \parallel G_{\mathrm{v}}).$$

Então, é provado em [27] que

$$L(\sup\mathcal{C}_V(E_{\mathrm{v}}, G_{\mathrm{v}})) = \sup\mathcal{C}(K, G).$$

Dessa maneira, garante-se que a equivalência da solução de controle seja o sistema modelado por um AF ou por um AFE.

F. Exemplo revisitado

Agora, o exemplo apresentado na Seção IV-E é revisitado com o objetivo de mostrar como o contexto do sistema pode ser armazenado em variáveis que são associadas ao modelo da planta. Na sequência, reintroduziremos as especificações de controle, que agora utilizam os valores das variáveis para a permissão da ocorrência de eventos. Finalmente, sintetizamos um supervisor que reconhece o contexto trazido pelas variáveis e atue de acordo com o mesmo.

G. Modelagem da planta

Primeiramente, reforça-se que a estrutura do ambiente da Figura 6 é a mesma da Figura 2. Sendo assim, o comportamento de transição entre os setores dos robôs $\mathcal{X}$e $\mathcal{Y}$ permanece o mesmo, e os autômatos $G_{\mathcal{X}}$ (Figura 3) e $G_{\mathcal{Y}}$ (4) são utilizados na modelagem do comportamento em malha aberta dos robôs.

Entretanto, agora deve ser levado em consideração também 8 novos eventos acrescentados ao ambiente, $l_i^{\mathcal{X}}$, $l_i^{\mathcal{Y}}$, $u^{\mathcal{X}}$, $u^{\mathcal{Y}}$, para $i = 2, 3, 4$, em que o valor de i identifica o setor em que é realizado o carregamento, l e u representam a carga e descarga, respectivamente, e $\mathcal{X}$ e $\mathcal{Y}$ identificam o robôs relacionados ao evento.

A ação de carga e descarga dos robôs é responsável pela troca de contexto da ação de controle, assim a flexibilidade do controlador deve atuar na ocorrência desses eventos. Com esta finalidade, associa-se um conjunto de variáveis ao modelo da planta para armazenar o contexto do sistema dada a ocorrência dos eventos $l_i^{\mathcal{X}}$, $l_i^{\mathcal{Y}}$, $u^{\mathcal{X}}$, $u^{\mathcal{Y}}$.

Assim, são utilizadas duas variáveis, $\mathrm{v}^{\mathcal{X}}$e $\mathrm{v}^{\mathcal{Y}}$, declaradas com um domínio booleano $\mathrm{dom}(\mathrm{v}^{\mathcal{X}}) = \mathrm{dom}(\mathrm{v}^{\mathcal{Y}}) = \{V, F\}$, com valor inicial $\mathrm{v}^{\mathcal{X}^{\circ}} = \mathrm{v}^{\mathcal{Y}^{\circ}} = F$, indicando que ambos robôs iniciam descarregados.

A semântica de atualização de valores de $\mathrm{v}^{\mathcal{X}}$ e $\mathrm{v}^{\mathcal{Y}}$ é dada pelos eventos de carga e descarga, sendo assim necessário associar as transições que contenham esses eventos com fórmulas que utilizem as variáveis. Assim, é modelado a planta da Figura 7.

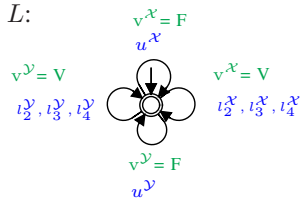

Fig. 7. Modelo da planta responsável pela troca de contexto

Por exemplo, quando algum evento $l_i^{\mathcal{X}}$ ocorre, $\mathrm{v}^{\mathcal{X}}$ é atualizada para *verdadeiro* pela formula $\mathrm{v}^{\mathcal{X}} = V$, indicando que o robô $\mathcal{X}$ está carregado. Ao ocorrer um evento $u^{\mathcal{X}}$, o robô $\mathcal{X}$ está novamente descarregado, e a variável $\mathrm{v}^{\mathcal{X}}$ é atualizada para o valor *falso*. O mesmo comportamento ocorre aos eventos relacionados ao robô $\mathcal{Y}$.

Agora, o comportamento global da planta do sistema é modelado por $\mathcal{G} = G_{\mathcal{X}} \parallel G_{\mathcal{Y}} \parallel L$, que corresponde a um autômato com 144 estados e 1596 transições.

H. Modelagem das especificações

Primeiramente, considera-se que o modelo L da Figura 7 necessita ser sincronizado na ocorrência dos eventos de carga e descarga, para a correta implementação da semântica de atualização de $\mathrm{v}^{\mathcal{X}}$ e $\mathrm{v}^{\mathcal{Y}}$. Por exemplo, deseja-se que o evento $u^{\mathcal{X}}$ apenas ocorra depois de algum evento $l_i^{\mathcal{X}}$. Este comportamento é representado pelos modelos de especificações Fl^1 e Fl^2, apresentados na Figura 8

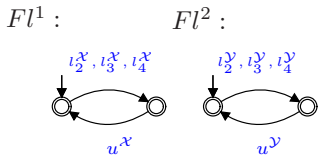

Fig. 8. Modelos de especificações para carga e descarga de cada robô

Também é necessário realizar uma sincronização entre o comportamento da planta L com a planta dos robôs $\mathcal{G} = G_{\mathcal{X}} \parallel G_{\mathcal{Y}}$, para que os eventos de carga $l_i^{\mathcal{X}}$ e $l_i^{\mathcal{Y}}$ apenas sejam permitidos quando cada robô estiver presente no respectivo setor i. Esta especificação é modelada por $UL^{\mathcal{X}} = \|_{j=2,3,4}\, UL_j^{\mathcal{X}}$ e $UL^{\mathcal{Y}} = \|_{j=2,3,4}\, UL_j^{\mathcal{Y}}$ na Figura 9.

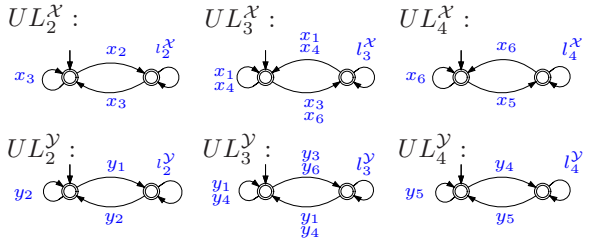

Fig. 9. Especificações para a coordenação de carregamento em cada setor

Por exemplo, o autômato $UL_2^{\mathcal{X}}$ está permitindo a ocorrência do carregamento do robô $\mathcal{X}$ no setor 2 apenas quando o mesmo adentrar esse setor (evento x_2). Ao sair do setor (evento x_3), o evento $l_2^{\mathcal{X}}$ passa a ser proibido pelo modelo. Assim, o evento de carga $l_2^{\mathcal{X}}$ está associado respectivamente e unicamente ao setor 2. A mesma semântica segue para os outros setores em ambos robôs.

Por fim, pode ser introduzido novas versões para a exclusão mútua de cada setor, de uma maneira que seja incorporado o contexto expresso pela planta através das variáveis. A exclusão mútua apenas deve ser aplicada quando pelo menos um dos robôs está carregado.

Como a planta $\mathcal{G}$ carrega informações sobre o contexto das variáveis, o modelo da especificação apenas precisa realizar a verificação do valor das variáveis, e validação de determinados eventos, conforme o contexto apresentado. Sendo assim, é necessário que a especificação que aplique a exclusão mútua seja representada por um AFE que contenha apenas fórmulas que apliquem restrições, as quais são também conhecidas como fórmulas de "guarda". Tal comportamento é expresso nos autômatos $\mathcal{E}^2, \mathcal{E}^3, \mathcal{E}^4$ da Figura 10.

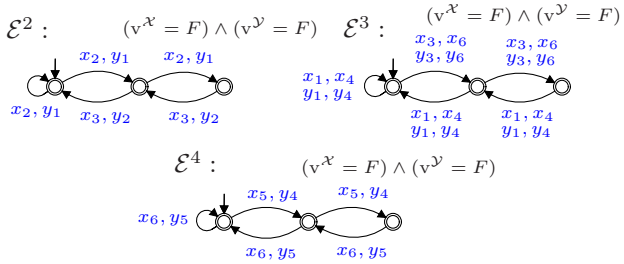

Fig. 10. Especificações $\mathcal{E}^2$, $\mathcal{E}^3$ and $\mathcal{E}^4$ aplicando a exclusão mútua dos setores 2, 3 e 4, respectivamente

Para explicar a ação das especificações da Figura 10, toma-se como exemplo a especificação $\mathcal{E}^2$, responsável por realizar a exclusão mútua no setor 2. No estado inicial, a primeira ocorrência de algum evento x_2 ou y_1 (representando que algum dos robôs entrou no setor), é sempre habilitada, pois até então o setor se encontrava vazio. Ao adentrar o setor, o robô pode deixá-lo a qualquer instante, pelos eventos x_3 (caso $\mathcal{X}$ esteja dentro) ou y_2 (caso $\mathcal{Y}$ esteja dentro), retornando ao estado inicial.

Entretanto, nesse mesmo estado, caso um robô já esteja dentro do setor e o outro robô tentar entrar (segunda ocorrência dos eventos x_2 ou y_1), a especificação irá analisar o contexto das variáveis de carga naquele instante, através da formula de guarda $(\mathrm{v}^{\mathcal{X}} = F) \wedge (\mathrm{v}^{\mathcal{Y}} = F)$. A guarda tem a função de verificar se ambos os robôs estão descarregados. Caso a resposta da guarda seja V, então ambos robôs estão vazios e transição é permitida. Caso contrário, a resposta da guarda seja F, então pelo menos um dos robôs se encontra carregado, assim não é permitida uma ocorrência de x_2 ou y_1, ao menos que o robô carregado (ou ambos) se descarreguem ou algum dos robôs deixe o setor (eventos x_3 ou y_2).

A composição das especificações é modelada por

$$\mathcal{E} = \mathcal{E}^2 \parallel \mathcal{E}^3 \parallel \mathcal{E}^4 \parallel Fl^1 \parallel Fl^2 \parallel UL^{\mathcal{X}} \parallel UL^{\mathcal{Y}},$$

que corresponde a um autômato com 1368 estados e 16994 transições.

I. Síntese do controlador

Para o procedimento de síntese do controlador para o exemplo, foi utilizada a ferramenta *Supremica* [28], para realização de todas as operações de produto síncrono e sintetização entre os modelos, bem como a verificação e a validação quanto ao não bloqueio e a controlabilidade do supervisor $\sup\mathcal{C}_V(\mathcal{E}, \mathcal{G})$.

Assim, para o exemplo, o supervisor resultou em um AFE contendo 157 estados e 722 transições.

Testes realizados no Supremica mostram um comportamento compatível com o desejado sob controle. Ademais, a fundamentação teórica apresentada em [26], [27] garantem formalmente a compatibilidade e a otimidade desse resultado em relação a uma solução que eventualmente pudesse ser obtida por meio de autômatos convencionais, sem variáveis.

VI. Conclusão

Este trabalho apresenta uma abordagem para a modelagem e síntese de controle que pode ser utilizada em um ambiente dinâmico e concorrente de coordenação de múltiplos robôs. Nesse ambiente, o robô se adapta automaticamente ao contexto apresentado pelo sistema, alterando a ação de controle em tempo de execução. Para isso tornar-se possível, o sistema é modelado por AFEs, formalismo que permite que o modelo de um SED possa ser enriquecido com variáveis para armazenamento de contexto do sistema. Assim, fórmulas são associadas ao modelo das transições, tanto no sentido de atualizar as variáveis quanto de implementar restrições ao sistema, com base em condições lógicas implementadas sobre os valores das variáveis.

A abordagem é ilustrada por meio de um exemplo, em que dois robôs circulam através de um ambiente dividido por setores. No exemplo, a ação de controle imposta se difere dependendo se os robôs estão carregando algum material ou não, e o contexto de controle é alterado pelo controlador de maneira dinâmica em tempo de execução.

Resultados sugerem que a proposta é tecnicamente válida e viável de ser implementada em escala industrial. Como trabalhos futuros, pretende-se focar na complementação da abordagem com uma estrutura formal mais consistente e ampla. Ainda, pretende-se focar na implementação de ferramental relacionado ao tema. Embora a literatura forneça ferramentas automatizadas para a geração de código a partir de autômatos, tais ferramentas são, em geral, estabelecidas com base em autômatos convencionais, o que dificulta a manipulação das estruturas lógicas dos AFEs. Para finalizar, pretende-se ainda expandir a abordagem para outras classes de problemas computacionais, os quais se mostrem imersos em ambientes cuja dinâmica é volátil em relação aos inúmeros contextos possíveis.

References

[1] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[2] S. M. A. K. Fei, Z. and B. Lennartson, "Efficient supervisory synthesis to large-scale discrete event systems modeled as extended finite automata," Chalmers University of Technology, Tech. Rep. 5, 2012.

[3] P. J. Ramadge and W. M. Wonham, "Modular feedback logic for discrete event systems," *SIAM Journal of Control and Optimization*, vol. 25, no. 5, pp. 1202–1218, 1987.

[4] M. H. D. Queiroz and J. E. R. Cury, "Modular supervisory control of large scale discrete event systems," in *Discrete Event Systems: Analysis and Control. Proc. WODES'00*. Kluwer Academic, 2000, pp. 103–110.

[5] M. Teixeira, R. Malik, J. Cury, and M. de Queiroz, "Supervisory control of des with extended finite-state machines and variable abstraction," *Automatic Control, IEEE Transactions on*, vol. 60, no. 1, pp. 118–129, Janeiro 2015.

[6] J. E. R. Cury, M. H. de Queiroz, G. Bouzon, and M. Teixeira, "Supervisory control of discrete event systems with distinguishers," *Automatica*, vol. 56, pp. 93–104, 2015.

[7] K.-T. Cheng and A. Krishnakumar, "Automatic generation of functional vectors using the extended finite state machine model," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 1, no. 1, pp. 57–79, Janeiro 1996.

[8] Y.-L. Chen and F. Lin, "Modeling of discrete event systems using finite state machines with parameters," in *Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on*, 2000, pp. 941–946.

[9] M. Skoldstam, K. Akesson, and M. Fabian, "Modeling of discrete event systems using finite automata with variables," in *Decision and Control, 2007 46th IEEE Conference on*, Dezembro 2007, pp. 3387–3392.

[10] Y. Yang, A. Mannani, and P. Gohari, "Implementation of supervisory control using extended finite-state machines," *International Journal of Systems Science*, vol. 39, no. 12, pp. 1115–1125, 2008.

[11] J. Zhao, Y.-L. Chen, Z. Chen, F. Lin, C. Wang, and H. Zhang, "Modeling and control of discrete event systems using finite state machines with variables and their applications in power grids," *Systems & Control Letters*, vol. 61, no. 1, pp. 212–222, 2012.

[12] M. Teixeira, R. Malik, J. Cury, and M. de Queiroz, "Variable abstraction and approximations in supervisory control synthesis," in *American Control Conference (ACC), 2013*, Junho 2013, pp. 132–137.

[13] M. Shoaei, L. Feng, and B. Lennartson, "Abstractions for nonblocking supervisory control of extended finite automata," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, Agosto 2012, pp. 364–370.

[14] L. Ouedraogo, R. Kumar, R. Malik, and K. Akesson, "Nonblocking and safe control of discrete-event systems modeled as extended finite automata," *Automation Science and Engineering, IEEE Transactions on*, vol. 8, no. 3, pp. 560–569, Julho 2011.

[15] A. Voronov and K. Akesson, "Verification of supervisory control properties of finite automata extended with variables," Tech. Rep. 03, 2009.

[16] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, vol. 27, no. 6, pp. 509–516, June 1978.

[17] M. Teixeira, "Explorando o uso de distinguidores e de autômatos finitos estendidos na teoria do controle supervisório de sistemas a eventos discretos," Ph.D. dissertation, Universidade Federal de Santa Catarina - UFSC, Florianópolis, 2013, tese de Doutorado (Doutorado em Engenharia de Automação e Sistemas) Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

[18] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[19] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and markov chains*. John Wiley & Sons, Inc, 2000.

[20] R. M. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd ed. ser. Series in Computer Science. Addison-Wesley, 2001.

[21] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York: Springer Science+Bussines Media, LLC, 2008.

[22] H. R. Lewis and C. H. Papadimitriou, *Elementos de Teoria da Computação*, 2nd ed. São Paulo: Artmed Editora S. A., 2008.

[23] J. Cury. (2001, nov) Teoria de controle supervisório de sistemas a eventos discretos. Acessado: 2015-08-30. [Online]. Available: http://user.das.ufsc.br/ cury/cursos/apostila.pdf

[24] C. A. R. Hoare, *Communicating Sequential Processes*, 1985.

[25] M. Teixeira, R. Ribeiro, M. Barbosa, F. Enembreck, and R. Massa, "A modeling architecture for the orchestration of service components in factory automation," in *Emerging Technologies Factory Automation (ETFA), 2015 IEEE 20th Conference on*, 2015, pp. 1–8.

[26] M. Teixeira, R. Malik, J. E. R. Cury, and M. H. de Queiroz, "Variable abstraction and approximations in supervisory control synthesis," in *2013 American Control Conference, ACC'13*, Washington, USA, Jun. 2013, pp. 120–125.

[27] M. Teixeira, R. Malik, J. Cury, and M. de Queiroz, "Supervisory control of des with extended finite-state machines and variable abstraction," *Automatic Control, IEEE Transactions on*, vol. 60, no. 1, pp. 118–129, 2014.

[28] K. Akesson, M. Fabian, H. Flordal, R. Malik, A. Vahidi, M. Skoldstam, and G. Cengic, *Supremica*, 2014. [Online]. Available: http://www.supremica.org/