Implementação de ferramentas para o controle supervisório de sistemas a eventos discretos

Implementation of tools for the supervisory control of discrete event systems

Cristian Roberto Pastro
cristian-pastro@hotmail.com
Universidade Tecnológica Federal do Paraná, Pato Branco, Paraná, Brasil

Marcelo Teixeira
marceloteixeira@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Pato Branco, Paraná, Brasil

RESUMO

A Teoria do Controle Supervisório é um método formal que sintetiza controladores ótimos para Sistemas a Eventos Discretos. Porém o Controle Supervisório tradicional pode implicar na modelagem manual de especificações complexas, desencorajando o seu uso na indústria. Pensando nesse problema foi desenvolvido um plug-in para uma ferramenta de modelagem que refina eventos. Um refinamento é uma instância de um evento original que atribui contexto a sua ocorrência no sistema, tendendo a facilitar tarefas de design. Estima-se que os resultados deste trabalho tenham potencial para favorecer o uso do Controle Supervisório na indústria. O trabalho é ilustrado por meio de um exemplo genérico parcial de um processo industrial.

PALAVRAS-CHAVE: Autômatos. Controle. Refinamento.

ABSTRACT

Supervisory Control Theory is a formal method that synthesizes optimal controllers for Discrete Event Systems. However, traditional Supervisory Control may involve manual modeling of complex specifications, discouraging their use in industry. Thinking about this problem, a plug-in for a modeling tool that refines events was developed. A refinement is an instance of an original event that assigns context to its occurrence in the system, tending to facilitate design tasks. It is expected that the results of this work show potential to favor the use of Supervisory Control in industry. The work is illustrated by a partial generic example of an industrial process.

KEYWORDS: Automata. Control. Refinement.

Recebido: 30 ago. 2018.
Aprovado: 04 out. 2018.

INTRODUÇÃO

Muitos tipos de problema relacionados a automação industrial podem ser resolvidos usando Sistemas a Eventos Discretos (SEDs), um sistema onde o espaço de estados é descrito por um conjunto discreto. Transições de um estado para outro são chamados de eventos, ocorrendo instantaneamente e em tempo arbitrário (CASSANDRAS; LAFORTUNE, 2008). O conjunto de todos os eventos do sistema é chamado alfabeto e é representado por Σ. A Teoria do Controle Supervisório (TCS) de Ramadge e Wonham (1984), agrega aspectos de segurança que permitem ajustar o comportamento do sistema a um modo desejado.

A teoria de SEDs exige a modelagem de plantas, que refletem o comportamento individual de cada componente do sistema, já especificações implementam as restrições necessárias à planta. A composição síncrona é uma operação que integra o comportamento de dois ou mais autômatos. Utilizando a composição síncrona podemos criar o modelo geral das plantas G realizando a composição de todas as plantas. Do mesmo modo o modelo geral das especificações E é feito compondo-se todas as especificações. O modelo geral do sistema K pode ser obtido compondo-se os modelos G com E.

A TCS exige a divisão do alfabeto de eventos Σ em dois: eventos controláveis e não controláveis. Após esta divisão é possível aplicar algoritmos que a partir do modelo K, removem do sistema estados que podem causar problemas de controlabilidade e/ou bloqueios.

MÁQUINAS DE ESTADOS COM DISTINGUIDORES: UM ESTUDO DE CASO

Os modelos de especificações podem conter centenas de estados para serem modelados, impossibilitando o uso da TCS convencional. Dentre as alternativas estão os distinguidores (CURY et al., 2015). Ao usar distinguidores assume-se que cada evento $\sigma \in \Sigma$passa por uma máscara para um conjunto$\Delta^{\sigma} \neq \phi$, criando um novo conjunto de eventos dado por $\Delta^{\sigma} = \cup_{\sigma \in \Sigma} \Delta^{\sigma}$.

Pode-se definir o um mapa mascarador inverso para qualquer cadeia $s \in \Delta^*$ através da Eq. (1).

$$\Pi^{-1}(s) = \{t \in \Delta^* / \Pi(t) = s\} \quad (1)$$

Distinguidor é um caso particular do mapa Π^{-1}que substitui cada cadeia Σ^* por um conjunto de cadeias em Δ^*, o qual é denotado nesse trabalho por D, conforme a Eq. (2).

$$D\big(L(G)\big) = \Pi^{-1}\big(L(G)\big) \cap L_D \quad (2)$$

Embora abordagens como a de variáveis e distinguidores deem uma sobrevida à TCS e potencializem o seu potencial prático, o seu uso na indústria ainda é restrito e muito disso se deve ao fato de não existirem ferramentas que transfiram essas teorias diretamente para o chão de fábrica. Um exemplo disso pode ser observado na ferramenta Supremica, que mesmo contando com inúmeros algoritmos de síntese e de verificação formal, ela não conta com

ferramentas que possibilitem aplicar os conceitos e mapeamentos referentes à abordagem com distinguidores.

MATERIAIS E MÉTODOS

Para compilação, edição e testes da ferramenta, foi utilizado a IDE Eclipse Oxygen 2 Release (4.7.2) em um Sistema Operacional Ubuntu 16.04 LTS com um ambiente gráfico XFCE. Foi utilizado o Java™ SE Runtime Envoronment build 1.8.0_181-b13. O plug-in foi baseado na ferramenta Waters/Supremica IDE 2.3.1, uma ferramenta desenvolvida originalmente por Akesson et al. (2006).

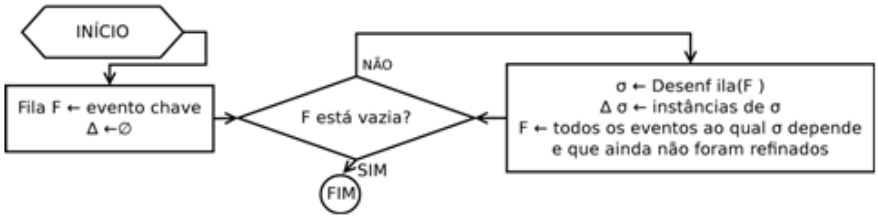
MODELAGEM DE SISTEMAS USANDO DISTINGUIDORES

Para a modelagem usando distinguidores foi elaborado um passo a passo baseado em Cury et al. (2015) que mostrou-se eficaz na maioria dos problemas:

a) **Identificar o(s) evento(s) chave e refiná-lo:** Evento chave é um evento que o seu contexto indica diretamente qual evento deve ser habilitado na nova especificação. Refiná-lo significa abstrair todos os seus contextos importantes para novos sub-eventos em Δ.

b) **Completar o conjunto Δ:** Para esta etapa algoritmo mostrado na Figura 2 tem se mostrado eficaz. Este algoritmo deve ser aplicado a todos os eventos chave encontrados no passo anterior.

Figura 2 – Fluxograma do algoritmo usado para completar o conjunto Δ



Fonte: Autoria Própria (2018).

c) **Criar um modelo preliminar da especificação extra:** Usando os conjuntos refinados no passo anterior, deve-se criar uma versão preliminar da especificação, chamaremos este modelo de H_x'.

d) **Modelar os distinguidores:** Cada distinguidor D_i deve ser modelado para que as instâncias de refinamento sejam habilitadas no momento correto. Após modelar os distinguidores realiza-se a composição síncrona de todos os distinguidores, obtendo o modelo H_d.

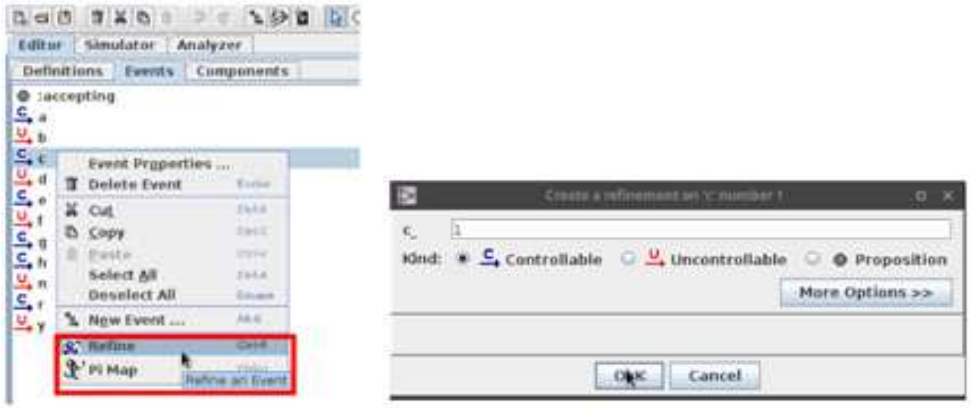
e) **Obter o modelo da especificação extra E_x:** O modelo da especificação E_x pode ser obtido através da composição síncrona $E_x = H_x' \,||\, H_d$.

Vale ressaltar que antes de aplicar a composição síncrona aos modelos de plantas e especificações deve-se converter cada evento $\sigma \in \Sigma$ para o conjunto Δ^{σ} de acordo com a Eq. 2.

RESULTADOS E DISCUSSÃO

A Eq. 2 foi implementada através da função R*efine*. Ao selecionar esta opção para um evento de acordo com a Figura 3 (a), e escolher o número **n** de refinamentos desejados, então o software apresentará **n** janelas para selecionar o nome dos refinamentos de acordo com a Figura 3 (b).

Figura 3 – Aplicação da Eq. (2) no Alfabeto Σ no software Supremica

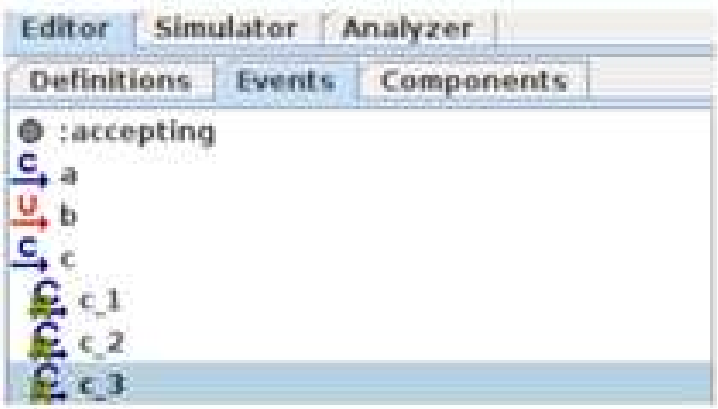


(a) (b)

Fonte: Autoria Própria (2018).

Ainda é possível ver na Figura 2 (a) que há uma opção chamada *Pi Map*, esta opção tem a função de verificar o Mapa de cada evento, apresentando todos os seus refinamentos. É possível notar também na Figura 2 (b) que é possível criar refinamentos controláveis e não controláveis independente das configurações do evento pai, assim como selecionar várias opções para os refinamentos. Após executadas estas opções é possível seguir para os passos *b* e *c* do passo a passo anterior. A Figura 4 mostra a visualização dos refinamentos de um evento qualquer *c*.

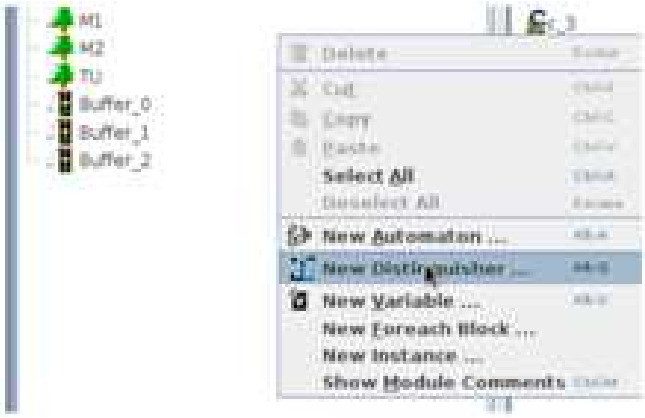
Figura 4 – Apresentação dos eventos refinados no software



Fonte: Autoria própria (2018).

Mais uma entidade fez-se necessária: o distinguidor. O *software* ainda permite criar uma entidade distinguidor para executar o passo *d* do passo a passo mostrado anteriormente. A Figura 5 mostra a opção criada no menu de componentes do Supremica.

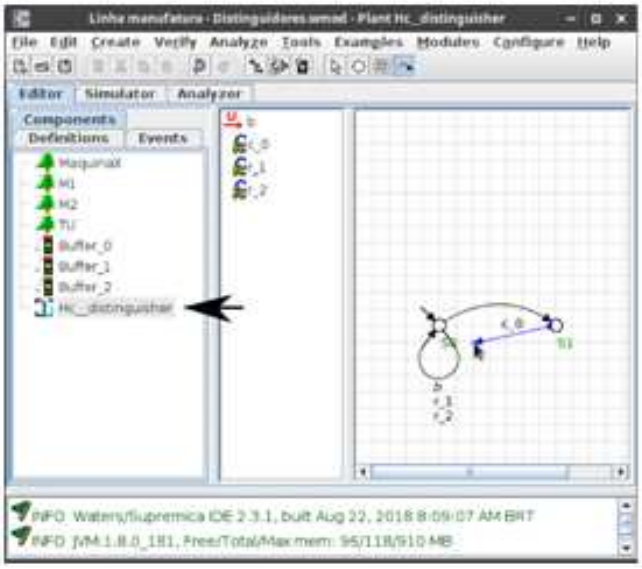
Figura 5 – Apresentação dos eventos refinados no software



Fonte: Autoria própria (2018).

Depois de selecionada a opção *New Distinguisher* o novo componente do sistema pode enfim ser criado. A tela de edição de um *Distinguisher* é mostrada na Figura 6.

Figura 6 – Tela de criação da entidade *Distinguisher*



Fonte: Autoria própria (2018).

Após modelados os distinguidores, o passo *e* do passo a passo anterior pode ser feito através da modelagem de uma especificação comum. Os outros passos podem ser feitos utilizando as ferramentas de síntese já implementadas anteriormente, que tiveram apenas que sofrer pequenos ajustes.

CONCLUSÃO

A TCS tradicional de Ramadge e Wonham (1984) oferece várias ferramentas para modelagem e controle de sistemas a eventos discretos, garantindo sistemas com respostas minimamente restritivas, controláveis e não bloqueantes. Além disso há varias ferramentas como Supremica que colocam em prática a teoria, possibilitando a implementação prática de sistemas.

Em alguns casos, a TCS clássica depende da modelagem de especificações gigantescas, impossíveis de serem feitas manualmente. Surgiu então a necessidade de incrementar a teoria clássica através de abordagens como a de distinguidores. Esta abordagem é capaz de criar especificações muito grandes através de métodos alternativos, com muito mais facilidade e rapidez e trazendo as mesmas garantias da TCS clássica. Porém ainda faltam implementações práticas em softwares computacionais capazes de trazer aos distinguidores e a TCS de forma geral aplicações reais na indústria.

O plug-in criado ameniza este problema, criando ferramentas para síntese e modelagem de autômatos com distinguidores e possibilitando assim uma maior utilização das teorias vistas neste trabalho em ambientes industriais.

REFERÊNCIAS

AKESSON, K. et al. **Supremica an integrated environment for verification, synthesis and simulation of discrete event systems.** In: IEEE. Discrete Event Systems, 2006 8th International Workshop on. [S.l.], 2006. p. 384–385.

CASSANDRAS, C. G.; LAFORTUNE, S. I**ntroduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2008.

CURY, J. E. R. et al. **Supervisory control of discrete event systems with distinguishers**. Automatica, Elsevier, v. 56, p. 93–104, 2015.

RAMADGE, P.J.; WONHAM, W.M. **Supervisory control of a class of discrete event pro cesses**. In: Analysis and Optimization of Systems. [S.l.]: Springer, 1984. p. 475– 498.

AGRADECIMENTOS

Agradeço ao CNPq pelo apoio financeiro, que me ajudou muito durante a realização deste trabalho.