IMPLEMENTAÇÃO DE SISTEMAS DE CONTROLE SUPERVISÓRIO BASEADOS EM MÁQUINA DE ESTADOS

MOACYR C. POSSAN JR. [(1)], ANDRÉ B. LEAL [(2)*]

[(1)] *Whirlpool Latin America, Rua Dona Francisca, 7200, Distrito Industrial, 89219-600, Joinville, SC, Brasil*

[(2)] *Grupo de Pesquisa em Automação de Sistemas e Robótica, Dpto. de Eng. Elétrica, Universidade do Estado de Santa Catarina – UDESC, Campus Universitário Prof. Avelino Marcante s/n, 89223-100, Joinville, SC, Brasil*

E-mails: moacyr_c_possan@whirlpool.com, leal@joinville.udesc.br

Abstract — This paper presents a new methodology for the implementation of control systems based on the Supervisory Control Theory (SCT). It consists on a technique where an algorithm is proposed to obtain a Mealy finite state machine from an automaton of the supervisor obtained by the Supervisory Control Theory. This machine may be simplified further in order to have a reduced number of state transitions. This machine is a template for the implementation in Programmable Logic Controller (PLC) using Ladder language. A manufacturing system is presented to exemplify such methodology. Besides, another contribution of this work refers to the possibility of treating more than one event in the same scan cycle of the Programmable Logic Controller.

Keywords — Discrete Event Systems, Supervisory Control Theory, Manufacturing Systems, Modular Local Theory, State Machines, Programmable Logic Controllers, Ladder Language.

Resumo — Este trabalho apresenta uma nova metodologia para implementação de sistemas de controle baseados na Teoria de Controle Supervisório (TCS). Consiste em uma técnica onde um algoritmo é proposto para, a partir do autômato do supervisor obtido pela Teoria de Controle Supervisório, construir uma máquina de estados de Mealy. Esta máquina pode ser simplificada em seguida para reduzir o número de transições entre estados. Essa máquina serve como base para implementação em Controlador Lógico Programável (CLP) usando linguagem *Ladder*. Um sistema de manufatura é apresentado para exemplificar essa metodologia. Além disso, outra contribuição deste trabalho refere-se à possibilidade de tratamento de mais de um evento dentro do mesmo ciclo de execução do Controlador Lógico Programável.

Palavras-chave— Sistemas a Eventos Discretos, Teoria de Controle Supervisório, Sistemas de Manufatura, Máquinas de Estado, Controladores Lógicos Programáveis, Linguagem *Ladder.*

1. Introdução

Nos últimos anos, a indústria tem se defrontado cada vez mais com a necessidade de uso de métodos eficazes para o controle de Sistemas a Eventos Discretos (SEDs), dentre os quais se inclui grande parte dos sistemas de manufatura. Um SED é um sistema de estados discretos dirigido a eventos, isto é, sua evolução de estado depende da ocorrência de eventos discretos assíncronos no tempo (Cassandras e Lafortune, 1999).

A Teoria de Controle Supervisório (TCS) de Ramadge e Wonham (1989) é uma metodologia bastante conhecida e apropriada para a síntese de controladores para SEDs. Entretanto, apesar de sua reconhecida importância no meio acadêmico, a TCS não é difundida no âmbito industrial, de forma que, em geral, a resolução de problemas de controle supervisório nas indústrias é feita sem a utilização de um procedimento formal. Na indústria, os projetos são geralmente baseados no conhecimento do projetista, ficando atrelados à sua experiência em programação. Segundo Fabian e Hellgren (1998), um motivo importante para este afastamento entre teoria e prática consiste nas dificuldades encontradas na implementação da solução obtida por intermédio da TCS.

Neste trabalho, apresenta-se um algoritmo para transformar o autômato do supervisor obtido por intermédio da TCS em uma máquina de estados finita. Além disso, no intuito de minimizar as dificuldades normalmente encontradas na etapa de implementação da estrutura de controle, é proposta uma metodologia de implementação de controle supervisório para CLPs que se baseia nesta máquina de estados finita. Uma vantagem desta metodologia se comparada com outras encontradas na literatura, como a proposta em (Queiroz e Cury, 2002), consiste na possibilidade de tratamento de diversos eventos em um mesmo ciclo de execução do CLP.

Este trabalho está dividido da seguinte forma: na seção 2 apresenta-se uma visão geral da metodologia proposta e a seção 3 apresenta o algoritmo proposto para converter o autômato do supervisor em uma máquina de estados; a seção 4 mostra um exemplo motivador para ilustrar a metodologia proposta, a seção 5 descreve como simplificar esta máquina de estados, enquanto a seção 6 ilustra como implementar a máquina de estados em linguagem *Ladder* para CLP. Para finalizar, a seção 7 trata das conclusões.

2. Metodologia de Projeto Proposta

A figura 1 ilustra uma visão geral da metodologia

* **Tutor do Programa de Educação Tutorial – Grupo PET Engenharia Elétrica da UDESC**

proposta. Primeiro é realizada a síntese de um supervisor monolítico baseado na TCS. O autômato deste supervisor serve como parâmetro de entrada para o algoritmo de conversão para obter a máquina de estados. A máquina é então simplificada para reduzir o número de transições. A máquina simplificada, por sua vez, representa um modelo para gerar o código para um controlador (CLP, microcontrolador ou alguma outra unidade de processamento).

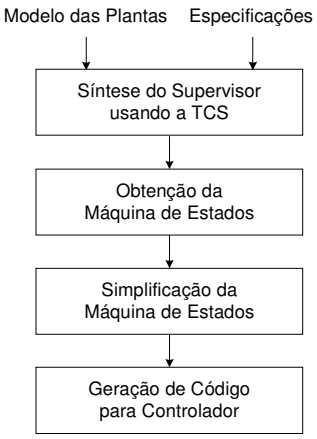

Figura 1. Metodologia Proposta

3. Algoritmo Para Gerar a Máquina de Estados

A máquina de estados obtida com o algoritmo proposto consiste em uma máquina de Mealy (Mealy, 1955). Nesta topologia, uma transição pode ter uma ou mais ações de saída e qualquer ação de saída pode ser usada em mais do que uma transição. As ações de saída estão associadas com as transições e não com os estados, que são passivos. Sendo assim, as ações podem ser associadas com mais do que um estado.

Um exemplo simples de uma máquina de Mealy de dois estados é mostrado na figura 2. Partindo do estado 1, a transição 1 faz a máquina ir desse estado para o estado 2 e toma a ação 1; partindo do estado 2, a transição 1 mais a transição 2 faz a máquina ir do estado 2 para o estado 1 e a ação 2 é tomada.

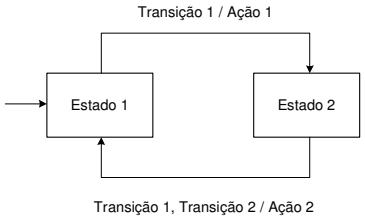

Figura 2. Exemplo de máquina de Mealy

O algoritmo proposto para obter a máquina de estados finita funciona de forma iterativa, varrendo os dados de entrada para obter os estados, transições e ações que compõem esta máquina. A máquina estará completa quando todos os dados de entrada forem processados. O algoritmo é mostrado na figura 3.

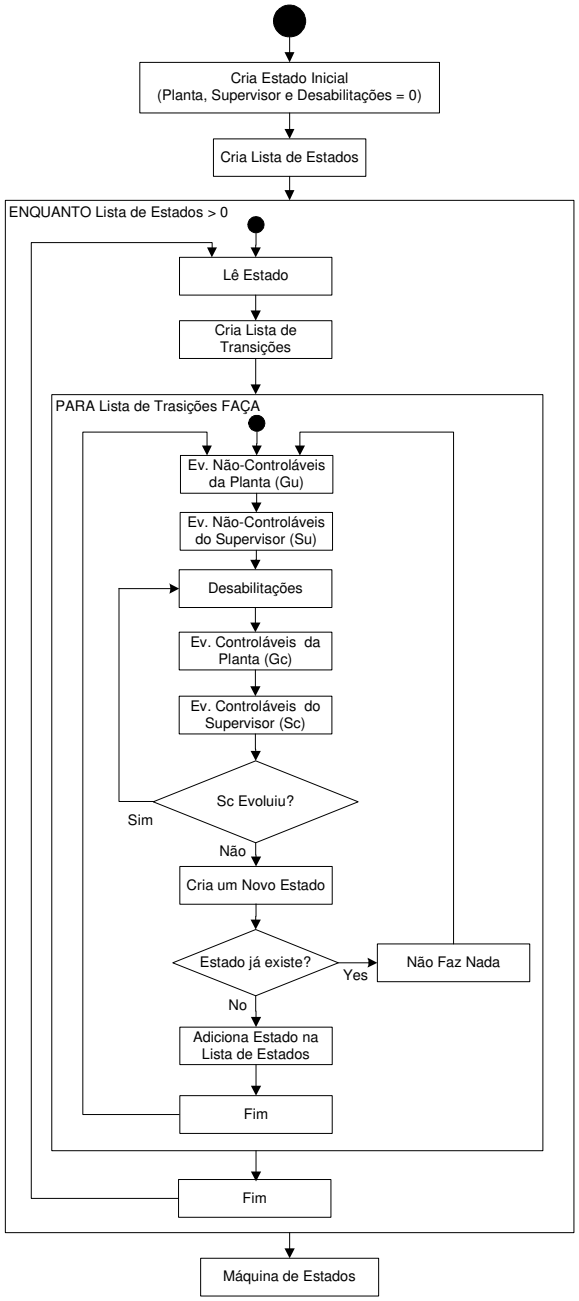

Figura 3. Algoritmo para Obtenção da Máquina de Estados

Os dados de entrada para o algoritmo são os autômatos da planta e do supervisor e as desabilitações, enquanto os dados de saída são os estados, as transições e as ações que compõem a máquina de estados.

No processo de transformação do autômato do supervisor para a máquina de Mealy os eventos não-controláveis na TCS equivalem às transições entre os estados, enquanto os eventos controláveis equivalem às ações de saída na máquina de Mealy. A transição entre dois estados na máquina de Mealy pode ocorrer por meio de um ou mais eventos não-controláveis no autômato. Para cada transição, uma ação de saída pode ser gerada ou não.

Nesse algoritmo, a inicialização considera os estados iniciais dos dados de entrada. Essa informação compõe o ponto de partida da máquina, representan-

do a condição onde a operação do processo físico ainda não começou.

O próximo passo é criar uma lista de estados. A lista é necessária para armazenar os estados que vão sendo obtidos iterativamente para serem tratados assim que o tratamento do estado atual é finalizado. A lista consiste em uma estrutura *FIFO* (*First In, First Out*). Enquanto ainda existirem estados na lista, haverá um processo iterativo para o seu tratamento. Um *loop While* é utilizado para esta finalidade.

Para cada estado, uma lista de transições é criada com o intuito de tratar todas as transições válidas para aquele estado específico. Para cada transição saindo de um determinado estado, será feito um processamento iterativo até que todas as transições sejam tratadas para todos os estados. Um *loop For* é utilizado para isto.

A fim de criar a lista de transições válidas para cada estado da máquina, é necessário primeiro dividir a planta e o supervisor em duas partes, de acordo com a controlabilidade dos eventos referentes às suas transições de estado. Define-se por *Gu* a parte da planta cujas transições se dão devido a eventos não-controláveis e por *Gc* a parte da planta cujas transições se devem a eventos controláveis. De forma semelhante, *Su* é a parte do supervisor cujas transições se dão por eventos não controláveis e *Sc* a parte cujas transições são devido a eventos controláveis.

A lista de transições equivale aos eventos não-controláveis que criam evoluções de estado em *Gu* e *Su*. A combinação de mais do que um evento não-controlável também é considerada uma transição.

Após a parte devido a eventos não-controláveis ser tratada, a parte devido a eventos controláveis é processada. Após a ocorrência de uma transição, o estado resultante em *Su* é analisado para verificar quais eventos controláveis são desabilitados por esse estado. Enquanto os eventos desabilitados estão proibidos de ocorrer, os remanescentes podem dar origem às ações. As ações válidas para um dado estado são os eventos controláveis que não estão desabilitados no supervisor e que causam evolução de estados em *Gc* e em *Sc*.

Quando uma ação ocorre, o algoritmo verifica se *Sc* evoluiu. Em caso afirmativo, então os eventos desabilitados para o estado destino são analisados a fim de verificar se alguma outra ação pode ocorrer. Este passo assegura que todas as ações possíveis de ocorrer para uma mesma transição são processadas. Isso significa que mais do que uma ação pode ocorrer para a mesma transição.

Caso *Sc* não tenha evoluído, então um novo estado é criado. Este estado é comparado com os outros estados e, caso já exista, é descartado. Caso contrário, é adicionado à fila de estados para ser tratado posteriormente.

Para cada transição devido a eventos não-controláveis, *Gu* e *Su* obtidos a partir da TCS evoluem. O mesmo ocorre para a parte controlável na ocorrência de uma ação. Esta metodologia é consistente com a definição de máquina de Mealy, onde as saídas (ações) dependem do estado atual e das entradas válidas (transições).

Após o término do procedimento de criar uma transição e as ações correspondentes o algoritmo trata as demais transições presentes na fila de transições. Quando todas as transições na fila são tratadas, o algoritmo analisa o próximo estado disponível na fila de estados. Estes processos iterativos são efetuados até que a fila de estados se torne vazia. Isto significa que a máquina de estados finita está completa.

4. Um Exemplo Motivador

Com o intuito de demonstrar a metodologia proposta considera-se um sistema de manufatura composto por 3 máquinas e 2 *buffers* intermediários de capacidade um entre as máquinas, conforme ilustrado na figura 4. As máquinas são representadas por Mi, onde i = 1,2,3 e os *buffers* são representados por Bj, onde j = 1,2.

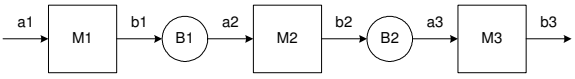

Figura 4. Sistema de Manufatura

A modelagem da planta e das especificações é feita por intermédio de autômatos de estados finitos, seguindo a TCS. Para modelagem da planta, são consideradas apenas as máquinas. Os *buffers* são levados em conta somente no modelo das especificações.

Os eventos controláveis que correspondem ao início de operação das máquinas são representados por *ax*, onde x = 1,2,3, enquanto os eventos não-controláveis que correspondem ao fim de operação das máquinas são representados por *bx*. O comportamento discreto das máquinas é modelado pelo autômato da figura 5.

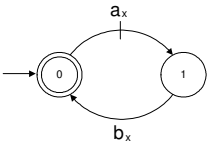

Figura 5. Modelo das Plantas

As especificações de controle para o sistema são restrições de coordenação para evitar *overflow* (sobrecarga) ou *underflow* (retirada de peças de um *buffer* vazio) nos *buffers*. Estas restrições expressam a idéia de que se devem alternar os eventos *bx* e *ax+1*, iniciando-se pelos eventos *bx*. O autômato que modela estas especificações é mostrado na figura 6.

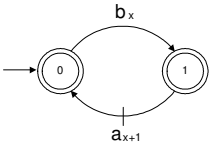

Figura 6. Especificações de Controle

O cálculo do supervisor minimamente restritivo e não bloqueante baseia-se num processo iterativo que identifica e elimina "maus estados" no autômato que modela a linguagem-alvo (Cury, 2001). O supervisor monolítico para este problema possui 18 estados e 32 transições e pode ser obtido com o auxílio da ferramenta Grail (Reiser *et al.*, 2006).

Utilizando o algoritmo proposto neste trabalho, obtêm-se, a partir do autômato do supervisor, dados para criar a máquina de estados ilustrada na figura 7, que é composta por 8 estados e 22 transições.

Os estados estão nomeados de acordo com as máquinas que estão ligadas em um determinado momento e os *buffers* que estão cheios. As transições são devido aos eventos não-controláveis e as ações, se houver, são separadas das transições por uma barra (/). Os eventos desabilitados são representados por traços. Observa-se que neste modelo as transições podem ocorrer devido a mais do que um evento não-controlável e as ações podem ser devido a mais do que um evento controlável.

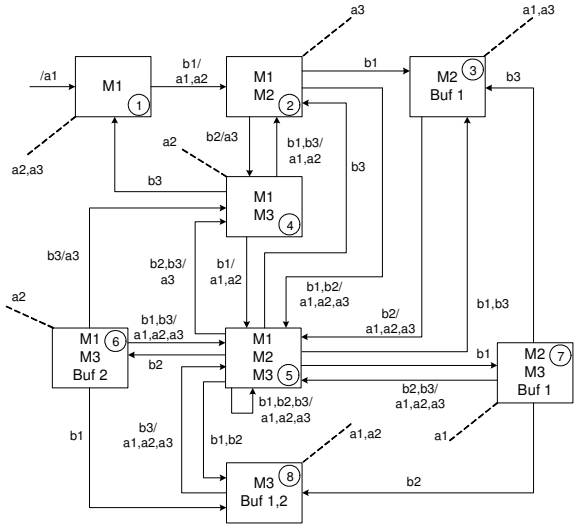

Figura 7. Máquina de Estados de Mealy

5. Simplificação da Máquina de Estados

A garantia de tratamento de mais do que um evento não-controlável dentro da mesma transição tem a desvantagem de resultar em um crescimento exponencial das transições, de acordo com o número de plantas presentes no modelo e quantas dessas podem ser habilitadas ao mesmo tempo. O número de transições criadas para um dado estado é da ordem de 2^n-1, onde n é o número de eventos não-controláveis válidos para aquele estado. Desta forma, para sistemas de grande porte, o tamanho de código para uma determinada implementação seria afetado significativamente para satisfazer essa condição. Uma solução alternativa para isso é considerar uma máquina de estados simplificada onde somente as transições devido a um único evento não-controlável são consideradas. Transições resultantes da combinação de mais do que um evento não-controlável são removidas do modelo.

A figura 8 mostra a máquina simplificada para o sistema de manufatura. Esta máquina ainda possui 8 estados, porém o número de transições foi reduzido para 14, se comparado com o modelo da figura 7.

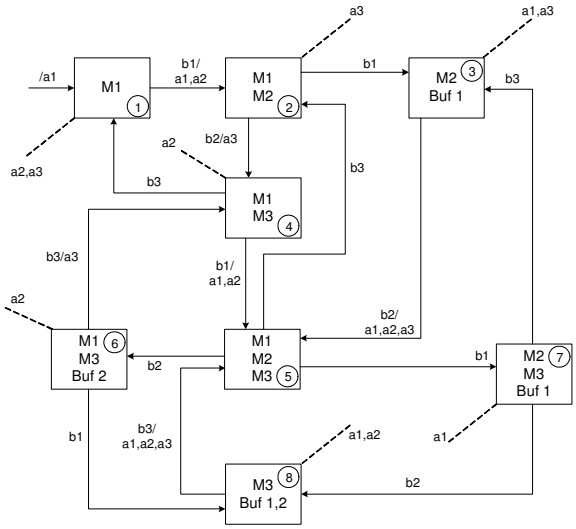

Figura 8. Máquina de Estados Simplificada

Esta solução garante economia de espaço em memória para uma implementação em CLP, por exemplo. Além disso, essa solução não restringe que mais do que um evento não-controlável seja tratado dentro do mesmo ciclo de execução. Se mais do que um evento não-controlável ocorrer, esses podem ser tratados ou não, dependendo da ordem em que os *rungs* são implementados em linguagem *Ladder*. Uma prática comum é nomear os estados da máquina em ordem crescente (1, 2, 3, e assim por diante) à medida que as transições são implementadas nos *rungs*. Caso um evento resulte em uma transição de um estado menor para um maior (estado 3 para o estado 12, por exemplo), esse evento será tratado; caso um evento resulte em uma transição de estado maior para um menor (estado 8 para o estado 5, por exemplo), o fluxo do programa permitirá que tal evento seja tratado somente no próximo ciclo de execução do CLP.

6. Geração de Código para CLPs

A TCS considera que todos os eventos são gerados espontaneamente pela planta e que o supervisor observa a cadeia de eventos gerados pela planta e atua na desabilitação de eventos controláveis de modo a evitar qualquer violação nas especificações de controle. A Figura 9-a ilustra o esquema de supervisão proposto por Ramadge e Wonham (RW). Entretanto, na maioria das aplicações práticas os eventos modelados como controláveis correspondem a comandos que, na verdade, devem ser gerados pelo elemento de controle e devem ser enviados para os atuadores, pois eles não ocorrem espontaneamente. A figura 9-b mostra a estrutura de controle normalmente empregada na prática (Malik, 2002).

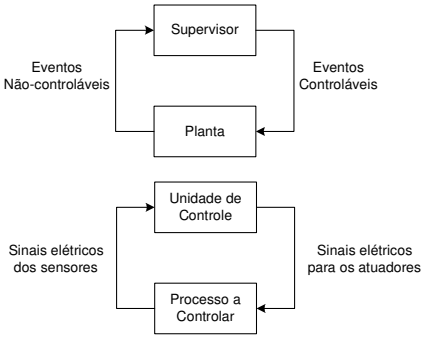

Figura 9. (a) Esquema de Supervisão RW (b) Esquema de Controle normalmente empregado na prática.

Com a transformação do autômato do supervisor em uma máquina de Mealy, obtém-se um modelo que mais se aproxima da forma do controlador apresentado por Malik (2002), isto é, da estrutura de controle mostrada na figura 9-b. Sendo assim, a geração da lógica de controle para o CLP fica mais intuitiva que aquela obtida a partir de autômatos.

A seguir, discute-se sobre alguns aspectos relativos à implementação da máquina de estados em CLP, cujo ciclo de execução do programa obedece ao seguinte funcionamento: leitura das entradas, execução da lógica de controle e escrita nas saídas.

Esta característica do CLP obriga que as saídas sejam atualizadas somente no fim do ciclo de execução. Devido a isso, a ativação dos atuadores requer um tratamento especial. Olhando para a estrutura da máquina de estados, pode acontecer que ao final de operação de um equipamento ele pode ser requisitado para um novo início de operação dentro do mesmo ciclo de execução. O CLP não reconhece o processo de fim/início de operação, mantendo sua saída física ativa o tempo inteiro ao longo do mesmo ciclo de execução. A fim de evitar isso, variáveis são necessárias para representar a evolução das plantas e garantir o sincronismo durante a dinâmica do sistema. Estas variáveis são chamadas *Planta i*, com *i* variando de 1 até *n* onde *n* é o número de plantas no sistema. Esta variável é *setada* toda vez que um equipamento termina sua operação. Este procedimento garante que o equipamento não seja requisitado para iniciar sua operação novamente dentro de um mesmo ciclo de execução. Ele será ligado somente no próximo ciclo de execução.

Assim como no modelo, as variáveis *bx* representam os eventos não-controláveis (transições) enquanto as variáveis *ax* (ações) representam os eventos controláveis na implementação.

O código *Ladder* pode ser dividido em cinco blocos: Inicialização, Entradas, Transições/Ações, Desabilitações e Saídas.

Inicialização: inicia a máquina de estados para o estado inicial e *seta* o evento controlável *a1* a fim de iniciar o processo, como mostrado na figura 10. Outras variáveis, tais como os demais eventos controláveis *ax*, os eventos não-controláveis *bx* e a evolução das plantas *Planta i*, são *resetadas*. As variáveis *resetadas* não são mostradas na figura 10.

Entradas: as variáveis de transição *bx* serão ativadas no controlador somente quando um *trigger* ocorrer nas entradas correspondentes. Assim, um detector de pulso é necessário para cada entrada a fim de capturar tal evento, como mostra a figura 11.

Transições/Ações: o requisito para uma transição ocorrer é a máquina estar em um dado estado e um evento não-controlável válido ocorrer. Caso estes requisitos sejam atendidos, um novo estado é *setado* e o estado anterior é *resetado*. O evento não-controlável é *resetado* para garantir que um evento não-controlável seja responsável por somente uma transição durante o ciclo de execução, evitando assim o risco do Efeito Avalanche (Fabian e Hellgren, 1998). Caso uma planta evolua, a variável correspondente *Planta i* é *setada* para evitar que a ação correspondente para aquela planta seja tomada durante o mesmo ciclo de execução, sendo tratada somente no próximo ciclo de execução. Isto é devido ao tratamento especial necessário para as ações, como descrito previamente nesta seção. As ações, se houver, são *setadas* para permitir que a planta correspondente seja ativada no mesmo ciclo de execução ou no próximo ciclo se for proibida de ocorrer no ciclo corrente. A figura 12 mostra a parte inicial desse bloco para o sistema de manufatura.

Desabilitações: este bloco é responsável por desabilitar eventos controláveis em cada estado da máquina. Isso significa que se a máquina alcançar um dado estado, os eventos controláveis relativos a este estado não serão permitidos de ocorrer. Isso é feito por intermédio de *resetar* as variáveis de ações *setadas* durante as transições. A figura 13 mostra o bloco de desabilitações para o exemplo analisado.

Saídas: uma saída será permitida de ocorrer somente se a ação correspondente for tomada e sua variável correspondente *Planta i* não estiver *setada*, como mostrado na figura 14. Caso isso aconteça, a bobina *Qi* que representa a saída física do CLP será ativada. Ainda, no final do programa todas as variáveis *Planta i* serão *resetadas* a fim de retornar para uma condição inicial antes de um novo ciclo de execução.

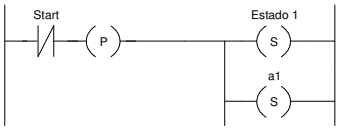

Figura 10. Bloco de Inicialização

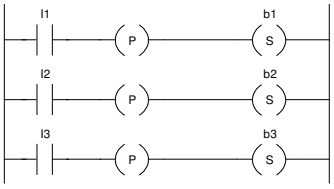

Figura 11. Bloco de Entradas

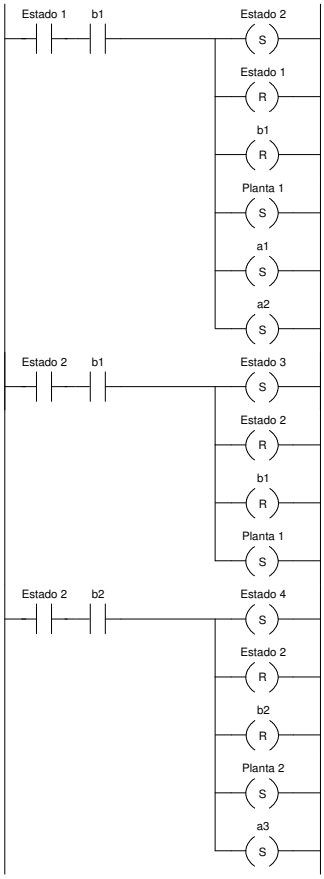

Figura 12. Parte Inicial do Bloco de Transições/Ações

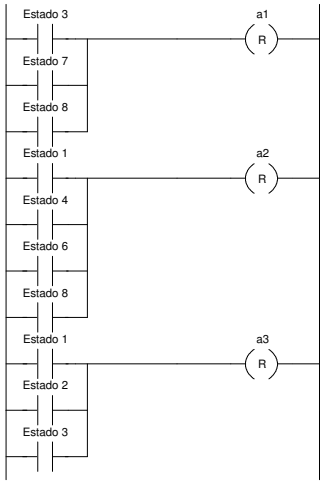

Figura 13. Bloco de Desabilitações

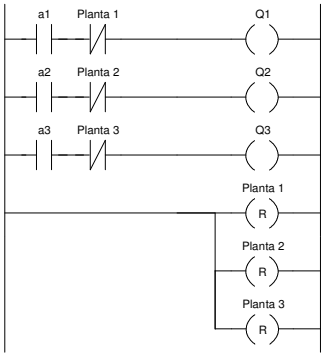

Figura 14. Bloco de Saídas

7. Conclusões

Este artigo apresenta um algoritmo para transformar o autômato de um supervisor obtido a partir da Teoria de Controle Supervisório (TCS) em uma máquina de estados finita, sua simplificação e conseqüente implementação em linguagem *Ladder* para Controladores Lógico Programáveis (CLPs). O código para efetuar o procedimento de controle é dividido em cinco blocos. Diversos eventos podem ser tratado no mesmo ciclo de execução do CLP dependendo da evolução das transições e das ações requeridas, respectivamente. Este trabalho serve como ponto de partida para o desenvolvimento de ferramentas automáticas para geração de código e de uma metodologia similar à apresentada aqui, mas baseada na Teoria de Controle Modular Local (Queiroz e Cury, 2002), abordagem que explora a natureza modular do sistema a ser controlado, criando supervisores específicos para partes específicas da planta modelada.

Agradecimentos

Os autores agradecem a FAPESC e ao CNPq pelo apoio financeiro (Contrato CON04504/2008-7) e à Whirlpool Latin America pelo incentivo à realização deste trabalho.

Referências Bibliográficas

Cassandras, C.G; Lafortune, S. (1999). *Introduction to Discrete Event Systems*. 2nd Ed. Massachussetts, Kluwer Academic Publishers.

Cury, J.E.R. (2001). Teoria de Controle Supervisório de Sistemas a Eventos Discretos. In: V *Simpósio Brasileiro de Automação Inteligente (SBAI)*.

Fabian, M. e Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete systems. In: *37th IEEE Conference on Decision and Control*, v. 3, pp. 3305-3310.

Malik, P. (2002). Generating controllers from discrete-event models. In: F. Cassez, C. Jard, F. Laroussinie, M. D. Ryan, *Proc. of the MOVEP*.

Mealy, George H. (1955). A Method to Synthesizing Sequential Circuits. *Bell Systems Technical Journal*. pp. 1045–1079.

Queiroz, M. H. e Cury J. E. R. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. *Proc. of 6th WODES*.

Ramadge, P. J.; Wonham, W. M. (1989). The control of discrete event systems. In: *Proceedings of IEEE, Special Issue on Discrete Event Dynamics Systems,* 77(1):81-98.

Reiser, C.; Da Cunha, A.E.C.; Cury, J.E.R. (2006). The Environment Grail for Supervisory Control of Discret Event Systems. In: *8th Intern. Workshop on Discrete Event Systems,* Michigan, USA, pp. 1-6.