UTILIZANDO EVENTOS IMINENTES NO REFINAMENTO DA SÍNTESE DO SUPERVISOR DE SISTEMAS A EVENTOS DISCRETOS TEMPORIZADOS

Eduard Montgomery Meira Costa e Antonio Marcus Nogueira Lima

Departamento de Engenharia Elétrica
Universidade Federal da Paraíba
58109-970 Campina Grande, PB, Brasil, Caixa Postal 10.105
Email: {eduard,amnlima}@dee.ufpb.br

Resumo— Esse trabalho trata do problema de refinamento do supervisor de um sistema a eventos discretos utilizando o conceito de eventos iminentes que é relacionado com uma definição alternativa para os tempos de vida de eventos. Para considerar a existência de eventos iminentes é utilizado um modelo temporizado para representar o sistema a eventos discretos. O refinamento proposto é efetuado no comportamento supervisionado por num supervisor sintetizado com o algoritmo da suprema sublinguagem que é executado para o modelo não temporizado. Esse refinamento é feito para melhorar a produtividade do sistema permitindo a existência de eventos paralelos. A utilização da técnica de refinamento proposta é demonstrada com um exemplo de um sistema de manufatura.

Abstract— This paper deals with the refinment of the discrete event supervisor based on the concept of imminent events that is related with an alternative definition of the lifetime of the events. To consider the imminent events a timed model is derived to represent the discrete event system. The supervisor is synthesized via the supremal sublanguage algorithm which is executed for the untimed model. This refinement is included to improve the throughput of the discrete event system and allows the execution of parallel tasks. The usefulness of the proposed technique is illustrated with an example for a manufacturing system.

Key Words— Imminent Events, Supervisory Control, Discrete Event Systems

1 INTRODUÇÃO

Um Sistema a Eventos Discretos (SEDs) (Ramadge e Wonham, 1982) apresenta uma evolução dinâmica descrita pela ocorrência de eventos que determinam sua interação com o ambiente e que alteram o seu estado interno. Os SEDs estão presentes em muitas aplicações do cotidiano, como redes de computadores, sistemas de manufatura e supervisão de tráfego. O estudo dos SEDs requer a utilização de uma representação adequada e que permita projetar um agente de controle automático, denominado de supervisor. A partir de tarefas especificadas para o sistema, o supervisor recebe informações dos eventos do sistema através de sensores, determina a ação de controle e envia comandos para os atuadores que inibem ou habilitam determinados eventos.

A Teoria de Controle Supervisório (TCS), foi desenvolvida em sua forma básica por Ramadge e Wonham (Ramadge e Wonham, 1987b; Ramadge e Wonham, 1987a; Ramadge e Wonham, 1989), e é baseada em linguagens formais e autômatos (Hopcroft e Ullman, 1979). Nesta teoria, faz-se uma distinção clara entre o sistema a ser controlado, denominado planta, e a entidade que o controla, o supervisor. A planta é um modelo que reflete o comportamento fisicamente possível do sistema, isto é, todas as ações que este é capaz de executar na ausência de ações de controle. O supervisor exerce uma ação de controle restritiva sobre a planta, de modo a confinar seu comportamento àquele que corresponde a uma dada especificação de controle.

No caso dos SEDs temporizados, utiliza-se o conceito de tempo de vida dos eventos. Este conceito é definido como o tempo decorrido desde a ocorrência de um evento, até uma nova ocorrência desse mesmo evento, levando-se em conta que nenhum outro evento tenha ocorrido durante este intervalo (Cao e Ho, 1990). A temporização dos eventos é definida em relação a um relógio global e, desse modo, a execução dos eventos fica diretamente relacionada aos ciclos desse relógio o que conseqüentemente determina seus tempos de vida.

A utilização do evento '*tick*' para representar a temporização foi proposta em (Brandin e Wonham, 1994). A inclusão do evento '*tick*' que é sincronizado com a descida (término de um ciclo) do relógio global, aumenta significativamente o número de transições do modelo e, conseqüentemente, o número de estados. O controle de SEDs temporizados também pode ser estudado utilizando a álgebra de dióides (Gaubert, 1992; Beachy, 1996), principalmente para os SEDs que apresentam periodicidade e necessitam de sincronização (Cohen et al., 1985; Gaubert, 1995; Gaubert, 1992; Baccelli et al., 1992). A álgebra de dióides também é utilizada na avaliação de desempenho de SEDs, utilizando-se das séries formais (Berstel e Reutenauer, 1988; Gaubert, 1994b; Gaubert, 1994a).

Em muitas aplicações, os sistemas exibem comportamentos que são completamente ou parcialmente seqüenciais. Isto é, determinados eventos só ocorrem quando há a completa finalização de uma tarefa. Entretanto, em alguns casos é necessário modificar as seqüências de execuções, para tornar o sistema mais seguro ou mais eficiente. A eficiência pode ser aumentada permitindo que haja paralelismo entre as partes de uma tarefa que podem ser executadas independentemente, sem alterar o comportamento global do sistema.

Esse trabalho utiliza uma definição de tempo de vida de eventos diferente da usual, para incluir o conceito de eventos iminentes. Nesse trabalho, o tempo de vida de um evento é o intervalo compreendido desde sua ocorrência até a ocorrência do próximo evento. Essa definição permite modelar o sistema por um autômato (max,+) (Gaubert, 1993; Gaubert, 1995). De acordo com essa nova definição, a existência de eventos não observáveis (Lin e Wonham, 1988; Cieslak et al., 1988; Özveren e Willsky, 1990; Lin e Wonham, 1990; Cao et al., 1997; Guan, 1997; Lawford et al., 1997) provoca

uma variação dos tempos de vida dos eventos observáveis. Definindo-se os eventos iminentes, utiliza-se um algoritmo que avalia e modifica a linguagem do supervisor. O supervisor é sintetizado através do algoritmo da suprema sublinguagem controlável que é executado para o modelo não temporizado. A modificação do supervisor, quando possível, reduz o tempo de execução das tarefas, pois substitui tarefas seqüenciais por tarefas executadas em paralelo. Essa modificação é feita de forma indireta pela modificação do comportamento supervisionado.

O artigo é organizado da seguinte forma: na seção 2 são formalizados os conceitos de tempo de vida de eventos aqui utilizado e introduz-se os eventos iminentes; na seção 3 é apresentado o algoritmo desenvolvido para a modificação da linguagem que define o comportamento do sistema sob supervisão (S/G) com um simplificado exemplo explicativo e, na seção 4 são apresentadas as conclusões deste trabalho.

2 CONCEITOS

Neste trabalho, o tempo de vida de um evento é definido como a seguir:

Definição 1 ***Tempo de vida*** *de um evento σ, denotado por $c(\sigma)$, é o intervalo de tempo compreendido desde o instante de tempo de sua ocorrência até o instante de tempo da ocorrência do próximo evento.*

Desta forma, o tempo de vida de um evento σ, isto é, $c(\sigma)$, é o intervalo decorrido desde o instante de tempo de sua ocorrência $t(\sigma,k)$, que leva o sistema a um estado $q(k)$, até o instante de tempo da ocorrência de um outro evento $\sigma^{'}$, $t(\sigma^{'},k+1)$, que muda o estado do sistema para um novo estado $q(k+1)$, em que k denota a ocorrência do k-ésimo evento. Isto é visto na Figura 1, onde o tempo de vida do evento α é denotado por $c(\alpha)$, determinado pela ocorrência de α no instante de tempo t_1, até a ocorrência do evento β, no instante de tempo t_2. Ou seja,

$$c(\alpha) = t_2 - t_1 = t(\beta, k+1) - t(\alpha, k). \qquad (1)$$

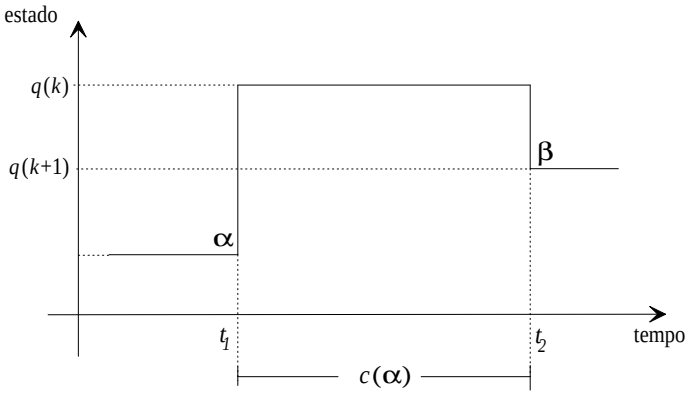

Figura 1. Tempo de vida do evento α.

A partir dessa definição do tempo de vida de um evento, introduz-se o conceito de evento iminente. Quando o tempo de vida dos eventos é considerado no estudo do comportamento de um SED, é necessário determinar os instantes de tempo nos quais o supervisor deve enviar comandos ao SED, para realizar a tarefa como desejada.

Definição 2 ***Tempo de iminência*** *de um evento σ, denotado por χ, é um intervalo de tempo anterior à ocorrência do próximo evento, a partir do qual não há mais possibilidade de bloqueá-lo. O tempo de iminência χ é definido como sendo uma fração $\mathcal{V} << 1$ do tempo de vida do evento corrente.*

O tempo de iminência, é determinado por

$$\chi(\sigma,k) = \mathcal{V}c(\sigma(k-1)) = t(\sigma,k-1) - t'(\sigma,k-1), \quad (2)$$

onde $t(\sigma,k-1)$ é o instante de tempo em que ocorreu o evento $\sigma(k-1)$ levando o sistema ao estado $q(k-1)$, e

$$t'(\sigma,k-1) = t(\sigma,k-1) - \mathcal{V}(t(\sigma,k) - t(\sigma,k-1)) \quad (3)$$

é o tempo que resta para a ocorrência do próximo evento $\sigma(k)$ que mudará o estado do sistema para $q(k)$. O tempo de iminência é visto na Figura 2, onde se pode observar que a ocorrência do evento $\sigma(k-1)$ se dá no instante de tempo $t(\sigma,k-1)$, que leva o sistema ao estado $q(k-1)$. O tempo de iminência é iniciado em $t'(\sigma,k-1) = c(\sigma,k-1) - \mathcal{V}(\sigma,k-1)$, e após $\chi(\sigma,k)$, ocorre o evento $\sigma(k)$ no instante de tempo $t(\sigma,k)$.

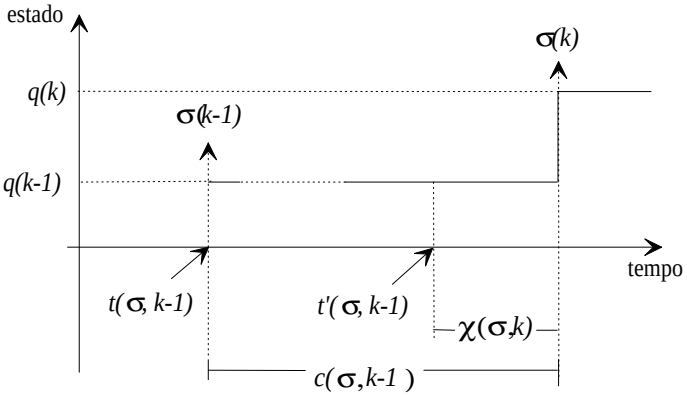

Figura 2. Exemplo do tempo de iminência para o evento $\sigma(k)$.

Definição 3 ***Evento Iminente*** *é o evento que deve ocorrer após o tempo de iminência, χ.*

Na Figura 3 é apresentado um exemplo de um evento iminente (evento β). Sua ocorrência levará o sistema do estado q para o estado q', ao término do tempo de vida do evento α. O evento β é um evento iminente a partir do instante de tempo $t'(\alpha)$, até o término do tempo de iminência $\chi(\beta)$, onde ele ocorre.

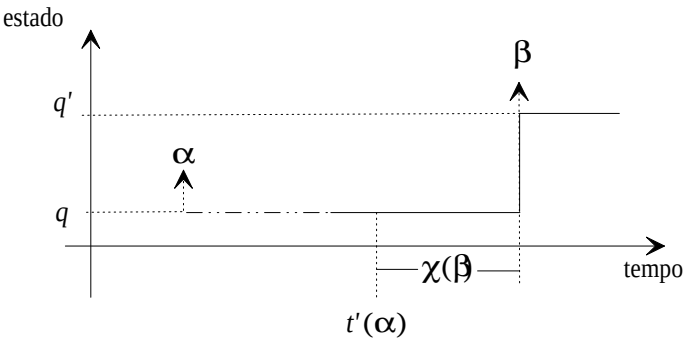

Figura 3. Exemplo de evento iminente.

Esta forma de temporização permite habilitar um evento desejado, antes do tempo de iminência para reduzir o tempo de execução da tarefa requerida.

Definição 4 ***Cobertura de um evento*** *σ é a execução de um outro evento σ' que reduz o tempo de vida do evento σ ao mínimo possível.*

3 A REFORMULAÇÃO DA LINGUAGEM

Uma palavra representa uma seqüência de eventos ou uma sub-tarefa realizada por um SED. De acordo com a definição do tempo de vida adotada nesse artigo, o tempo de execução de uma sub-tarefa é igual à soma dos tempos de vida dos eventos que a compõem. Dessa forma, pode-se utilizar os tempos de vida dos eventos e os eventos iminentes para definir uma nova linguagem para especificação de comportamento desejada. Esta nova linguagem é construída incluindo partes das palavras que representam subtarefas independentes e que tenham tempos de vida menores que o de um determinado evento para serem executadas paralelamente. Isso aumenta a eficiência do sistema, reduzindo o tempo de execução da tarefa.

Cada palavra da linguagem que especifica o comportamento, pode ser dividida em subpalavras, denominadas de sílabas, as quais representam partes independentes da tarefa, cuja execução é independente, além de não impedir a execução de uma outra sílaba. Dessa maneira, mais de uma sílaba pode ser executada paralelamente.

Definição 5 *Uma sílaba ρ é uma parte de uma palavra s que representa uma subtarefa independente do sistema.*

Por essa definição, uma palavra $s \in \Sigma^*$, é composta por várias sílabas,

$$s = \rho_1 \rho_2 ... \rho_n \tag{4}$$

onde ρ_1, $\rho_1\rho_2$,...,$\rho_1\rho_2...\rho_n$ são prefixos de s.

Cada evento $\sigma_i \in \Sigma$ que é parte da palavra s tem seu tempo de vida $c(\sigma_i)$. Desse modo, a palavra s tem um tempo de vida definido por

$$t_s = \sum_{i=1}^{|s|} c(\sigma_i). \tag{5}$$

Como cada sílaba ρ_i representa uma parte da palavra s, então ρ_i tem um tempo de vida t_{ρ_i}, com $t_{\rho_i} \leq t_s$. Dessa forma, o tempo de vida da palavra s também pode ser definido por

$$t_s = t_{\rho_1} + t_{\rho_2} + ... + t_{\rho_n} = \sum_{i=1}^{n} \rho_i, \tag{6}$$

com

$$t_{\rho_i} = \sum_{j=1}^{|\rho_i|} c(\sigma_j). \tag{7}$$

Dada uma especificação de comportamento não temporizada, utilizando o algoritmo da suprema sublinguagem controlável constrói-se um supervisor para o SED temporizado. Para a realização de tarefas paralelas, o modelo de S/G é modificado, acarretando uma aparente redução nos tempos de vida dos eventos do sistema. Essa modificação começa pela avaliação da possibilidade de incluir uma outra sílaba ρ' para ser executada paralelamente a uma sílaba ρ. Essa inclusão é feita para reduzir o tempo de vida de um evento de ρ, pela ocorrência do primeiro evento de ρ'. O último evento de ρ', deve cobrir o próximo evento de ρ.

Exemplo 1 *Considere que a palavra $s = \alpha\beta\gamma o\gamma\delta\varepsilon\eta$ seja composta pelas sílabas $\rho_1 = \alpha\beta$, $\rho_2 = \gamma o\gamma\delta$ e $\rho_3 = \varepsilon\eta$. Assim, na execução da tarefa representada pela sílaba $\rho_1 = \alpha\beta$, se o evento α tem um tempo de vida tal que seja possível incluir a sílaba $\rho_2 = \gamma o\gamma\delta$, e esta seja executada completamente antes do término do tempo de vida $c(\alpha)$, isto é,*

$$c(\alpha) > t_{\rho_2},$$

então, a partir da ocorrência do evento α pode-se executar a sílaba $\rho_2 = \gamma o\gamma\delta$ paralelamente ao evento α da sílaba ρ_1, sem que haja a necessidade da finalização da execução da tarefa representada pela sílaba $\rho_1 = \alpha\beta$. Dessa maneira, pode-se utilizar a nova palavra $s' = \alpha'\gamma o\gamma\delta\beta\varepsilon\eta$ em substituição à palavra original $s = \alpha\beta\gamma o\gamma\delta\varepsilon\eta$. Essa situação é ilustrada através do diagrama da Figura 4.

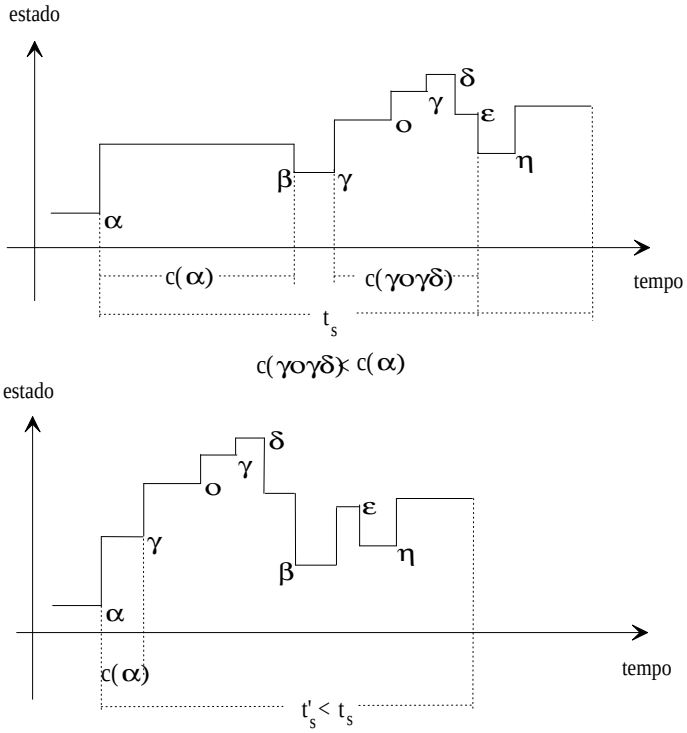

Figura 4. Exemplo de modificação do comportamento de um SED supervisionado.

Esse exemplo sugere que é possível construir um algoritmo para substituir uma palavra s por outra palavra s' que executa a mesma tarefa de modo a obter $t_{s'} < t_s$. Efetuando essa substituição em todas as palavras de uma linguagem, obtém-se uma nova linguagem que é executada em menos tempo.

3.1 ALGORITMO PARA A CONSTRUÇÃO DA NOVA LINGUAGEM DO SISTEMA SUPERVISIONADO

Para obter o supervisor de um SED não temporizado é necessário definir uma especificação de comportamento e executar o algoritmo da suprema sublinguagem controlável para sintetizá-lo. O novo sistema resultante da composição do SED com o supervisor exibe o comportamento definido na especificação.

O exemplo estudado anteriormente mostrou que é possível, incluir sílabas em palavras sem que isso altere a tarefa realizada e que isso permite reduzir o tempo de execução das tarefas. Generalizando essa idéia para todas as palavras da linguagem que define a composição S/G é possível

aumentar a produtividade do sistema. Esse processo de inclusão de sílabas gera uma nova linguagem para a execução da tarefa requerida para o SED supervisionado. Essa nova linguagem modifica a planta do sistema supervisionado. Isso é devido ao fato de que as inclusões de sílabas, as quais definem o paralelismo de execuções entre subtarefas, podem ser feitas antecipando ações de controle que já estavam incluídas no supervisor sintetizado para o modelo não temporizado.

Algoritmo 1 *Construção da nova linguagem*

1. *Ler uma palavra observada pertencente à linguagem do sistema supervisionado $L(S/G)$;*
2. *A cada sílaba ρ_i que determina uma subtarefa independente, calcular seu tempo de execução total t_{ρ_i}, através da soma dos tempos de vida de seus eventos, isto é, para*

$$s = \rho_1\rho_2...\rho_n, t_{\rho_i} = \sum_{j=1}^{k_i} c(\sigma_j), i = 1, ..., n$$

   com $\sigma_j, j = 1, ..., k_i$ sendo os k_i eventos que compõem a sílaba ρ_i;
3. *Avaliar cada evento da sílaba que inicia o processo e fazer:*

   i) *Para cada evento σ_k da sílaba ρ_i, verificar se existe alguma sílaba ρ_j que realiza uma subtarefa que não dependa do término da atividade do evento σ_k, e cujo tempo de execução satisfaça $t_{\rho_j} < c(\sigma_k)$, para ver se é possível incluí-la entre este evento e o próximo, isto é,*

$$\{\exists\sigma_k \subset \rho_i | c(\sigma_k) > t_{\rho_j} \wedge \rho_j \text{ não depende de } \sigma_k : c'(\sigma_k) = \mathcal{V}c(\sigma_k) \wedge \rho_i = \sigma_k\rho_j\sigma_{k+1}\};$$

   ii) *Caso nenhuma sílaba possa ser inserida, avaliar qual é a que melhor cobre o evento iminente (o próximo evento da sílaba), ou seja,*

$$\exists\sigma_k \subset \rho_i \wedge \forall t_{\rho_j} | c(\sigma_k) > t_{\rho_j} \wedge \rho_j \text{ não depende de } \sigma_k : c'(\sigma_k) = \mathcal{V}c(\sigma_k) \wedge \begin{cases} \rho_i = \sigma_k\rho_j \text{ se } \sigma_n \text{ cobre } \sigma_{k+1} \\ \rho_i = \sigma_k\sigma_l...\sigma_{n-1}\sigma_{k+1}\sigma_n \text{ se} \\ \qquad \sigma_n \text{ não cobre } \sigma_{k+1} \end{cases}$$

   com

$$\sigma_l, ..., \sigma_{n-1}, \sigma_n \subset \rho_j;$$

   iii) *Se há alguma sílaba que possa ser incluída, seguindo os passos 3.i e 3.ii, incluí-la após o tempo*

$$\mathcal{V}c(\sigma_i)$$

   de sua ocorrência;

   iv) *Avaliar a próxima sílaba e repetir o processo voltando ao passo 3.i;*
4. *Repetir o passo 3 com a nova palavra enquanto houver possibilidades de inclusões;*
5. *Repetir o procedimento com cada palavra da linguagem;*
6. *Definir a nova linguagem.*

Tabela 1. Eventos do sistema, descrições dos eventos e respectivos tempos de vida

σ	descrição	$c(\sigma)$
α	*br* leva peça 1 para *m2*	$2s$
β	*br* leva peça 2 para *m1*	$2s$
γ	*br* leva peça 3 para *m1*	$2s$
δ	*br* leva peça 3 de *m1* para *m3*	$1s$
η	*br* leva peça de *m1* para *b*	$2s$
μ	*br* leva peça de *m2* para *b*	$2s$
ν	*br* leva peça de *m3* para *b*	$2s$
ξ	*br* leva peça 1 de *b* para *m4*	$1s$
ε	*br* leva peça 2 de *b* para *m4*	$1s$
θ	*br* leva peça 3 de *b* para *m4*	$1s$
φ	processamento da *m1*	$3s$
ϕ	processamento da *m2*	$5s$
ψ	processamento da *m3*	$2s$
λ	processamento da *m4*	$1s$

3.2 EXEMPLO: SÍNTESE DE UM SUPERVISOR PARA UM SISTEMA DE MANUFATURA

O sistema de manufatura que é apresentado na Figura 5 é composto por um braço robótico (*br*), três lugares com os diferentes tipos de matérias primas, uma máquina furadeira (*m1*), uma máquina que usina roscas externas para fazer parafusos (*m2*), uma máquina que usina roscas internas para fazer porcas (*m3*), um buffer (*b*) para colocar as peças processadas e uma estação de montagem e fixação (*m4*), que junta duas chapas de aço perfuradas usando um parafuso e uma porca.

Nesse sistema, *br* sempre volta à sua posição inicial (centro), ao executar qualquer atividade. A Tabela 1 apresenta os eventos desse sistema, suas descrições e seus tempos de vida. De fato, para determinar os tempos de vida apresentados nessa tabela é necessário compor o SED temporizado com o supervisor sintetizado para o modelo não temporizado. Observando o comportamento do sistema supervisionado e utilizando o algoritmo da inversão-d (Park, 1996) calculam-se os tempos de vida dos eventos e, em seguida redefine-se a nova linguagem através do algoritmo proposto.

Para usinar a rosca interna para fazer uma porca em *m3*, é necessário primeiro perfurar a matéria prima *2* em *m1*. As chapas de aço também devem ser perfuradas em *m1*. Para fazer um parafuso deve-se usinar uma rosca externa da matéria prima *1* em *m2*. Para efetuar a fixação das peças processadas, utiliza-se a máquina *m4*.

A especificação de comportamento é que o sistema produza dez produtos finais. Um produto final pronto é composto de duas chapas de aço perfuradas e juntas por um parafuso e uma porca. Considere que o sistema realiza tarefas seqüencialmente e que a linguagem do sistema supervisionado é dada por

$$L(S/G) = (\alpha\phi\eta\gamma\varphi\delta\psi\nu\beta\varphi\mu\beta\varphi\mu\xi\varepsilon\theta\lambda)^{10}$$

que é uma seqüência de eventos determinando que *br* deve pegar a *peça 1* e colocar em *m2*, esperar seu término e em seguida colocá-la em *b*; pegar *peça 3* e colocar em *m1*, esperar seu término e em seguida colocá-la em *m3*, esperar seu término e em seguida colocá-la em *b*; repetir 2 vezes com a *peça 2* em *m1* colocando-as prontas em *b* e daí, pegar as peças processadas em *b* e colocá-las em *m4* para terminar

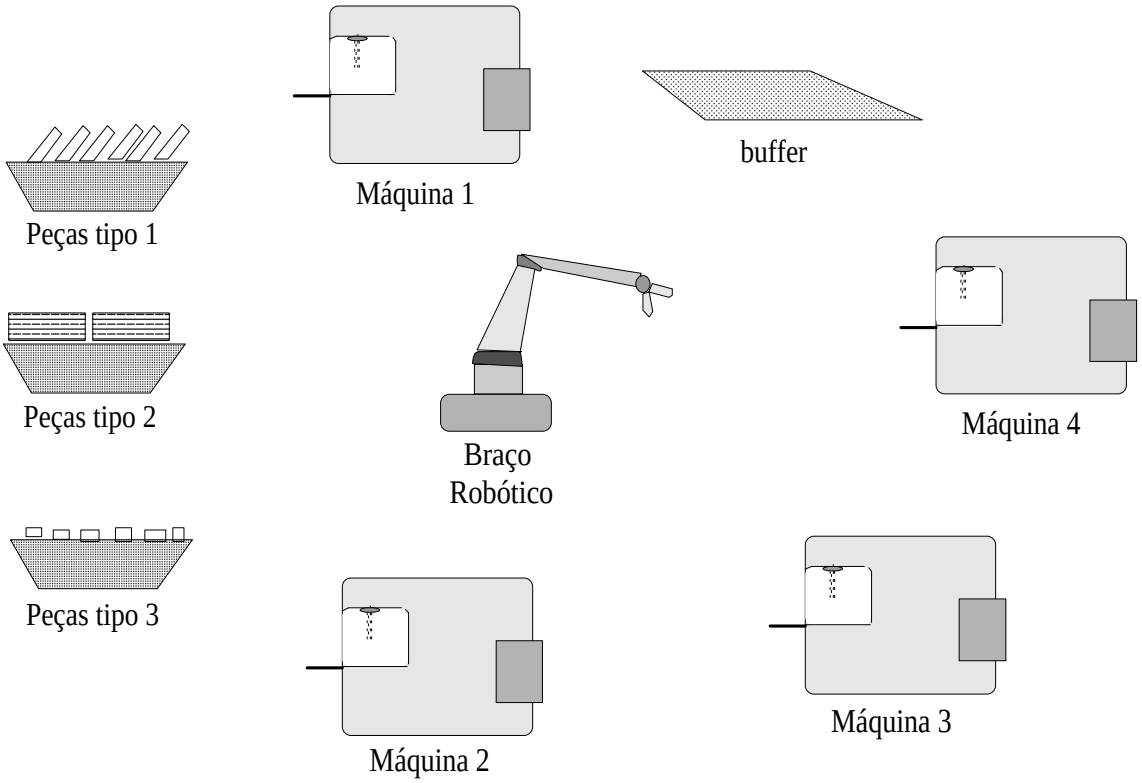

Figura 5. Sistema de manufatura utilizado para demonstrar a utilização do algoritmo de construção da nova linguagem do comportamento do sistema supervisionado.

cada produto final. Essa seqüência deve ser repetida dez vezes.

Considerando os tempos de vida apresentados na Tabela 1, o tempo de execução de um produto final, que é dado pela soma dos tempos de vida de cada evento é

$$t_s = 38s,$$

desde que é uma tarefa seqüencial (só repete o processo após o término de um item). Consequentemente, o tempo de processamento dos dez produtos é

$$10t_s = 380s.$$

Observando o comportamento do sistema verifica-se que há situações de ociosidade para o braço robótico. De fato, *braço robótico só realiza uma tarefa quando uma peça é terminada e colocada no buffer*. Desse modo, analisa-se as sílabas da linguagem $L\left(S/G\right)$, que são

$$\alpha\phi, \eta, \gamma\varphi, \delta\psi, \nu, \beta\varphi, \mu, \xi, \varepsilon, \theta \text{ e } \lambda$$

e cujos respectivos tempos de vida são

$$7, 2, 5, 3, 2, 5, 2, 1, 1, 1 \text{ e } 1s.$$

Na primeira sílaba, tem-se

$$c(\alpha) = 2s$$

e

$$c(\phi) = 3s.$$

Quando *m2* é ativada, o tempo de vida do processamento ($c(\varphi) = 5s$) pode ser reduzido pela inclusão de uma outra atividade (outra sílaba), para que o sistema trabalhe em paralelo. A primeira sílaba independente que se pode colocar para reduzir o tempo de vida do evento ϕ é $\gamma\varphi$, que será iniciada após *br* colocar a peça em *m2*. Assim, encontra-se

$$\rho^{'} = \alpha\phi'\gamma\varphi\eta.$$

Nesse exemplo utiliza-se $\mathcal{V} = 0.01$ para definir $\chi\left(\sigma, k\right)$ desde que essa escolha corresponde a 1% do tempo de vida do evento corrente e para fins práticos, não há mais possibilidade de que uma ação de controle impeça a ocorrência do próximo evento. Como a sílaba $\gamma\varphi$ é colocada com o tempo de 1% do tempo de vida do evento φ, isto é, $0,05s$, o tempo desta nova sílaba é, então,

$$t_{\rho'} = 2 + 0,05 + 2 + 3 + 2 = 9,05s.$$

Repetindo este procedimento para cada sílaba, encontra-se a palavra

$$s = \alpha\phi'\gamma\varphi\eta\delta\psi'\beta\varphi'\nu\xi\beta\varphi'\varepsilon\theta\mu\varepsilon\lambda'$$

onde λ' tem o tempo de execução reduzido em um por cento do tempo de vida de λ, isto é, $0,01s$, desde que já se pode reiniciar a tarefa a ser realizada pela palavra reformulada. Para a linguagem final, tem-se:

$$L'\left(S/G\right) = s^9\alpha\phi'\gamma\varphi\eta\delta\psi'\beta\varphi'\nu\xi\beta\varphi'\varepsilon\theta\mu\varepsilon\lambda.$$

Deve-se observar que, na última execução do sistema, o evento λ tem seu tempo de vida completo, isto é, $c\left(\lambda\right) = 1s$. Assim, vê-se que a nova palavra s tem o tempo de execução de $22,11$ segundos. A nova linguagem do sistema supervisionado tem um tempo de execução:

$$t_{L'(S/G)} = 9 \times t_s + \left(t_s - c\left(\lambda'\right) + c\left(\lambda\right)\right) = 222,09s.$$

Isso implica numa redução de

$$\left(380 - 222,09\right)/380 = 41,56\%$$

no tempo de realização da tarefa especificada.

4 CONCLUSÃO

Este trabalho mostra que é possível construir um supervisor para um SED temporizado que executa uma especificação de comportamento com mais eficácia. A lógica para a construção do supervisor é a mesma utilizada em (Ramadge e Wonham, 1987a; Ramadge e Wonham, 1989). Com o supervisor sintetizado e, através da definição dada dos tempos de vida de eventos e dos eventos iminentes, o algoritmo proposto refina a linguagem do sistema supervisionado e gera uma nova linguagem que inclui a paralelização de atividades independentes, ou sílabas, que realiza o comportamento especificado em menos tempo.

De fato, a solução proposta nesse artigo, consiste no re-escalonamento determinístico das sub-tarefas independentes do SED. O re-escalonamento é feito a partir da análise do comportamento do SED temporizado operando com o supervisor sintetizado para o modelo não temporizado. Nesse re-escalonamento consideram-se os tempos de vida dos eventos para reduzir a ociosidade dos recursos.

Referências

Baccelli, F., Cohen, G., Olsder, G. e Quadrat, J. (1992). *Synchronisation and Linearity. An Algebra for Discrete Event Systems*, John Wiley Sons.

Beachy, J. (1996). *Abstract Algebra II*, Waveland Press, Inc.

Berstel, J. e Reutenauer, C. (1988). *Rational Series and their Languages*, Springer.

Brandin, B. e Wonham, W. (1994). Supervisory control of timed discrete-event systems, *IEEE Transactions on Automatic Control* **39**(2): 329–342.

Cao, C., Lin, F. e Lin, Z. (1997). Why event observation: Observability revisited, *Discrete Dynamic Systems. Theory and Applications* (7): 128–149.

Cao, X. e Ho, Y. (1990). Models of discrete event dynamic systems, *IEEE Control System Magazine* **10**(4): 69–76.

Cieslak, R., Desclaux, C., Fawaz, A. e Varaiya, P. (1988). Supervisory control of discrete-event processes with partial observations, *IEEE Transactions on Automatic Control* **33**(3): 249–260.

Cohen, G., Dubois, D., Quadrat, J. e Viot, M. (1985). A linear system theoretic view of discrete event process and its use for performance evaluation in manufacturing, *IEEE Transactions on Automatic Control* **30**(3): 210–220.

Gaubert, S. (1992). *Théorie des Systèmes Linéaires dans les Dioïdes*, PhD thesis, École Nationale Supérieure des Mines de Paris.

Gaubert, S. (1993). Performance evaluation of timed automata, *Technical report*, Institut National de Recherche en Informatique et en Automatique.

Gaubert, S. (1994a). On rational series in one variable over certain dioids, *Technical report*, Institut National de Recherche en Informatique et en Automatique.

Gaubert, S. (1994b). Rational series over dioids and discrete event systems, *Proc. of the 11th Int. Conf. on Analysis and Optimization of Systems: Discrete Event Systems, Sophia Antipolis, Lectures Notes in Control and Information Sciences 199, Springer* .

Gaubert, S. (1995). Performance evaluation of (max,+) automata, *IEEE Transactions on Automatic Control* **40**(12): 2014–2025.

Guan, Y. (1997). *Implementation of hierarchical observer theory*, Master's thesis, University of Toronto.

Hopcroft, J. e Ullman, J. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, USA.

Lawford, M., Wonham, W. e Ostroff, J. (1997). State-event observers for labeled transition systems, *Technical report*, University of Toronto, USA.

Lin, F. e Wonham, W. (1988). On observability of discrete event systems, *Information Sciences* pp. 173–198.

Lin, F. e Wonham, W. (1990). Decentralized control and coordination of discrete-event systems with partial observation, *IEEE Transactions on Automatic Control* **35**(12): 1330–1337.

Özveren, C. e Willsky, A. (1990). Observability of discrete event systems, *IEEE Transactions on Automatic Control* **35**(7): 797–806.

Park, Y. (1996). *Model-Based Monitoring of Discrete Event Systems*, PhD thesis, Purdue University.

Ramadge, P. e Wonham, W. (1982). Supervision of discrete event processes, *Procedings of 21st Conference on Decision and Control* pp. 1228–1229.

Ramadge, P. e Wonham, W. (1987a). On the supremal controllable sublanguage of a given language, *SIAM Journal of Control and Optimization* **25**(3): 637–659.

Ramadge, P. e Wonham, W. (1987b). Supervisory control of a class of discrete event processes, *SIAM Journal of Control and Optimization* **25**(1): 206–230.

Ramadge, P. e Wonham, W. (1989). The control of discrete event systems, *Proceedings of the IEEE* **77**(1): 81–98.