UNIVERSIDADE FEDERAL DE OURO PRETO

ESCOLA DE MINAS

COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU

FERNANDO BOLIVAR CRUZ CASTRO

MODELAGEM E CONTROLE DE UM SISTEMA FLEXÍVEL DE MANUFATURA UTILIZANDO LINGUAGENS FORMAIS E AUTÔMATOS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2014

FERNANDO BOLIVAR CRUZ CASTRO

MODELAGEM E CONTROLE DE UM SISTEMA FLEXÍVEL DE MANUFATURA UTILIZANDO LINGUAGENS FORMAIS E AUTOMATOS

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientadora: Regiane de Sousa e Silva Ramalho

Co-orientador: Danny Augusto Vieira Tonidandel

Ouro Preto
Escola de Minas – UFOP
JULHO / 2014

RESUMO

Propõe-se uma nova plataforma para estudo de sistemas a eventos discretos e teoria de controle supervisório, que simula um sistema flexível de manufatura. Para isto, modela-se um sistema que fabrica seis tipos de peças utilizando linguagens formais e autômatos. Em seguida, um sistema de controle (supervisor) é projetado utilizando a abordagem "controle modular local". Por fim, desenvolve-se uma plataforma de simulação da planta, a fim de observar seu comportamento e servir como base para a realização de trabalhos futuros.

Palavras-chave: autômatos e linguagem formais, sistemas a eventos discretos, supervisor, teoria de controle supervisório.

ABSTRACT

This work intends to propose a new platform for the study of discrete event systems and supervisory control theory, which simulates a flexible manufacturing system. For this, it models a system that manufactures six types of pieces using formal languages and automata. Then, a control system (supervisor) is designed using the "local modular control" approach. Finally, it develops a simulation platform plant, in order to observe their behavior and serve as a basis for future work.

Keywords: automata and formal language, discrete event systems, supervisor, supervisory control theory.

AGRADECIMENTOS

Gostaria de agradecer à minha família, especialmente aos meus pais Geraldino e Isaura, por todo apoio, o qual permitiu concluir mais esta etapa na minha vida, sempre próximos quando precisei. À minha Irmã, Gislaine, pelo grande auxílio na resolução dos problemas que surgiram durante a minha faculdade. A minha Vó, Ermatina, por todas as suas orações na intenção de alcançar meus objetivos. Agradeço também aos meus amigos da faculdade e por sempre poder contar com eles.

Sou grato a toda UFOP pela estrutura oferecida e pelo seu corpo docente qualificado, que me proporcionou todo o suporte para o aprendizado. Em especial ao professor Agnaldo, pela boa vontade em me ajudar na busca de estágio e trainees, aos professores Regiane e Danny, pelo auxílio e oportunidade para realizar este trabalho.

Por fim, gostaria de agradecer ao CPNQ pela oportunidade de intercâmbio, que me proporcionou uma grande aprendizagem, tanto do ponto de vista de conhecimento, quanto no meu amadurecimento como pessoa.

LISTA DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SIGLAS

ADEF - Autômato determinístico de estado finito

ANDEF - Autômato não determinístico de estado finito

ER - Expressões regulares

Est 1 - Esteira 1

Est 2 - Esteira 2

Est 3 - Esteira 3

Est 4 - Esteira 4

Est 5 - Esteira 5

Est 6 - Esteira 6

Est 7 - Esteira 7

MA - Máquina A

MB - Máquina B

MC - Máquina C

RSP - Representação por sistema produto

RW - Ramadge e Wonham

SDVC - Sistema dinâmico de variáveis contínuas

SED - Sistemas a eventos discretos

TCS - Teoria de controle supervisório

TCT - Toy control theory

LISTA DE SÍMBOLOS

Σ - Conjunto de símbolos (eventos) ou alfabeto

Σ_c - Conjunto de eventos controláveis

Σ_{uc} - Conjunto de eventos não-controláveis

Σ^* - Conjunto de todas as cadeias finitas de elementos de Σ

ε - Cadeia vazia

L - Linguagem

G - Autômato

G_{ac} - Componente acessível

X- Conjunto finito de estados do autômato

X_{ac} - Conjunto de estado de G que são acessíveis

x_0 -Representa o estado inicial

X_m - Representa os estados finais, chamados de estados marcados

f - Função de transição de estados

$L(G)$ - Linguagem gerada por G

$L_m(G)$- Linguagem marcada por G

× - Composição por produto

|| - Composição paralela ou composição síncrona ou composição síncrona completa

Γ - Entrada de controle

A - Limite superior

E - Limite inferior/ especificação de segurança

S - Supervisor

$Sred$ - Supervisor reduzido

S/G - Sistema controlado

K - Linguagem desejada

$SupC$ - Supervisor que implementa a máxima sublinguagem controlável

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de automação são empregados em muitas áreas, como na indústria em geral, na robótica, em redes de computadores e em outros diversos contextos em que o custo ou a complexidade da tarefa a ser realizada justificam o seu uso. Estes sistemas, quando capazes de receberem estímulos do ambiente a sua volta, os eventos, são caracterizados como sistemas a eventos discretos, também chamados de SEDs (CURY, 2001).

A abordagem dos SEDs permite a representação das interações de subsistemas, que são as transições de estado. Enquanto a representação clássica faz o uso de equações diferenciais para descrever os sistemas dinâmicos de variáveis contínuas (SDVCs), os SEDs podem ser modelados com diversas ferramentas, tais como: Redes de Petri (MURATA, 1989; KROGH; HOLLOWAY, 1991; ZHOU; VENKATESH, 1999 apud QUEIROZ, 2004), Cadeias de Markov (ÇINLAIR, 1975; PUTERMAN, 1994 apud QUEIROZ, 2004), Teoria das Filas (KLEINROCK, 1975 apud QUEIROZ, 2004), Processos Semi-Markovianos Generalizados (CAO; HO, 1990 apud MONTOGOMERY, 2004), Álgebra de Processos (MILNER,1980 apud MONTOGOMERY, 2004), Álgebra de Dióides (GAUBERT, 1992; BOIMOND, 1999; L'HER, 1997; LIBEAUT, 1996 apud MONTOGOMERY, 2004) e Teoria de Linguagens Formais e Autômatos (GRILL, 1962; HOPCROFT, 1969; SALOMMA, 1973; PINNCHIANAT,1999 apud MONTGOMERY, 2004).

Segundo Cassandras e Lafortune (2008), os autômatos são formas mais básicas de representar modelos de SEDs, uma vez que são representados por uma linguagem fácil e intuitiva. Em alguns casos, estes podem se tornar grandes e complexos para estruturas com muitos estados. Apesar disso, o autômato é bastante recomendável para SEDs, já que o uso de autômatos agregados, linguagens formais e expressões regulares é uma poderosa ferramenta para modelar um sistema de automação aplicado a um controle supervisório posterior.

Na indústria de manufatura, o controle supervisório, quando bem estruturado, permite o controle de uma planta flexível. Segundo Monotgomery (2004), através de linguagens formais e autômatos é possível representar um supervisor como um autômato parte de um modelo de SED. Assim, com este autômato, é possível realizar as tarefas requeridas pelo sistema. Esta abordagem é conhecida como modelo R-W (RAMADGE; WONHAM, 1982 apud MONTGOMERY, 2004).

Ao se modelar e fazer o controle supervisório de uma planta flexível de manufatura utilizando linguagens formais e autômatos, é possível, através de simulações, obter resultados consistentes que permitam um controle eficiente e seguro do sistema. Estas simulações podem

ser feitas por meio de linguagens de alto nível, como Java, C#, e também por meio de linguagens específicas como GRAIL, TCT.

1.1 Objetivos

O objetivo deste presente trabalho é fazer a modelagem de um sistema flexível de manufatura utilizando linguagens formais e autômatos aplicados a um controle supervisório para sistemas a eventos discretos.

1.1.1 Objetivos específicos

- Estudo teórico de sistemas a eventos discretos;
- Modelagem de um sistema flexível de manufatura;
- Desenvolvimento de um controle supervisório consistente;
- Simulação e análise de resultados do sistema obtido.

1.2 Metodologia

No primeiro momento, faz-se um estudo sobre sistemas a eventos discretos e de sua representação utilizando linguagens formais e autômatos. Neste ponto, busca-se analisar exemplos e trabalhos modelados a partir destas linguagens.

Posteriormente, a planta é divida em subsistemas levando em consideração sua estrutura descentralizada, ou seja, seus componentes: as esteiras (superior, inferior, interna), as máquinas (em série, em paralelo) e robôs (da esteira, do estoque). Estes são modelados separadamente, obtendo-se a representação gráfica dos autômatos. Esta divisão é feita visando tratar um problema de grande complexidade, transformando-o em problemas de complexidade menores, o que facilita a visualização e a modelagem (CURY, 2001).

Definem-se, para o processo de modelagem, as restrições ou especificações de segurança às quais o sistema está sujeito. Dessa forma, evita-se comportamentos indesejáveis, como choque de ferramentas, tentativa de alocação de mais de uma peça em zonas unitárias, dentre outros problemas. Estas restrições são associadas aos subsistemas predeterminados, por composição paralela (CASSANDRAS; LAFORTUNE, 2008), também chamada de composição síncrona (MONTGOMERY, 2004).

Para cada novo subsistema (resultado da composição síncrona dos subsistemas com suas restrições), se obtém um autômato supervisor. Este é responsável pelas ações de controle

sobre a malha a qual está associado. O conjunto de supervisores resultantes são então reduzidos e representam a ação de controle para o sistema em estudo.

O processo de modelagem e composição síncrona é feito com auxílio da ferramenta de programação TCT (WONHAN, 2014). Esta ferramenta permite trabalhar com linguagens formais e autômatos além de efetuar operações nos mesmos. O resultado obtido deste procedimento será usado para implementação, em C#, de um simulador que permita visualizar todo o processo físico. Assim, será feita a análise e avalição do sistema, verificando se seu comportamento está dentro do esperado.

1.3 Justificativas

A flexibilidade na indústria de manufatura expressa um grande potencial a longo prazo e um diferencial competitivo como parte da estratégia de qualidade e custo de uma empresa (ZUKIN; DACOL, 2001). Com o uso de autômatos e linguagens formais, é possível modelar e controlar este tipo de sistema a eventos discretos.

Segundo Nourelfath e Niel (2004), o uso das técnicas de teoria de controle supervisório para modelagem de sistemas a eventos discretos possui a vantagem sobre outras abordagens de que os comportamentos resultantes não violam as especificações e são livres de impasses. Além disso, o comportamento do supervisor é máximo permissivo desde que ele esteja dentro do comportamento especificado, ou seja, todos os eventos que não contradizem as especificações são autorizados a acontecer.

Outro ponto interessante dessa abordagem é que os modelos obtidos dos sistemas a eventos discretos podem ser simulados. A simulação de sistemas de manufatura auxilia na tomada de decisões, pois é possível caracterizar impactos de mudanças de parâmetros no desempenho destes sistemas (BANKS et al., 2005; CHO, 2005; GARZA-REYES et al., 2010; SARGENT, 2009 apud COSTA , 2010). Garza-Reyes et al. (2010, apud COSTA, 2010) destaca o fato de que simulações permitem examinar o sistema sobre condições controladas, o que garante um ambiente seguro e livre de risco em comparação a condições reais.

A modelagem de sistemas a eventos discretos usando autômatos e linguagens formais apresenta uma abordagem simples e de fácil compreensão, se bem estruturada. Apresenta ferramentas que permitem controle seguro, além de possuir a possibilidade de utilizar simulação para obter resultados mais consistentes.

2 REVISÃO BIBIOGRÁFICA

2.1 Sistemas a eventos discretos

Os sistemas podem ser classificados de acordo com a característica que eles possuem. A figura 2.1 abaixo representa como estes podem ser classificados:

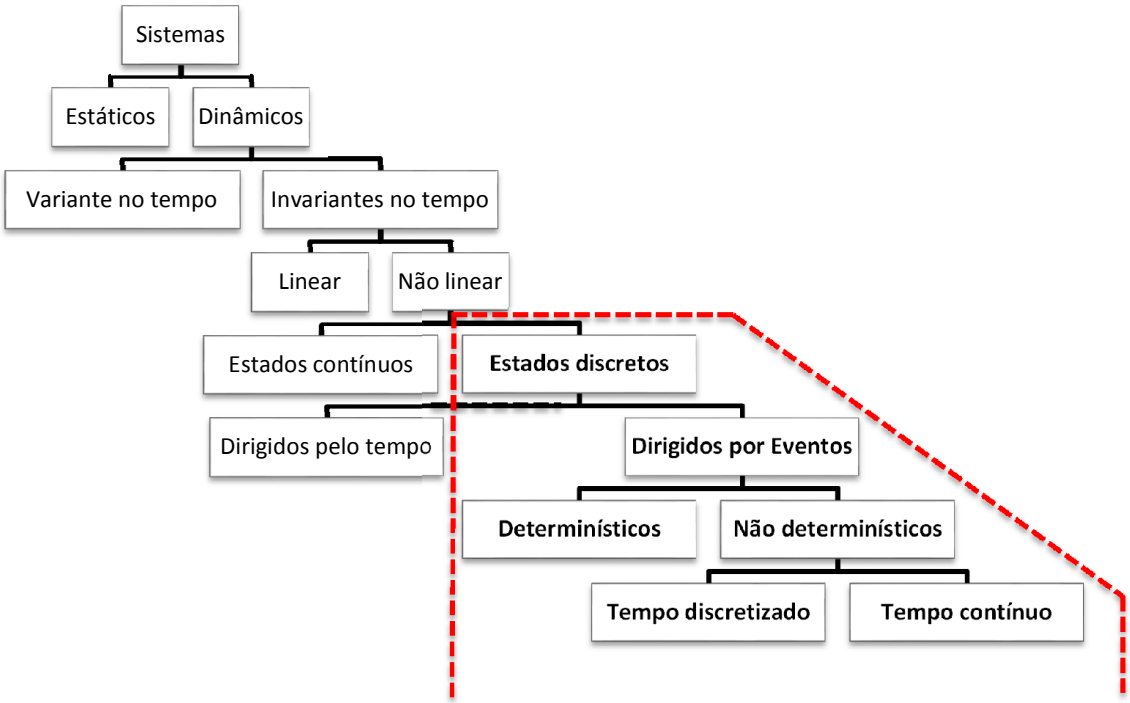

Figura 2.1: Classificação dos sistemas
Fonte: MONTGOMERY, 2004

A parte destacada da figura 2.1 compõe os sistemas a eventos discretos (SEDs). Observa-se que eles também estão dentro do conjunto de sistemas dinâmicos, invariantes no tempo e não lineares. Define-se sistemas dinâmicos como sistemas que a saída depende de valores anteriores da entrada (CASSANDRAS; LAFORTUNE, 2008). Se eles são invariantes no tempo, de acordo com Montgomery (2004), o deslocamento temporal em uma determinada entrada resulta em um deslocamento temporal na saída, de forma equivalente, sem afetar a transformação que o sistema realiza. Este comportamento pode ser descrito pela relação abaixo:

$$y(t) = g\big(u(t)\big) \leftrightarrow y(t-\tau) = g\,(u(t-\tau)) \tag{2.1}$$

Em que:

- y(t) é o vetor de saída;
- u(t) é o vetor aplicado a entrada;
- g(.) é a transformação de estado em relação a u(t).

Segundo Ogata (2002), sistemas não lineares são aqueles nos quais não se pode aplicar o principio da superposição.

A característica base dos SEDs é que eles são dependentes da ocorrência de eventos assíncronos (dirigidos por eventos) que definidos em intervalos discretos de tempo (estados discretos), os quais compreende o conjunto de inteiros. Os eventos são resultado de uma transição de estados que ocorre em intervalos irregulares de tempo (MONTGOMERY, 2004). Um exemplo de transição de estados é o simples ato de se apagar uma lâmpada, em que a lâmpada passa do estado aceso para o apagado.

Outra característica que um SED pode assumir é ser determinístico ou não determinístico. Quando ele apresenta um comportamento não determinístico significa que pelo menos uma de suas saídas é aleatória, ou seja, ele representa um processo estocástico, que só pode ser caracterizado com um modelo probabilístico. Sendo estocástico, este sistema pode ser de tempo contínuo, em que as variáveis de entrada, de estado e de saída estão definidas nos valores possíveis de tempo. Ou ele pode ser em tempo discreto, para o caso de suas variáveis assumirem apenas valores discretos no tempo, como em um processo de amostragem. (CASSANDRAS; LAFORTUNE, 2008).

O SED pode ser representado por uma sequência ordenada de eventos que convergem para realização de um objetivo particular (CURY, 2001). Há várias formas de se trabalhar e analisar SEDs. Nesse trabalho utiliza-se a abordagem das com Linguagens Formais e Autômatos.

2.2 Autômatos e Linguagens Formais

2.2.1 Linguagens Formais

Conforme Cassandras e Lafortune (2008), uma forma de se estudar o comportamento lógico de um SED é baseando-se em linguagens formais e autômatos. Para isso, considera-se um SED como um conjunto de eventos, Σ, chamado de alfabeto, e a sequências destes eventos forma uma palavra.

Uma palavra vazia, simbolizada por ε, representa uma palavra sem símbolo. Sendo s uma palavra, o comprimento desta é simbolizado por |s|. Assim, para um alfabeto Σ, a relação de

todas as palavras possíveis sobre ele é dado por Σ^* ou por Σ^+, sendo que neste último exclui-se a palavra vazia.

Exemplo 2.1: para um alfabeto $\Sigma\{\varepsilon, \alpha, \beta\}$, temos abaixo alguns possíveis representações de linguagens sobre este:

$$L_1 = \{\varepsilon, \alpha\alpha\} \tag{2.2}$$
$$L_2 = \{\alpha, \alpha\beta, \alpha\beta\alpha\beta, \dots\}$$
$$L_3 = \{\alpha, \beta\alpha, \beta\beta\alpha, \beta\beta\alpha, \dots\}$$

Segundo Montgomery (2004), para um alfabeto Σ, L é uma linguagem sobre ele apenas se $L \subseteq \Sigma$. Dessa forma, esta linguagem pode ser $L = \Sigma^*$ ou até mesmo $L = \{\}$. Observa-se que $L = \{\}$ é diferente de $L\{\varepsilon\}$, pois o último possui um elemento.
Algumas propriedades importantes sobre a linguagem serão definas abaixo:

Definição 2.1 (Concatenação) Concatenação é uma operação que associa palavras e linguagens fazendo a união da primeira com a segunda.

Exemplo 2.2: Sendo $L_1 = \{\alpha, \beta\beta\}$ e $L_2 = \{\gamma, \gamma\delta\}$ a concatenação de L_1 em relação L_2 é:

$$L_1L_2 = \{\alpha\gamma, \beta\beta\gamma, \alpha\gamma\delta, \beta\beta\gamma\delta\} \tag{2.3}$$

Observe que se for feita a concatenação de L_2 em relação a L_1 o resultado é diferente ($L_2L_1 = \{\gamma\alpha, \gamma\delta\alpha, \gamma\beta\beta, \gamma\delta\beta\beta\}$). Isso significa que esta operação não é comutativa, ou seja, $L_1L_2 \neq L_2L_1$..

Definição 2.2 (Prefixo-fechado). O fechamento de uma linguagem L denotado por $\bar{L}$ representa o conjunto de todos os prefixos desta linguagem. A linguagem é prefixo fechada quando $L = \bar{L}$.

Exemplo 2.3:

$$L = \{\varepsilon, \beta, \beta\beta, \dots\} \tag{2.4}$$
$$\bar{L} = \{\varepsilon, \beta, \beta\beta, \dots\}$$
$$L = \bar{L}$$

Definição 2.3 (Fechamendo-Kleene). O fechamendo-Kleene representado por L^* é definido por um número finito arbitrário de concatenação dos elementos de L, ou seja:

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup LLLL \cup \ldots \quad (2.5)$$

2.2.2 Expressões Regulares

De acordo com Menezes (2000), é possível representar toda linguagem regular, como os autômatos, por expressão simples, chamada de expressão regular (ER). Este formalismo é denominado também de gerador, pois é possível gerar palavras de uma linguagem. Algumas características das expressões regulares podem ser vistas abaixo:

- Para uma linguagem vazia a ER é representado por $\emptyset$;
- O símbolo + representa o "ou lógico";
- Sendo s uma palavra, para um ER com s*, pode repetir um número arbitrário de vezes;

Exemplo 2.4: Sendo o alfabeto $\Sigma = \{\alpha, \beta\}$ e sua linguagem $L = \{\epsilon, \alpha, \alpha\beta,\ \alpha\beta\alpha, \alpha\beta\alpha\beta, \ldots\}$ a expressão regular capaz de representar essa linguagem é:

$$(\alpha\beta)^*(\varepsilon + \alpha) \quad (2.6)$$

Observa-se que a expressão regular 2.6 é capaz de representar um conjunto variado de eventos, por exemplo, apenas os elementos α ou ε. Outra característica é que $(\alpha\beta)$ pode apresentar diferentes quantidades de incidências, isto tudo pela característica de fechamento-kleene $(*)$ adicionada a estes elementos. Porém, vale ressaltar que estes elementos só podem ocorrer de maneira conjunta, ou seja, α sempre ocorre seguido de β.

2.2.3 Autômatos

Segundo Cassandras e Lafortune (2008), autômato é um dispositivo capaz de representar uma linguagem com regras bem definidas. Um autômato pode representar o comportamento de um sistema dinâmico através de estados. Se a quantidade de estados é finita, o autômato é denominado autômato finito ou máquina de estado finito (MONTGOMERY, 2004).

Um autômato determinístico de estado finito (ADEF) é representado por uma quíntupla $G = (X, \Sigma, f, x_0, X_m)$ (CURY, 2001), em que:

- X: Conjunto finito de estados do autômato;
- Σ: Conjunto de símbolos (eventos) os quais definem um alfabeto;
- $f: X \times \Sigma \rightarrow X$: Representa a função de transição de estados, podendo ser parcial, sem que haja a necessidade de se definir para todo elemento do alfabeto Σ em cada estado X;

- x_0: Representa o estado inicial;
- X_m: Representa o conjunto de estados finais, chamados de estados marcados, $X_m \subseteq X$.

Exemplo 2.5: Considere o alfabeto $\Sigma = \{a, b, g\}$. Um autômato para este alfabeto pode ser visto na figura 2.2:

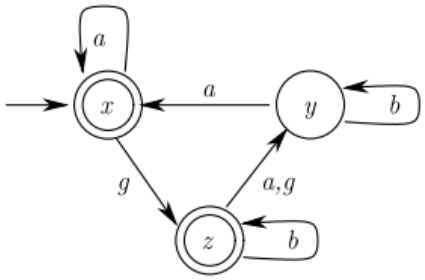

Figura 2.2: Diagrama de transição de estado para o exemplo 2.5
Fonte: CASSANDRAS e LAFORTUNE (2008)

Os arcos representam as transições de estados para os estados $X = \{x, y, z\}$. A função de transição de estado, $f: X \times \Sigma \rightarrow X$, para este autômato, pode ser vista abaixo na equação 2.7:

$$f(x,a) = x \qquad f(x,g) = z; \tag{2.7}$$
$$f(y,a) = x \qquad f(y,b) = y;$$
$$f(z,b) = z \qquad f(z,a) = f(z,g) = y.$$

A linguagem de um autômato representa o conjunto de palavras (conjunto de eventos) que levam de um estado inicial, x_0, para um estado de aceitação (HOPCROFT; ULLMAN; MOTWANI, 2002). Ela pode ser definida como "gerada", pois compreende todos os caminhos ao longo das transições de estados possíveis começando pelo estado inicial.

$$L(G) := \{ s \in \Sigma^* : f(x_0, s) \text{ é } definido \}, \tag{2.8}$$

ou ela pode ser definida como linguagem marcada, um subconjunto de $L(G)$ que consiste apenas em palavras (cadeias) "s" com $f(x_0, s) \in X_m$.

$$L_m(G) := \{ s \in L(G) : f(x_0, s) \in X_m \}. \tag{2.9}$$

Definição 2.4 (Equivalência). Sendo os autômatos G_1 e G_2, eles apresentam linguagens equivalentes apenas se (CASSANDRAS; LAFORNTE, 2008):

$$L(G_1) = L(G_2) \text{ e } Lm(G_1) = Lm(G_2) \tag{2.10}$$

2.2.4 Autômato gerador

Um gerador tem a mesma definição de um autômato com apenas uma diferença: para os autômatos a função de transição não pode ser parcial, ou seja, definida apenas para um subconjunto de eventos para cada estado do gerador. Por permitir transições parciais, os geradores são mais compatíveis com os SEDs. (MONTGOMERY, 2004).

Em outras palavras, um gerador é uma aproximação mais robusta utilizando autômatos de um sistema físico real, já que em sistema reais algumas transições podem não ser fisicamente possíveis em todos os estados, por exemplo:

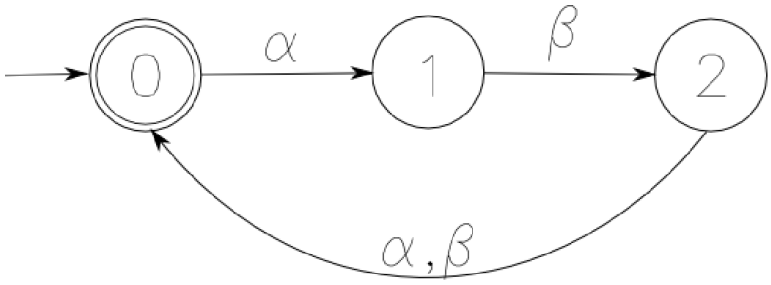

Figura 2.3: Gerador

A representação do gerador da figura 2.3 por um autômato seria, por exemplo:

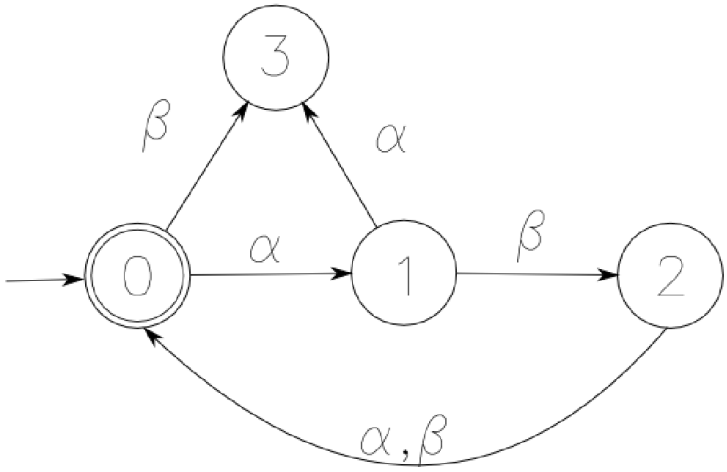

Figura 2.4: Representação por autômato

Neste caso no sistema "real", os eventos α e β não são possíveis nos estados 0 e 1 respectivamente.

Em seguida algumas definições importantes sobre geradores:

Definição 2.5 (Acessibilidade). Um estado $x \in X$ (conjunto de estados) é dito acessível se $x = f(x_0, u)$ para u $\in X$, ou seja, este estado pode ser alcançado a partir de estado inicial. Assim, um autômato determinístico de estado finito é acessível se seus conjuntos de estados

x for acessível para $x \in X$. Considerando um autômato G, é possível, através da eliminação de seus estados não acessíveis, obter uma componente acessível, G_{ac}, definida da seguinte forma (CURY, 2001):

$G_{ac} = (X_{ac}, \Sigma, f_{ac}, x_0, X_{m_{ac}})$, onde:

- X_{ac}: Conjunto de estado de G que são acessíveis;
- $f_{ac}: f \mid X_{ac} \times \Sigma \rightarrow X_{ac}$;
- $X_{mac}: X_{ac} \cap X_m$ Representa os estados finais, chamados de estados marcados, $X_m \subseteq X$;

Assim, para um autômato acessível, $G = G_{ac}$.

Definição 2.6 (Co-acessibilidade). Sendo um autômato G, ele é co-acessível (não bloqueante) se, a partir do estado inicial, existe pelo menos um caminho que leve para um estado marcado. A condição para co-acessibilidade é descrita abaixo:

$$L(G) = Lm(G) \tag{2.11}$$

Se um autômato for acessível e co-acessível simultaneamente ele é dito trim (CURY, 2001).

Definição 2.7 (Bloqueio). Um autômato é dito não bloqueante se satisfaz a equação abaixo:

$$L(G) = \overline{Lm(G)} \tag{2.12}$$

Definição 2.8 (Autômato não-determinístico de estados finitos). Um autômato não-determinístico de estado finito (ANDEF) é bastante similar ao ADEF, mas difere deste no comportamento da função de transição de estado, de forma que se um ANDEF recebe um símbolo de entrada, ele pode retornar um conjunto de zero, um, mais estados, diferente de um ADEF que só retorna um estado (HOPCROFT; ULLMAN; MOTWANI, 2002).

Exemplo 2.6: Considere um ANDEF com as seguintes funções de transição de estado: $f\,(1, b) = \{\,2\,\}$, $f(1, \varepsilon) = \{\,3\,\}$, $f(2, a) = \{\,2, 3\,\}$, $f\,(2, b) = \{\,3\,\}$, e $f\,(3, a) = \{\,1\,\}$.

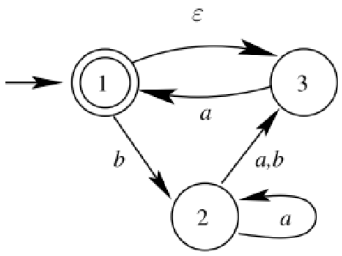

Figura 2.5: Autômato não determinístico com transição em ε para o exemplo 2.6
Fonte: CASSANDRAS e LAFORTUNE (2008)

Observe que $f\ (1, a)$ não é definido. Neste caso conclui-se que há uma transição de estado instantânea de 1 para 3 representado por ε.

Definição 2.9 (Composição de autômatos). A composição permite unificar dois ou mais autômatos em um único. Essa abordagem é interessante para SEDs com muitos estados, em que sua representação pode se tornar complexa demais. Neste caso, é recomendável tratar seus componentes de forma individual, ou dividi-los em subsistemas, que ao final serão reagrupados pelo processo de composição. A composição pode ser por produto ($\times$) ou paralela ($||$), também chamada de composição síncrona e composição síncrona completa, respectivamente (CASSANDRAS; LAFORTUNE, 2008). A figura 2.6 representa um exemplo de composição por produto entre dois autômatos.

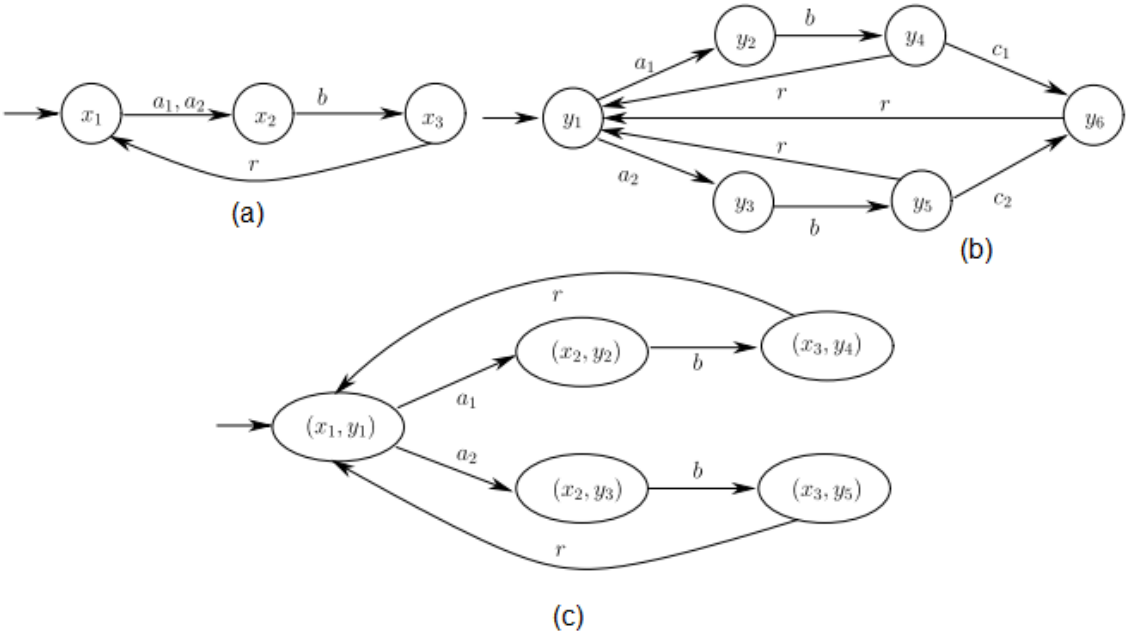

Figura 2.6: Composição por produto (c) entre dois autômatos (a) e (b).
Fonte: CASSANDRAS; LAFORTUNE, 2008

2.3 Teoria de controle supervisório de SEDS

2.3.1 Controle Monolítico

Segundo Cury (2001), um sistema está sujeito a um conjunto de restrições que definem suas especificações de comportamento. As linguagens $L(G)$ e $L_m(G)$ deste sistema podem ter palavras e cadeias que levem a comportamentos indesejados que violem essas restrições, como um choque de componentes, bloqueios, entre outros problemas. Para garantir que este sistema trabalhe de forma coordenada, insere-se um supervisor, S. Este é capaz de interagir

com a planta G, em malha fechada, e definir, a partir dos estados observados, quais devem ser permitidos. A representação da malha de controle pode ser visto na figura 2.7:

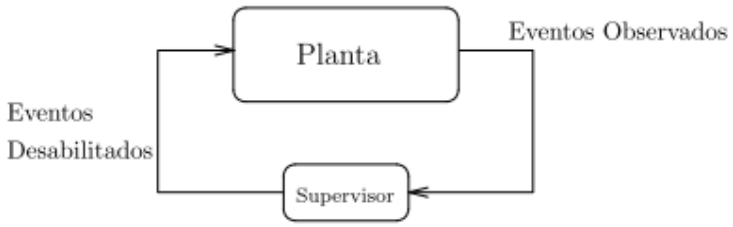

Figura 2.7: SED em malha fechada
Fonte: CURY (2001)

O alfabeto, Σ, que representa um conjunto de eventos da planta, pode ser dividido em dois subconjuntos ($\Sigma = \Sigma_c \cup \Sigma_{nc}$):

- Σ_c: Conjunto de eventos controláveis, os quais podem ser previstos ou desabilitados pelo supervisor;
- Σ_{uc}: Conjunto de eventos não controláveis, que não podem ser previstos pelo supervisor.

Observe que $\Sigma_c \cap \Sigma_{uc} = \emptyset$. Assim, é possível definir uma entrada de controle, Γ, que representa um conjunto de eventos habilitados em determinado estado pelo supervisor. Esta pode ser caracterizada por:

$$\Gamma = \{\gamma | \Sigma_{uc} \subseteq \gamma \subseteq \Sigma\} \quad (2.13)$$

Para uma entrada de γ, o conjunto de eventos não controláveis, por não sofrer com ações de controle, são considerados sempre habilitáveis, ou seja, $\Sigma_{uc} \subseteq \gamma$ (MONTGOMEY, 2004).
De acordo com Cury (2001), um SED obtido pela composição síncrona do supervisor S com a planta G descreve um sistema controlado S/G. O controle realizado por este supervisor é dito monolítico, pois existe apenas um supervisor responsável por habilitar e desabilitar eventos controláveis. Dessa forma, o supervisor restringe o comportamento de G além de ter a função de demarcar estados. Assim, uma tarefa é considerada completa, em malha fechada, quando marcada pela planta e pelo supervisor. A composição síncrona S||G resulta em um sistema controlado em que somente transições permitidas tanto no supervisor quanto na planta são permitidas neste.

Definição 2.10 (Problema de controle supervisório). Sendo uma planta G com especificação definida por $A \subseteq E \subseteq \Sigma^*$, existe um supervisor não bloqueante S para G tal que (CURY, 2001):

$$A \subseteq L_m(S/G) \subseteq E \quad (2.14)$$

Sendo A e E os limites superior e inferior, respectivamente, em malha fechada.

Definição 2.11 (Controlabilidade I). Segundo Cury (2001), para uma planta G com comportamento $(L(G), L_m(G))$ e estrutura de controle Γ, em um conjunto de eventos Σ e linguagem $E \subseteq L(G)$. Ela é dita controlável se:

$$\bar{E}\Sigma_u \cap L \subseteq \bar{E} \quad (2.15)$$

Definição 2.12 (Controlabilidade II). Segundo Cury (2001), para uma planta G, com comportamento $(L(G), L_m(G))$ e estrutura de controle Γ, em um conjunto de eventos Σ e a linguagem $K \subseteq L_m(G)$. Existe um supervisor não bloqueante tal que:

$$L_m(S/G) = K \quad (2.16)$$

Apenas se K for controlável.

Definição 2.13 (Solução do problema de controle supervisório). Segundo Cury (2001), o problema de supervisório só tem solução se:

$$supC(E) \supseteq A \quad (2.17)$$

Neste caso, $supC(E)$ representa o comportamento menos restritivo possível de se implementar por supervisão no SED G que satisfaz as especificações de A e E. Para isto, $L_m(S/G) = sup\, C(E)$.

De acordo com Queiroz e Cury (2001), a teoria de controle supervisório apresentada por Ramadge e Wonham (1989 apud QUEIROZ; CURY, 2002) permite a síntese de supervisor e o cálculo da máxima linguagem controlável para se obter um comportamento desejado de uma linguagem. Porém, Queiroz e Cury (2001) afirmam que a complexidade deste cálculo possui crescimento exponencial com o aumento do número de estados. Logo, para sistemas complexos, o controle monolítico é pouco eficiente e exige um esforço computacional muito grande.

2.3.2 Controle Modular

O controle modular introduzido por Wonham e Ramadge (1988 apud QUEIROZ, 2004) surge como alternativa para tratar sistemas compostos, já que estes envolvem um grande número de especificações. Neste tipo de abordagem, o sistema é divido por módulos, onde há um supervisor para cada uma das especificações. A ação conjunta dos supervisores realiza o controle sobre a planta global, habilitando ou desabilitando eventos controláveis (QUEIROZ, 2004). A figura 2.8 ilustra este tipo de controle.

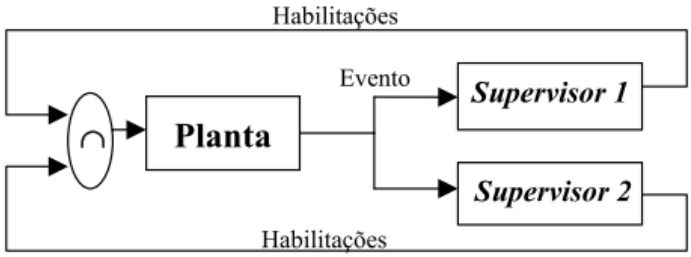

Figura 2.8: Esquema do controle modular
Fonte: QUEIROZ (2004)

Segundo Queiroz (2004), o controle modular permite tratar problemas complexos reduzindo-os em problemas mais simples (módulos). Os módulos, em geral, são flexíveis e de fácil alteração, ou seja, para se corrigir, atualizar e modificar um comportamento de uma planta, usando esta abordagem, pode-se apenas atuar sobre o supervisor de um módulo específico responsável por este comportamento, sem precisar alterar o sistema todo. Por outro lado, Queiroz (2004) afirma que os módulos possuem ação fragmentada em relação ao sistema global, isto pode gerar conflitos como o bloqueio do sistema na ação conjunta de controle. O uso da abordagem de controle modular, dessa forma, possui a tendência de ser degradado em relação ao controle monolítico.

Segundo Wonham e Ramadge (1988 apud QUEIROZ; CURY, 2002) a condição necessária e suficiente para que o resultado do controle modular seja igual ao monolítico é a modularidade de sua linguagem marcada, ou seja, o supervisor deve ser ótimo e não bloqueante. Para que o supervisor não seja bloqueante ele deve satisfazer a seguinte condição:

Definição 2.14: Sejam o conjunto de linguagens, $L_i \subseteq \Sigma^*, i = 1,\ldots,n$, estas serão modulares, não bloqueantes, apenas se $\cap_{i=1}^{n} \overline{L_i} = \overline{\cap_{i=1}^{n} L_i}$. Isto significa que sempre que um prefixo for aceito em um conjunto de linguagem, uma palavra que contém este prefixo deverá ser aceita por este conjunto para que não se gere conflitos (QUEIROZ; CURY, 2002).

Entretanto, Queiroz (2004) destaca que o controle modular clássico gera especificações expressas em relação ao sistema global, de forma que com o aumento do número de subsistemas a quantidade de especificações cresce exponencialmente. Embora o controle modular seja melhor que a abordagem monolítica, em se tratando de sistemas compostos, ele possui também uso limitado com o aumento da complexidade destes sistemas.

2.3.3 Controle Modular Local

A fim de explorar a modularidade das especificações e da planta, como também as características de sistemas descentralizados, como é o caso de sistemas de manufatura, Queiroz e Cury (2000) propõem um novo tipo de abordagem chamada de controle modular

local (QUEIROZ; CURY,2002). Esta, segundo Queiroz (2004), permite implementar supervisores a partir de modelos de tamanho menor, de forma a diminuir o esforço computacional no processo de síntese e de implementação de controle destes.

Definição 2.15: Seja o conjunto de linguagens $L_i \subseteq \Sigma^*, i = 1,\ldots,n$, este é localmente modular se $\cap_{i=1}^{n} \overline{P_i^{-1}(L_i)} = \overline{\cap_{i=1}^{n} P_i^{-1}(L_i)}$, ou seja, $||_{i=1}^{n} \overline{L_i} = \overline{||_{i=1}^{n} L_i}$. Isto significa que se para $i = 1,\ldots,n$, linguagens L_i forem marcadas por geradores aparados G_i , é possível provar que a modularidade local é verificada apenas se $G = ||_{i=1}^{n} G_i$ for um gerador aparado (QUEIROZ; CURY, 2002).

Vale ressaltar, segundo Queiroz e Cury (2002), que a modularidade local de todos os subconjuntos não garante a modularidade do conjunto total. Assim, não é recomendável verificar a modularidade de forma decomposta, requerendo então uma composição síncrona.

A máxima linguagem para múltiplas restrições pode ser executada, a partir das restrições locais, sem perda de desempenho, mesmo com o aumento das restrições em relação ao controle monolítico. Para isto, a modularidade local precisa ser válida (QUEIROZ, 2004).

Para cada especificação, Queiroz e Cury (2002) afirmam que é possível construir um supervisor não bloqueante. Este será responsável por desabilitar eventos controláveis da sua respectiva planta local, gerando, em malha fechada, a máxima linguagem local controlável. Na figura 2.9 é possível observar os supervisores locais atuando sobre um sistema descentralizado, em que estes supervisores possuem ação de controle apenas sobre seus respectivos subsistemas.

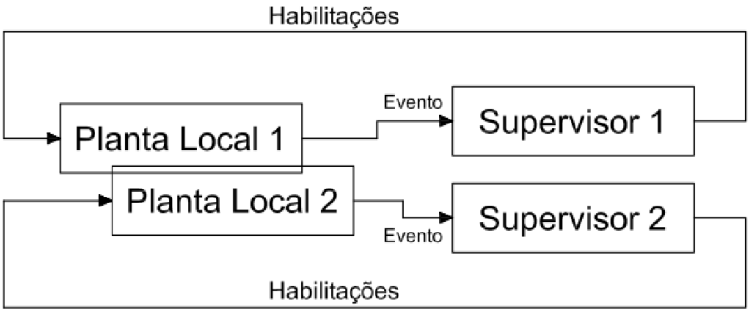

Figura 2.9: Esquema do controle modular local
Fonte: Queiroz (2004)

Existem casos em que a modularidade das restrições não é verificada, significando que há supervisores em conflito que impendem o uso da abordagem modular local (QUEIROZ; CURY, 2002). Para este caso, Queiroz e Cury (2002) sugerem o controle monolítico para o menor subconjunto de especificações conflitantes. Eles ainda indicam abordagens mais

elaboradas, como as apresentadas na literatura de Wong et all (1995), Chen et all (1995) e Wong e Wonham (1998).

Um conceito importante agregado ao controle modular local é a representação por sistema produto (RSP). Este conceito, segundo Queiroz e Cury (2002), auxilia na diminuição do esforço computacional. Para isto, é preciso que sejam feitas composições síncronas entre os subsistemas. Deve-se gerar um conjunto com o maior número de subsistemas que apresentem comportamento síncrono, ou seja, possuam estados comuns. Para o conjunto resultante com subsistemas assíncronos distintos, aplica-se uma representação por sistema produto (RSP), que através de um mínimo esforço computacional com operações de produto é capaz de modelar o sistema global em malha aberta.

2.3.4 Síntese de supervisores ótimos

Ramadge e Wonham (1982 apud MONTGOMERY, 2004) desenvolveram uma abordagem, chamada R-W, para síntese de supervisores ótimos. Segundo Cury (2001), essa abordagem pode ser desenvolvida em 3 partes:

- Geração de um modelo para a planta a ser controlada;
- Obtenção de um modelo que representa as restrições a serem respeitadas;
- Síntese de uma lógica de controle não bloqueante.

2.3.4.1 Passo para obtenção de um modelo para a planta

De acordo com Cury (2001), aplicando uma abordagem local obtém-se a planta da seguinte forma:

1. Identificar e separar o conjunto de equipamentos e sub-sistemas do SED a ser controlado;
2. Construir o modelo, G_i, de ADEF para cada componente i identificado;
3. Realizar uma composição síncrona sobre todos os componentes G_i para se obter o modelo da planta a ser controlada;
4. Definir uma estrutura de controle, através dos eventos controláveis e não controláveis, que influenciam no sistema.

2.3.4.2 Passo para obtenção de um modelo para especificação

Para se obter uma linguagem $E \subset L_m(G)$ que atenda a especificação de comportamento do sistema, pode-se usar também uma abordagem local. Os passos para se obter uma especificação global podem ser vistos abaixo (CURY, 2001):

1. Obter autômatos E_j para as restrições de coordenação j, do SED a ser controlado;
2. Fazer a composição síncrona destes autômatos, compondo o resultado com o autômato da planta G, gerando um autômato K.
3. Atualizar K, de acordo com a eliminação de estados proibidos e pelo cálculo da componente co-acessível, ou não bloqueante de K, que atendam a especificação global de E, dada por: $E = L_m(K)$.

Deve-se fazer algumas considerações deste processo:

- O autômato resultante K (passo 2) deve ter: $L_m(K) \subset L_m(G)$;
- Considerando que K pode ter estados proibidos, após a eliminação destes pode-se afirmar que: $L\,m(K) = E$.
- Ao fim, com o cálculo da componente co-acessível e a eliminação dos bloqueios pode-se dizer que:

$$L_m(K) = E \subset L_m(G) \text{ e } L(K) = L_m(K) \qquad (2.18)$$

2.3.4.3 Passo para síntese do supervisor não bloqueante ótimo

Segundo Cury (2001), há casos onde E não atende as condições de controlabilidade. Neste caso, deve-se calcular uma linguagem controlável que se aproxime de E e que não tenha comportamento indesejável.

Este procedimento representa o cálculo de $sup\,C(E)$ e pode ser descrito através de um método iterativo que identifica os maus estados do autômato K, resultante da composição síncrona, realizada no passo 2, para obtenção de um modelo que atentada a especificação. Este método pode ser visto abaixo, segundo Cury (2001):

Para $G = (X, \Sigma, f, x_0, X\,m)$ e $K = (Q, \Sigma, f\,R, q_0, Q_m)$, calculado nas alíneas 2.3.1.1 e 2.3.1.2 considerando $L_m(K) = E \subset L_m(G)$;

1. Se existir, identificar maus estados em K; caso contrário faça $S = K$, terminar método;
2. Caso existam, eliminar estes e atualizar K a cada eliminação;
3. Obter a componente trim de K, retornar ao passo 1.

Um estado $q(x,.)$ é definido como mau estado se um símbolo $\sigma \in \Sigma_u$ e para planta G, $\sigma \in \Sigma(x)$, mas $\sigma \notin \Sigma_u$.

3 DESCRIÇÃO DO PROBLEMA

O problema proposto consiste no projeto de um sistema flexível de manufatura aplicando técnicas de sistemas a eventos discretos. Numa primeira etapa, faz-se a modelagem do problema utilizando linguagens formais e autômatos. Uma vez obtido o modelo deste sistema, projeta-se um controlador (supervisor) utilizando o controle modular local, uma das abordagens da teoria de controle supervisório, para controlar a planta de forma que todas as especificações de segurança sejam respeitadas.

Os tópicos seguintes apresentam todo processo de construção do problema, tomando como base o passo a passo para síntese de supervisores ótimos, descrito no capitulo anterior.

3.1 Sistema flexível de manufatura

A planta da figura 3.1, simula o comportamento de uma linha de produção flexível, em que é possível produzir certa variedade de peças usando os mesmo equipamentos.

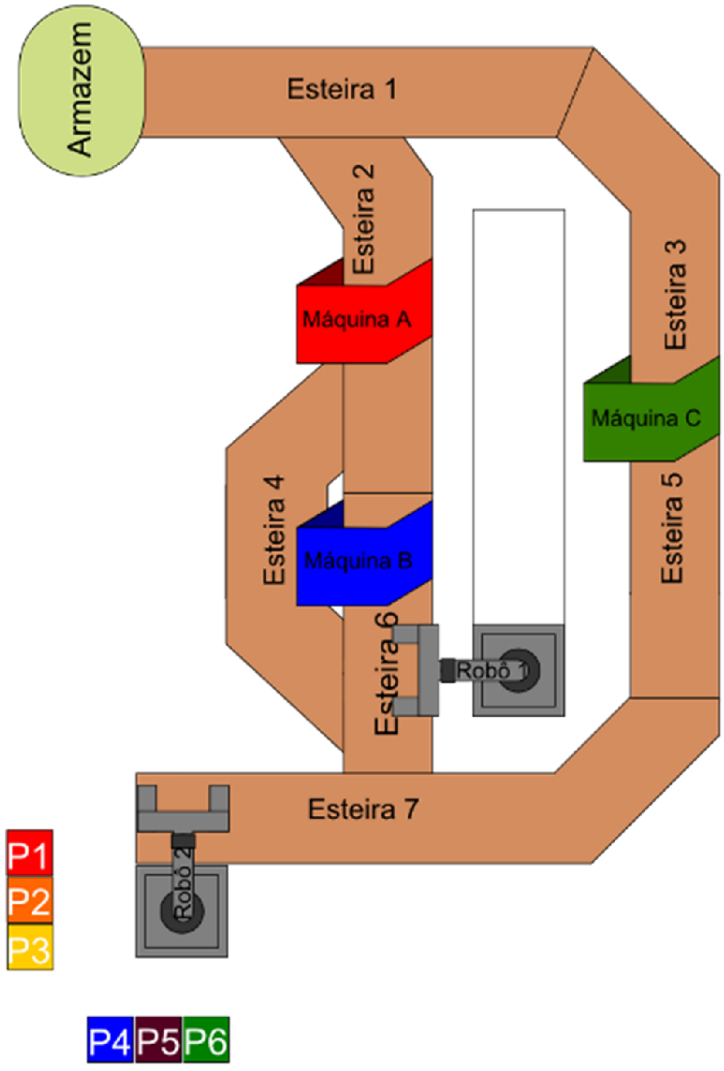

Figura 3.1: Sistema flexível de manufatura

A planta pode ser dividida em vários segmentos, de acordo com a figura 3.1. O primeiro é a esteira (ou tapete) superior, que corresponde a esteira 1 (Est 1). Este é responsável por levar a matéria prima do armazém para as esteiras internas (Est 2 e Est 3), podendo utilizar 2 caminhos. O primeiro caminho é do tapete 1 para 2 (Est 2), para que o material seja processado pela máquina A (MA). O segundo caminho é da esteira 1 para 3 (Est 3). Neste caso, a matéria prima será trabalhada na máquina C (MC).

As esteiras internas (2, 3, 4, 5 e 6) compreendem os tapetes responsáveis por ligar a esteira superior e a inferior (esteira 7). Após o material ser trabalhado na máquina A, ele pode seguir por dois caminhos, a partir da esteira 4 (Est 4). No primeiro caminho, o material vai direto para a esteira 6. No segundo, a peça passa pela máquina B onde será processada. Entre as esteiras internas há um robô (1) que possibilita a movimentação de peças do tapete 6 (Est 6) para o 3 ou do tapete 5 (Est 5) para o 2. Esta movimentação é importante, pois garante que todas as operações entres as peças sejam possíveis, por exemplo, uma peça que sofre transformação nas máquinas A e C.

A esteira inferior corresponde a esteira 7 (Est 7). Esta transporta a peça final para que o robô 2 encaminhe esta peça para seu depósito específico.

Esta planta é capaz de produzir 6 tipos de peças, de acordo com a sequência de operações que sofre nas máquinas. A relação de transformações de peças pode ser vista na tabela 3.1.

Tabela.3.1: Tabela de transformação de peças a partir da matéria prima

Peça	**Operação na máquina**
P1	MA
P2	MA→MB
P3	MA→MB→MC
P4	MC
P5	MC→MA
P6	MC→MA→MB

3.2 Modelagem da planta

Para modelar o sistema flexível de manufatura utilizou-se autômatos, uma das técnicas empregadas para modelar sistemas a eventos discretos.

A primeira etapa da modelagem da planta foi construir um diagrama que representasse todos os equipamentos e o conjunto de estados. A figura 3.2 apresenta uma visão global do sistema.

Os eventos ímpares são chamados de controláveis e os pares de não controláveis. Os eventos controláveis são aqueles que podem ser desabilitados pela ação de controle. Já os eventos não controláveis não sofrem nenhuma influência da ação de controle, ou seja, eles não podem ser impedidos de ocorrer. A tabela 3.2, apresenta uma descrição detalhada sobre a ação de cada evento (E) que ocorre na planta.

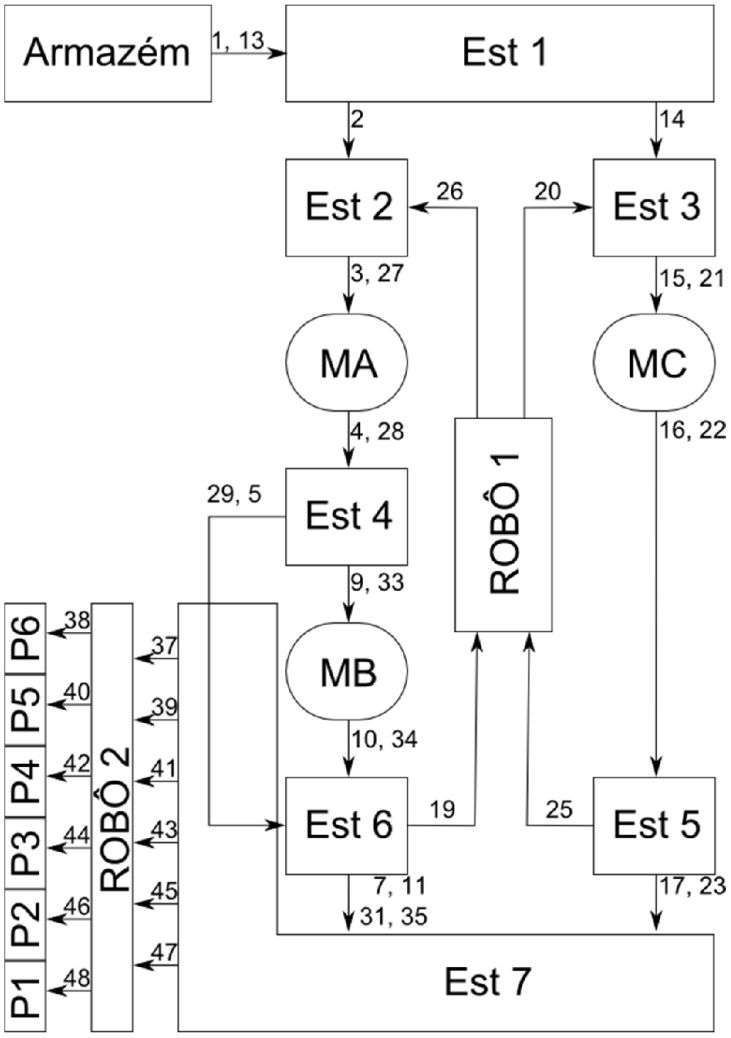

Figura 3.2: Modelo global da planta

Tabela 3.2: Conjunto de eventos da planta

Armazém → Est 1			
E		**E**	
1	Direciona MP da esteira 1 para 2	**13**	Direciona MP da esteira1 para 3
	Est 1 → Est 2		**Est 1 → Est 3**

2	Leva a MP da esteira 1 para 2	**14**	Leva a MP da esteira 1 para 3
	Est 2 → MA		**Est 3 → MC**
3	Leva a MP da esteira 2 para MA	**15**	Leva a MP da esteira 3 para MC
27	Leva a peça 4 da esteira 2 para MA	**21**	Leva a peça 3 da esteira 2 para MC
	MA → Est 4		**MC → Est 5**
4	Leva a peça 1 da MA para esteira 4	**16**	Leva a peça 4 da MC para esteira 5
28	Leva a peça 5 da MA para esteira 4	**22**	Leva a peça 3 da MC para esteira 5
	Est 4 → MB		**MB → Est 6**
9	Leva a peça 1 da esteira 4 para MB	**10**	Leva a peça 2 da MB para esteira 6
33	Leva a peça 5 da esteira 4 para MB	**34**	Leva a peça 6 da MB para esteira 6
	Est 4 → Est 6		**Est 5 → Robô 1**
29	Leva a peça 1 da esteira 4 para esteira 6	**25**	Leva a peça 4 da esteira 5 para Robô 1
5	Leva a peça 5 da esteira 4 para esteira 6		
	Est 6 → Robô 1		**Robô 1 → Est 3**
19	Leva a peça 1 da esteira 6 para Robô 1	**20**	Leva a peça 1 do Robô 1 para esteira 3
	Robô 1 → Est 2		**Est 5 → Est 7**
26	Leva a peça 4 do Robô 1 para esteira 2	**17**	Leva a peça 3 da esteira 5 para esteira 7
		23	Leva a peça 4 da esteira 5 para esteira 7
	Est 6 → Est 7		
7	Leva a peça 1 da esteira 6 para esteira 7	**31**	Leva a peça 5 da esteira 6 para esteira 7
11	Leva a peça 2 da esteira 6 para esteira 7	**35**	Leva a peça 6 da esteira 6 para esteira 7
	Est 7 → Robô 2		**Robô 2 → Depósitos**
37	Leva a peça 1 da esteira 7 para Robô 2	**38**	Leva a peça 1 da Robô 2 para Depósito P1
39	Leva a peça 2 da esteira 7 para Robô 2	**40**	Leva a peça 2 da Robô 2 para Depósito P2
41	Leva a peça 3 da esteira 7 para Robô 2	**42**	Leva a peça 3 da Robô 2 para Depósito P3
43	Leva a peça 4 da esteira 7 para Robô 2	**44**	Leva a peça 4 da Robô 2 para Depósito P4
45	Leva a peça 5 da esteira 7 para Robô 2	**46**	Leva a peça 5 da Robô 2 para Depósito P5
47	Leva a peça 6 da esteira 7 para Robô 2	**48**	Leva a peça 6 da Robô 2 para Depósito P6

Com base nas definições da Figura 3.2 e da tabela 3.2, parte-se para a modelagem dos autômatos, sendo um autômato para cada equipamento. Os modelos foram obtidos usando o programa TCT (WONHAN, 2014) e podem ser vistos nas Figuras 3.3, 3.4, 3.5. Observe que para todos os autômatos o estado marcado é o "0", que simboliza que a tarefa sempre estará completa neste estado.

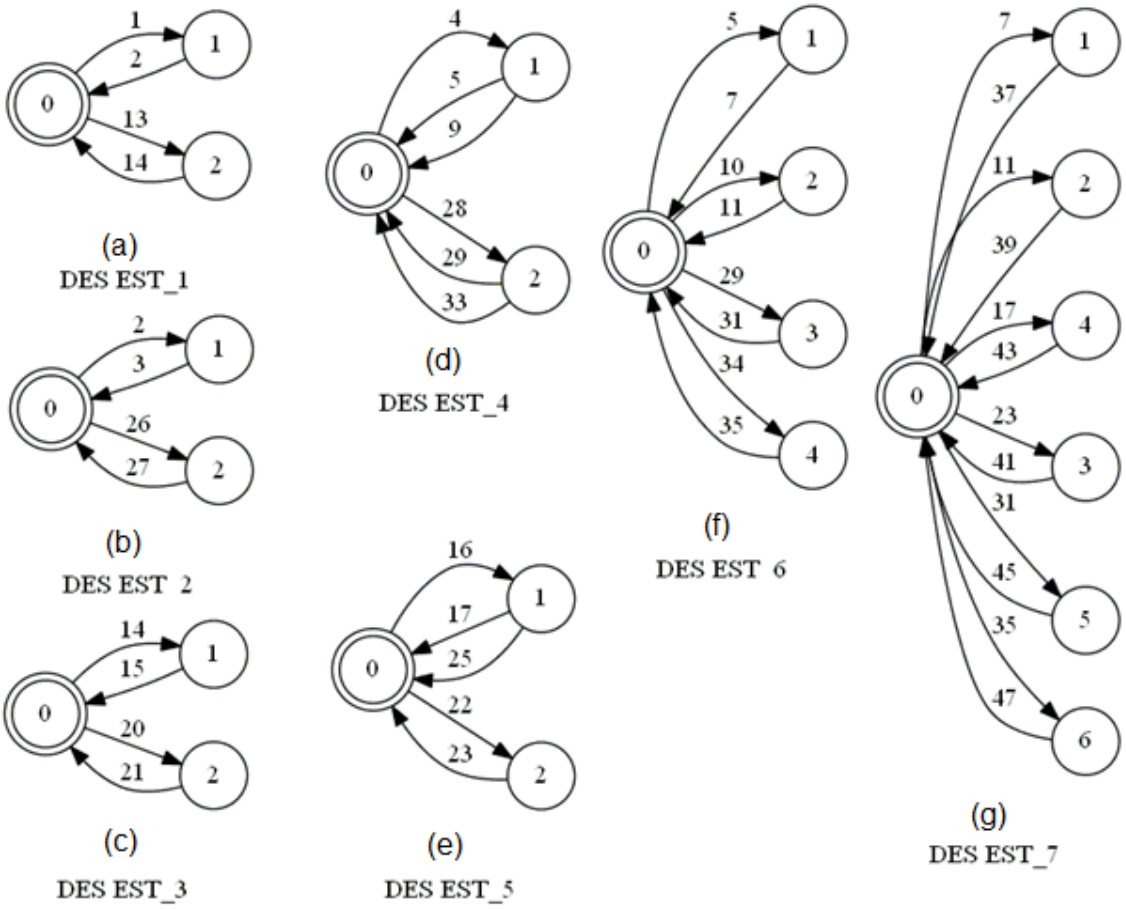

Figura 3.3: Autômatos das esteiras

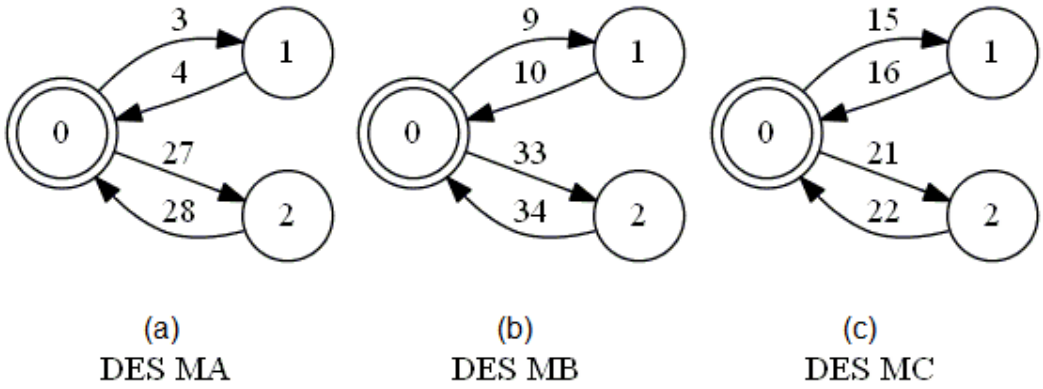

Figura 3.4: Autômatos das máquinas A, B e C

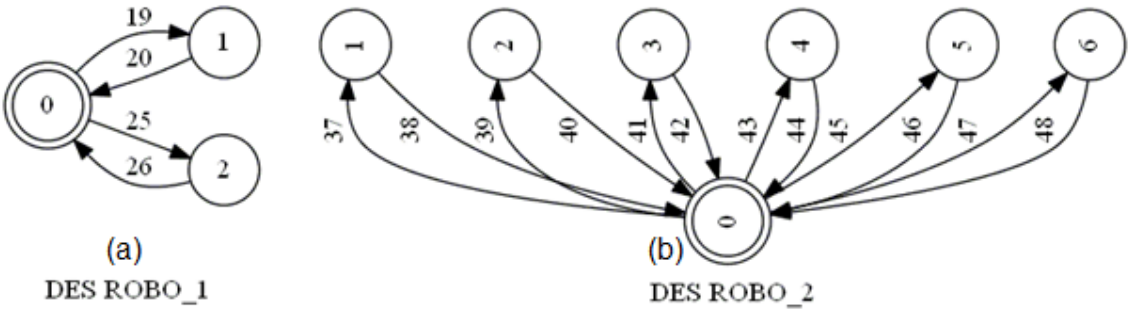

Figura 3.5: Autômatos dos robôs 1 e 2

Os estados relativos aos autômatos de cada equipamento da planta são descritos nas tabelas 3.3, 3.4. 3.5.

Tabela 3.3: Conjunto de estados das esteiras

Esteiras	Estados	Descrição
1	→0*	Esteira 1 sem MP
	1	Esteira 1 com MP direcionada para esteira 2
	2	Esteira 1 com MP direcionada para esteira 3
2	→0*	Esteira 2 sem Material
	1	Esteira 2 com MP
	2	Esteira 2 com P4
3	→0*	Esteira 3 sem Material
	1	Esteira 3 com MP
	2	Esteira 3 com P2
4	→0*	Esteira 4 sem peça
	1	Esteira 4 com P1
	2	Esteira 4 com P5
5	→0*	Esteira 5 sem peça
	1	Esteira 5 com P4
	2	Esteira 5 com P3
6	→0*	Esteira 6 sem peça
	1	Esteira 6 com P1
	2	Esteira 6 com P2
	3	Esteira 6 com P5
	4	Esteira 6 com P6
7	→0*	Esteira 7 sem peça
	1	Esteira 7 com P1
	2	Esteira 7 com P2
	3	Esteira 7 com P3
	4	Esteira 7 com P4
	5	Esteira 7 com P5
	6	Esteira 7 com P6

Tabela 3.4: Conjunto de estados dos robôs

Robô	Estados	Descrição
1	→0*	Robô 1 sem peça
	1	Robô 1 com peça 2
	2	Robô 1 com peça 4
2	→0*	Esteira 7 sem peça
	1	Esteira 7 com P1
	2	Esteira 7 com P2
	3	Esteira 7 com P3
	4	Esteira 7 com P4
	5	Esteira 7 com P5
	6	Esteira 7 com P6

Tabela 3.5: Conjunto de estados das máquinas

Máquina	Estados	Descrição
MA	→0*	Máquina A sem peça
	1	Máquina A com MP
	2	Máquina A com P4
MB	→0*	Máquina B sem peça
	1	Máquina B com P1
	2	Máquina B com P5
MC	3	Máquina C sem peça
	4	Máquina C com MP
	5	Máquina C com P2

Para fabricar cada tipo de peça, o sistema segue uma sequência de eventos, que pode ser visto abaixo:

- $P1\colon 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 37 \rightarrow 38$ (3.1)
- $P2\colon 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 39 \rightarrow 40$
- $P3\colon 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 19 \rightarrow 20 \rightarrow 21 \rightarrow 22 \rightarrow 23 \rightarrow 41$
- $P4\colon 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 17 \rightarrow 43 \rightarrow 44$
- $P5\colon 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 25 \rightarrow 26 \rightarrow 27 \rightarrow 28 \rightarrow 29 \rightarrow 31 \rightarrow 45 \rightarrow 46$
- $P6\colon 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 25 \rightarrow 26 \rightarrow 27 \rightarrow 28 \rightarrow 33 \rightarrow 34 \rightarrow 35 \rightarrow 47$

3.3 Especificações de segurança do sistema

Para um funcionamento correto do sistema em estudo, é necessário definir as restrições que o limita, isto é, evitar comportamentos indesejáveis, como exceder a quantidade de material em determinado local, chocar um robô com um equipamento, entre outros problemas que podem causar danos na linha de produção, ou até mesmo a saúde das pessoas envolvidas diretamente com o processo de produção desta linha. Este passo é importante para construção do supervisor.

Na planta utilizada, tem-se como restrições para todas as máquinas e esteiras, o limite de apenas uma peça por vez. Assim, quando a matéria prima entra na esteira 1, apenas quando ela estiver na esteira 2 ou 3 outro material pode ser inserido novamente na esteira 1. O mesmo acontece para todas as esteiras e máquinas do sistema. Em particular, a esteira 4 só terá peça se a esteira 6 e a máquina B estiverem vazias, já que estes segmentos compartilham o mesmo destino (esteira 6). O robô 1 também só transporta uma peça por vez. Além disso, ele só pode transportar peça entre dois tapetes se a esteira de destino estiver livre. O robô 2, assim como todos equipamentos, opera apenas uma peça por vez, sendo o destino destas peças os depósitos (P1, P2, P3, P4, P5 e P6) com capacidade para 50 peças cada.

Para garantir que as restrições expostas sejam bem estruturadas e não gerem conflitos, é preciso analisar as interações do sistema. Dessa forma, é possível estabelecer subsistemas com equipamentos que compartilhem os mesmos recursos. Por exemplo, tomando como base o caminho percorrido da matéria prima até a máquina A, observa-se que a esteira 2 é o ponto crítico deste trajeto, já que pode-se violar a condição unitária de material nesta esteira se o robô 1 estiver alocando a peça 4 no momento que a matéria prima estiver passando na esteira 2. Por outro lado, a ocorrência da ação inserir material na esteira 2 está condicionada aos eventos 2 (Est1 → Est 2) e 26 (Robô 1→ Est 2), representados na figura 3.5 e na tabela 3.2. Estes eventos também fazem parte do autômato correspondente a esteira 2 e só podem ocorrer alternadamente. Isto garante a não violação da capacidade de material no tapete 2, como também de todos os componentes que interagem com esta esteira, já que eles possuem eventos correlacionados. Logo, o autômato desta esteira representa uma restrição, definida como E_1, para o subsistema formado pelo tapete 1, o robô 1 e a máquina A, chamado de G_1.

A esteira 3 possui atuação análoga a esteira 2, atuando como restrição (E_4) para o subsistema, definido como G_4, formado pelos equipamento que interagem com esta. São eles: esteira 1, robô 1 e máquina A. Por sua vez, a esteira 4 faz a ligação entre a máquina A e a máquina C ou a esteira 6. Dessa forma, a fim de garantir que nesse conjunto de equipamentos apenas uma peça por vez possa ocupá-los, o autômato do tapete 4 atua como restrição (E_2) para este subsistema (máquina A, máquina B e esteira 6) definido como G_2. Observe-se que E_2 só permite que as peças 1 e 5 tomem uma direção, ou seja, ou ela irá para a máquina B (eventos 9 ou 33) ou para a esteira 6 (eventos 29 e 5).

De maneira semelhante, todo o sistema foi divido em subsistemas e, para cada um destes, são definidas restrições que garantam sempre o fluxo unitário de peça nos equipamentos e evitem conflitos em pontos que há mais de uma entrada de peças. A relação dos subsistemas e suas restrições podem ser vista na tabela 3.6. Note que para formar os subsistemas é necessário que

seja feita uma composição síncrona entre seus autômatos. Na figura 3.6 é possível observar os autômatos referentes a cada restrição. Na figura 3.7 tem-se o autômato que representa o subsistema G_1. Todo os resultados foram obtidos pelo TCT (WONHAN, 2014). Observa-se que devido ao crescimento exponencial do número de estados e transições fica inviável apresentar todos os subsistemas.

Tabela 3.6: Restrições e subsistemas

Subsistema	Componentes	Nº de estados	Nº de transições	Restrições
G_1	$Est\ 1 \mid\mid MA \mid\mid Rob\hat{o}\ 1$	27	108	E1 (Est 2)
G_2	$MA \mid\mid MB \mid\mid Est\ 6$	45	150	E2 (Est 4)
G_3	$MB \mid\mid Est\ 7 \mid\mid Est\ 4 \mid\mid Rob\hat{o}\ 2$	189	996	E3 (Est 6)
G_4	$Est\ 1 \mid\mid MC \mid\mid Rob\hat{o}\ 1$	27	108	E4 (Est 3)
G_5	$MC \mid\mid Est7 \mid\mid Rob\hat{o}\ 1$	63	276	E5 (Est 5)
G_6	$Rob\hat{o}\ 2$	7	12	E6 (Est 7)

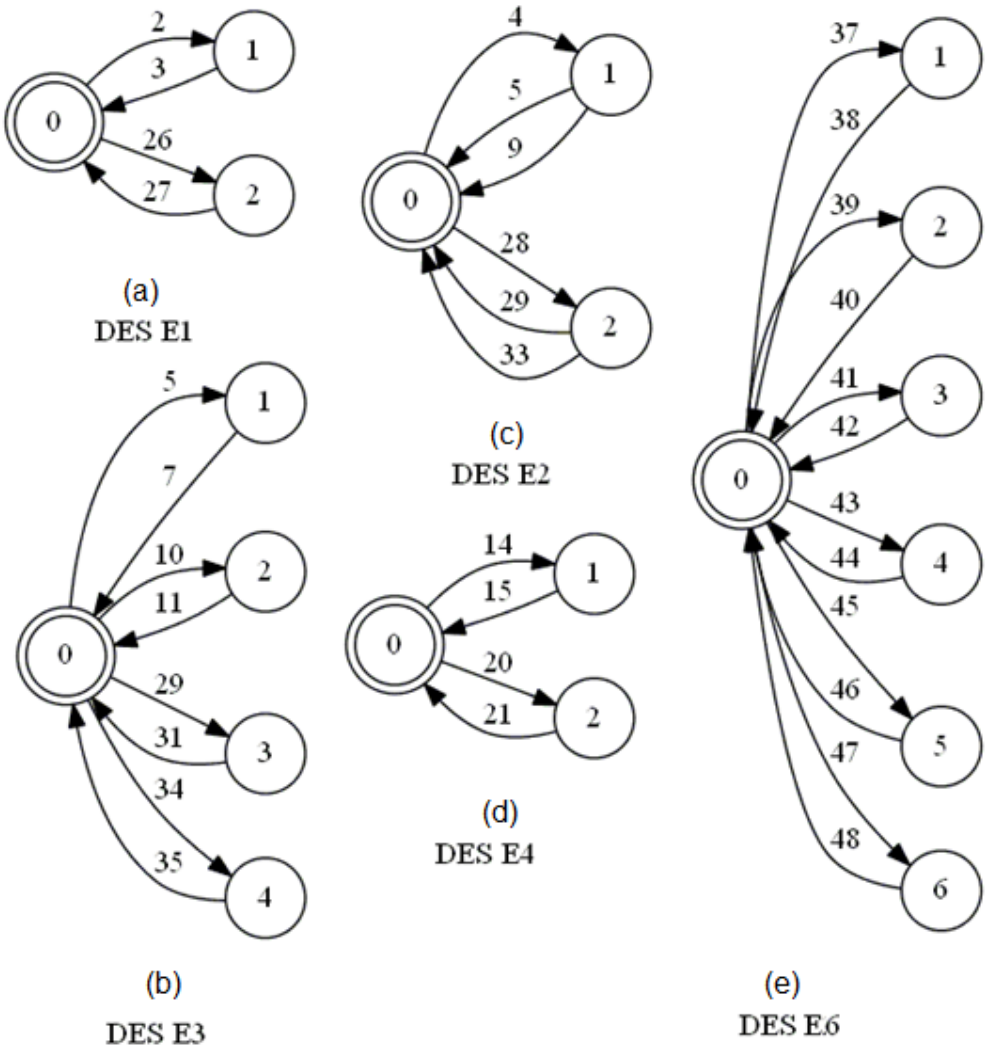

Figura 3.6: Especificações de segurança

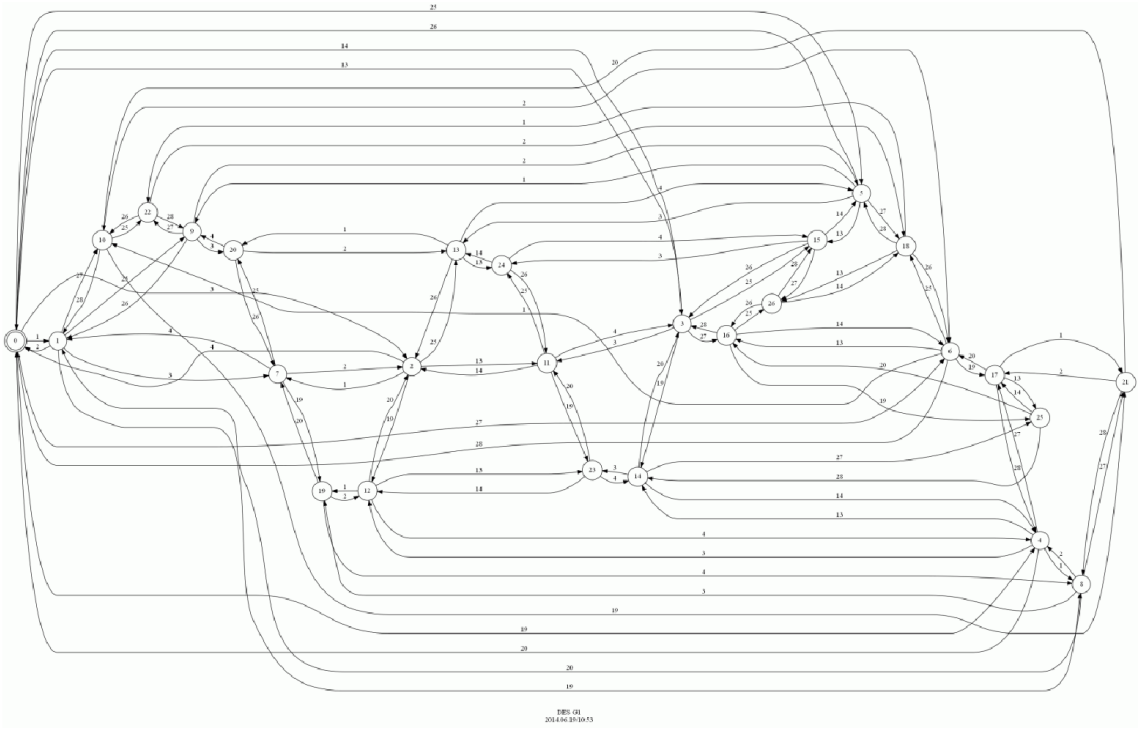

Figura 3.7: Subsistema G1

3.4 Síntese dos Supervisores

Uma vez definido os subsistemas e suas respectivas restrições, o próximo passo é obter a linguagem desejada para cada subsistema G_i fazendo uma composição síncrona entre o subsistema e a restrição E_i. O resultado pode ser visto abaixo na tabela 3.7.

Tabela 3.7: Obtendo a linguagem desejável

Linguagem Desejada	Composição Síncrona	Nº de estados	Nº de transições
K1	$E1\|\|G1$	81	252
K2	$E2\|\|G2$	135	294
K3	$E3\|\|G3$	945	3540
K4	$E4\|\|G4$	81	252
K5	$E5\|\|G5$	189	666
K6	$Robô\ 2\ ou\ G6$	7	12

Para cada linguagem desejada $K(= E_j||G_j)$ ser controlável, deve existir um supervisor não bloqueante tal que $L_m(S/G) = K$. Em outras palavras, $K_j = S_j$. Mas, nem sempre é possível afirmar que K_j atende as condições de controlabilidade. Deve-se então calcular uma linguagem controlável que se aproxime de K_j que não tenha comportamento indesejável. Isto

pode ser feito calculando $SupC(K_j, G_j)$. Feitas estas considerações, tem-se na tabela 3.8 o cálculo do $SupC(K_j, G_j)$ para cada subsistema.

Tabela 3.8: Síntese dos supervisores

Supervisores	$SupC(K_j, G_j)$	Nº de estados	Nº de transições
S1	$SupC(K_1, G_1)$	48	148
S2	$SupC(K_2, G_2)$	75	146
S3	$SupC(K_3, G_3)$	441	1632
S4	$SupC(K_4, G_4)$	48	148
S5	$SupC(K_5, G_5)$	105	352

Para garantir que o sistema seja não bloqueante, os supervisores devem ser modulares, ou seja, verifica-se a igualdade abaixo:

$$C_1 = \overline{S_1}||\overline{S_2}||\overline{S_3}||\overline{S_4}||\overline{S_5} \; e \; C_2 = \overline{S_1||S_2||S_3||S_4||S_5} \qquad (3.2)$$

$$C_1 = C_2$$

Para C_1 obteve-se 27195 estados e 98806 transições e para C_2 27195 estados e 98806 transições. Logo, os supervisores são modulares.

Uma vez obtidos os supervisores, é necessário aplicar uma operação de redução sobre eles. Segundo Queiroz e Cury (2002), para implementação do supervisor em controladores lógicos programáveis, é preciso levar em conta os recursos deste equipamento, que em geral são limitados. Supervisores com muitas estados são de difícil visualização, logo o trabalho do programador fica comprometido. Dessa forma, a redução de supervisores deve ser empregada. O algoritmo de minimização é formalizado por Vaz e Wonham (1986 apud, QUEIROZ; CURY; 2002). Este algoritmo sugere que o supervisor original seja agrupado em blocos. Assim, as debilitações dos eventos de um bloco não devem entrar em conflito com os eventos habilitados em outros estados deste bloco. Além disso, o supervisor reduzido deve possuir a estrutura de transição entre os blocos determinística. É importante ressaltar que nem sempre é possível reduzir os supervisores, pois esta operação é considerada de complexidade exponencial ao longo do tempo (QUEIROZ; CURY, 2002).

O resultado obtido, através do TCT (WONHAN, 2014), de redução de supervisores pode ser visto na tabela 3.9 e na figura 3.8.

Tabela 3.9: Supervisores reduzidos

Supervisores Reduzidos	$Sred(G_{j,}S_{j,}DAT_{j,})$	Nº de estados	Nº de transições
S1red	$Sred(G_{1,}S_{1,}DAT_{1})$	2	4
S2red	$Sred(G_{2,}S_{2,}DAT_{2})$	2	8
S3red	$Sred(G_{3,}S_{3,}DAT_{3})$	2	8
S4red	$Sred(G_{4,}S_{4,}DAT_{4})$	2	4
S5red	$Sred(G_{5,}S_{5,}DAT_{5})$	2	7

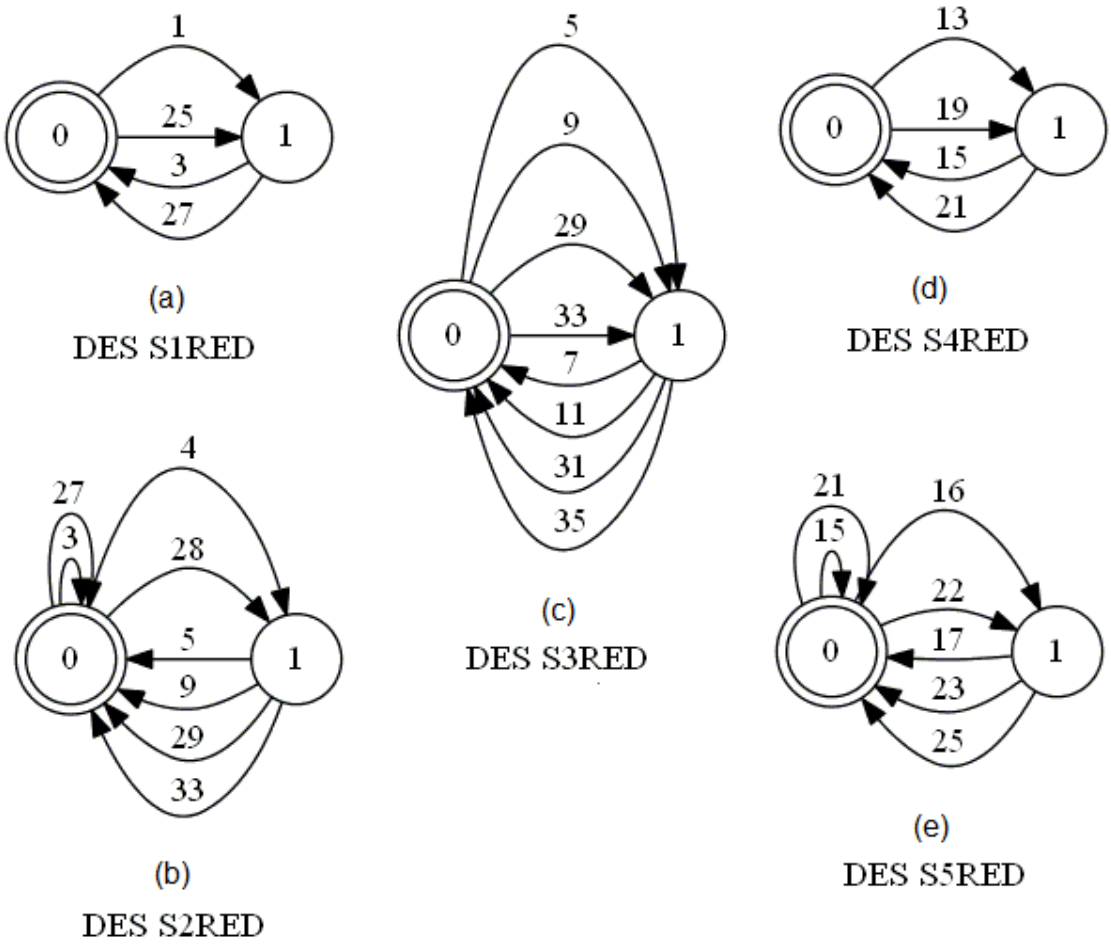

Figura 3.8: Supervisores reduzidos

4 SIMULADOR

Com a proposta de aplicar o controle supervisório neste trabalho, foi projetado, em C#, um simulador que descreve o comportamento do sistema flexível de manufatura em questão sob ação dos supervisores projetados para este fim.

O simulador deste projeto permite a visualização do comportamento de todo o processo de produção das peças, respeitando as restrições impostas, como também exemplificando a interação entre os componentes da planta.

A interface do simulador pode ser vista na figura 4.1.

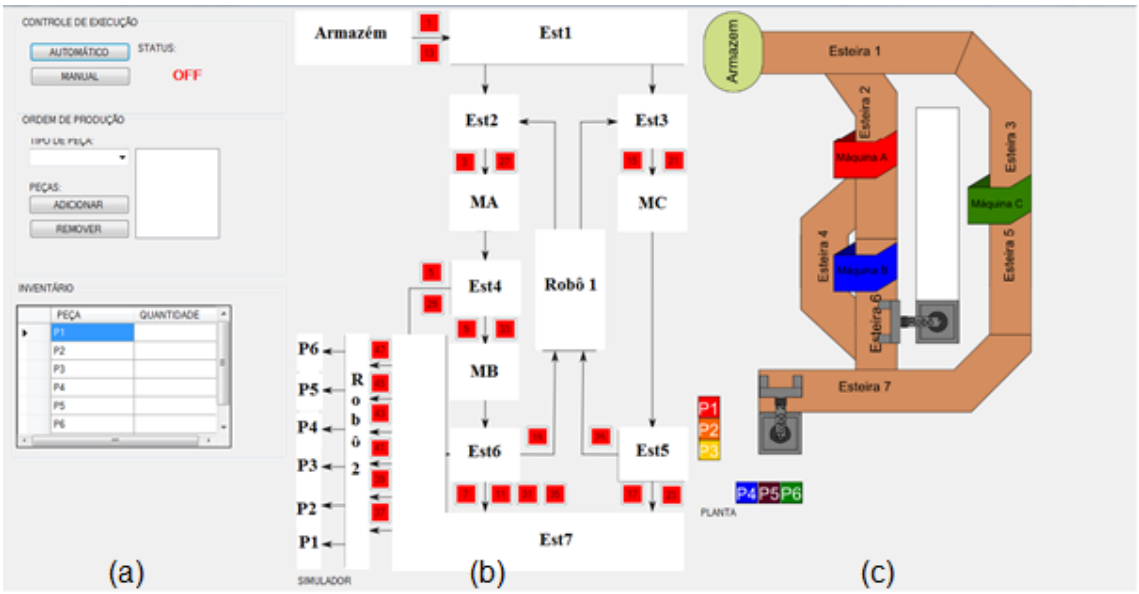

Figura 4.1: Interface do simulador

O sistema pode operar em modo manual ou automático. Quando em manual, é possível escolher quais botões (de 1 ao 47) podem ser pressionados. Estes botões representam os eventos controláveis da planta. De acordo com a ordem de escolha destes, é possível determinar qual peça será produzida. Um ponto importante é que apenas estarão habilitados os botões cujo evento correspondente esteja habilitado pelo supervisor. Por exemplo, se a matéria prima se encontra na esteira 2, o único evento possível de acontecer seria o evento 3, que levaria o material para a Máquina A. Logo, o botão 3 estará disponível para ser pressionado. Ainda analisando este exemplo, observa-se que as esteiras 2 e 3 são independentes, ou seja, mesmo que a esteira 2 esteja ocupada, o tapete 3 pode receber material se estiver vazio. Dessa maneira, o botão 13 (evento que leva matéria prima para este tapete) e o botão 19 (evento que leva a peça 2 da esteira 6 para o robô 1 e posteriormente para o tapete 3) estariam disponíveis para serem utilizados, porém este último (botão 19) poderia ser pressionado apenas se a peça 2 estivesse na esteira 6. Os botões, quando disponíveis para serem pressionados, ficam verde; caso contrário, permanecem vermelhos.

No modo automático, é possível produzir peças sem a necessidade de pressionar os botões. Basta selecionar o tipo de peça e adicionar em uma lista localizada em ordem de produção, como mostra a figura 4.1 (a). Neste contexto, é possível adicionar ou remover peças nesta lista, o que serve de parâmetro para a produção das peças desejadas. Porém, neste modo, só é possível fabricar uma peça por vez, diferente do modo manual em que é possível trabalhar várias peças simultaneamente.

É possível observar no inventário, visto na figura 4.1 (a), a quantidade de peças já produzidas. Isto garante o controle de estoque para esta planta.

No figura 4.1 (b) onde se encontram os botões (eventos), há uma representação por blocos dos componentes da planta. Cada bloco representa um equipamento, e quando este se encontra em uso, o bloco fica verde.

A fim de tornar o processo de produção das peças mais visual, há no simulador um modelo da planta, representado na figura 4.1 (c). Por ele é possível observar qual peça está sendo trabalhada. A figura 4.2 mostra a fabricação da peça 6, através deste modelo.

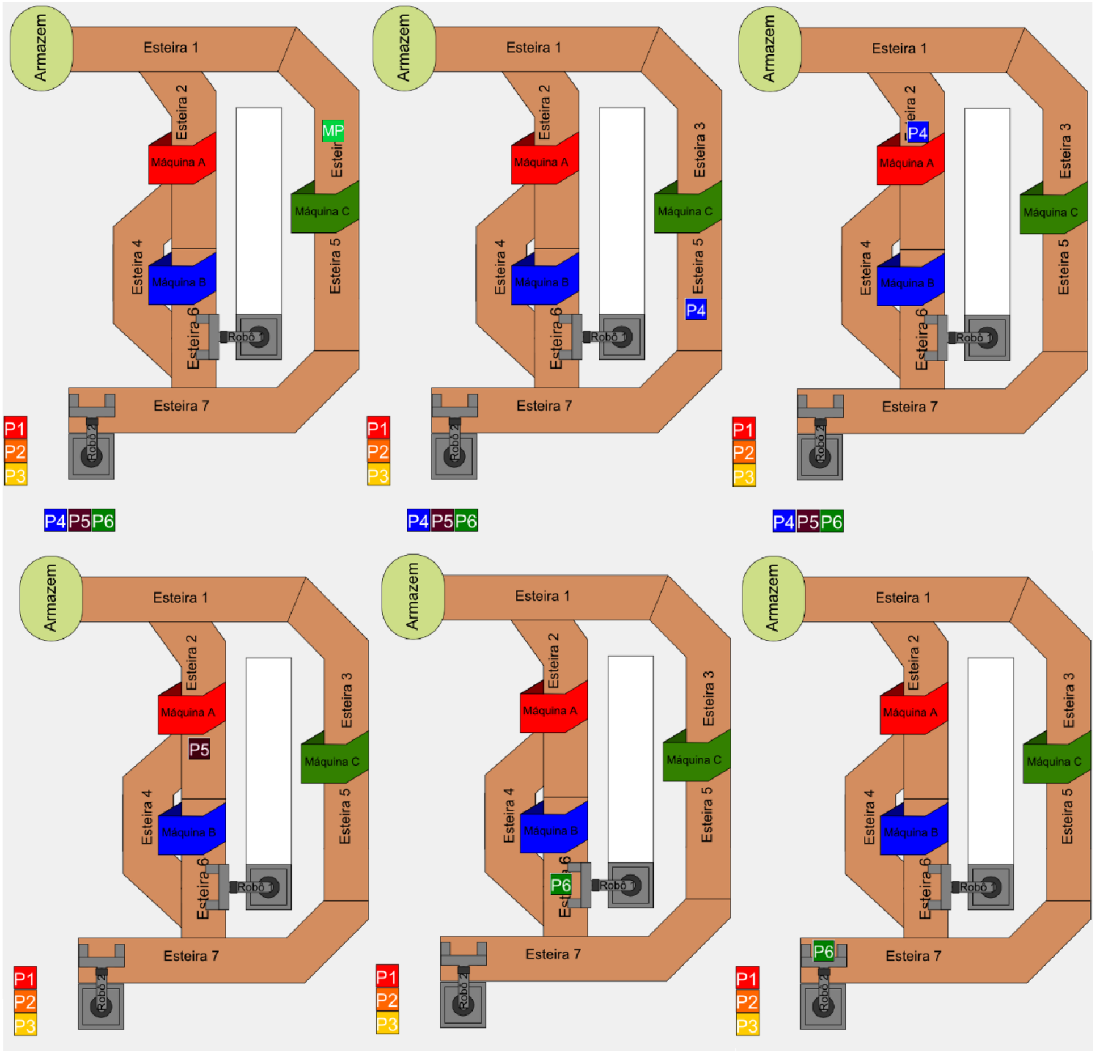

Figura 4.2: Processo da produção de P6

As etapa do desenvolvimento do simulador são descritas no apêndice 1.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto um Sistema Flexível de Manufatura (SFM), desde a sua concepção inicial, passando pela modelagem e controle, até a simulação do sistema através de um software desenvolvido para este fim.

Por se tratar de um sistema a eventos discretos, utilizou-se a abordagem de Linguagens Formais e Autômatos para a modelagem e o Controle Modular Local, uma técnica no escopo da Teoria de Controle Supervisório, para o projeto do controlador (supervisor).

Optou-se pelo Controle Modular Local em detrimento às técnicas clássicas de controle, como, por exemplo, o Controle Monolítico, devido à complexidade da planta, uma vez que o número de estados e eventos dos autômatos (geradores) obtidos a partir da síntese monolítica não é possível de ser tratado computacionalmente.

Para verificar o desempenho do sistema de controle e supervisão, um simulador foi construído com o objetivo não só de observar o funcionamento do sistema, como também para servir de base à construção de uma planta didática a ser desenvolvida em um próximo trabalho.

Por fim, pode-se afirmar que os objetivos do trabalho foram alcançados, uma vez que pôde-se validar o modelo do sistema flexível de manufatura proposto pelo simulador.

Sugere-se, para trabalhos futuros, a construção de um modelo didático desta planta, implementando o sistema de controle e supervisão a partir de um Controlador Lógico Programável (CLP). Com isto, uma nova plataforma ficaria disponível tanto para disciplinas quanto para desenvolvimento de pesquisas do curso de Engenharia de Controle e Automação da UFOP.

Além disso, sugere-se também aumentar a funcionalidade do simulador de forma a ser possível fabricar vários tipos de peça simultaneamente no modo de operação automático.

Por fim, sugere-se aplicar técnicas de otimização para auxiliar na escolha da melhor sequência de produção a fim de se obter um sistema mais eficiente.

REFERÊNCIAS

BANKS, Jerry; CARSON, John S.; NELSON, Barry L.; NICOL, David M. **Discrete event system simulation.** 4.ed. Upper Saddle River, NJ: Prentice-Hall, 2005.

BOIMOND. J.L. **Sur l'étude des sustèmes à événements discrets dans l'algèbre des dioïdes: Identification, comande des graphes d'événements temporisés, représentation des graphes, d'événements temporisés à paramètres variables**. l'Université d' Angers, 1999.

CAO, X.R; HO, Y.C. Models of discrete event dynamics systems. **IEEE Control System Magazine 10(4),** p.69-76, 1990.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. N.J, USA: Springer Sciences+Business Media, LCC, 2008.

CHEN, Y.-L.; LAFORTUNE, S.; LIN, F. Modular supervisory control with priorities for discrete event systems. In: **Proceedings of the 34th IEEE Conference On Decision and Control**, pp. 409-415, December 1995.

CHO, S. A distributed time driven simulation method for enabling real time manufacturing shop flor control. **Computer & Industrial Engineering**, p.572–590. 2005.

COSTA, R. F. da S. **Abordagem sistemática para avaliação econômica de cenários para modelos de simulação discreta em manufatura**. 2010. 136f. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Itajubá, Itajubá-MG, p. 14-18. 2010.

CURY, J. E. R**.. Teoria de Controle Supervisório de Sistemas a Eventos Discretos. C**anela-RS**:** Minicurso do V Simpósio Brasileiro de Automação Inteligente**.** 2001.

GARZA-REYES, J. A.; ELDRIDGE, S.; BARBER, K. D.; SORIANO-MEIER, H. Overall equipment effectiveness (OEE) and process capability (PC) measures: a relationship analysis. **International Journal of Quality & Reliability Management**, v.27, n.1, p. 48-62, 2010.

GRILL, A. **Introduction to the Theory of Finite-State Machines.** McGraw-Hill Electronic Sciences Series. McGRAW-HILL Book Company. 1962.

GUABERT, S. **Théorie des Systèmes Linéaires dans les dioïdes**. 1992. 302f. Tese (Doutorado) - École Nationale Supérieure des Mines de Paris, Paris. 1992.

HOPCROFT, J. E; ULLMAN, J. D. **Formal Languages and Their Relation to Automata.** Assison-Wesly Poblishing Company. 1969.

HOPCROFT, J. E; ULLMAN, J. D; MOTWANI, R. **Introdução à Teoria dos Autômatos, Linguagens e Computação.** Editora Campus, 2002.

KLEINROCK, L**. Queueing Systems, Volume I: Theory**. John Wiley & Sons, Canada, 1975.

KROGH, B. H.; HOLLOWAY, L. E. Synthesis of Feedback Control Logic for Discrete Manufacturing Systems**. Automatica**, v. 27, n. 4, p. 641-651. 1991.

LIBEAUT, L. **Sur l'utilisation des Dioïdes pour la Commande des Systèmes a Evénements Discrets**. Tese (Doutorado) — École Doctorale Sciences pour L'Ingenieur de Nantes. 1996.

L'HER, D. **Modélisation du GRAFCET Temporisé et Vérification de Propriétés Temporelles**. 1997. 219f. Tese (Doutorado) - École Doctorale Sciences pour L'Ingenieur de Nantes. 1997.

MENEZES, P. B. **Linguagens Formais e Autômatos.** Editora Sagra Luzzatto, Série de Livros Didáticos, n. 3, ed. 3, 2000. p. 1-80.

MILNER, R. **A Calculus of Comunicating Systems**. New York: Springer Verlag, , 1980.

MONTGOMERY, E. **Introdução a eventos discretos e à teoria de controle supervisório**. Rio de Janeiro: Altas Books Ltda. 2004.

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541-580. 1989.

NOURELFATH, M. and E. NIEL. Modular supervisory control of an experimental automated manufacturing system. **Control Engineering Practice**, vol. 12, p. 205-216. 2004.

OGATA, K. **Modern Control Engineering.** 9 ed. New Jersey: Prentice Hall, 2002. 964 p.

PINCHINAT, S.; MARCHAND, H.; BORGNE, M. L. **Symbolic abstractions of automata and their application to the supervisory control problem**. 1999.

PUTERMAN, M. L. Markov Decision Process. John Wiley and Sons. 1994.

QUEIROZ, M. H. **Controle Supervisório Modular e Multitarefa de Sistemas Compostos.** 2004. 150f. Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Florianópolis, 2004.

QUEIROZ, M. H. e CURY, J. E. R. Controle Supervisório Modular de Sistemas de Manufatura. **Revista Controle & Automação**, p. 123–133. 2002.

QUEIROZ, M. H. D.; CURY, J. E. R. Modular supervisory control of large scale discrete event systems. **Discrete Event Systems: Analysis and Control. Proc. WODES'00: Kluwer Academic**, 2000. p. 103–110.

RAMADGE, P.JG; WONHAM, W. M.. Supervision of discrete event processes. **Proceding of 21st Conference on Decision and Control**. 1982.

RAMADGE, P.JG; WONHAM, W. M.. On the supremal controllable sublanguege of a given languege**. SIAM Journal of Control and Optimization**, 25(3):637-659. 1987.

RAMADGE, P.JG; WONHAM. The control of discrete event systems. **Proceedings IEEE, Special Issue on Discrete Event Dynamic Systems**, v. 77, n. 1, pp. 81-98, Jan. 1989

SALOMAA, A. **Formal Languages.** ACM Monograph Series**.** Academic Press, Inc. New York. 1973.

SARGENT, R. G. Verification and validation of simulation models. Austin, TX, USA: **WINTER SIMULATION CONFERENCE**, 2009.

VAZ, A. F.; WONHAM, W. M. On supervisor reduction in discrete-event systems. **International Journal of Contro**l, v. 44, n. 2, pp. 475-491, 1986.

WONHAM, W. M. Research on Supervisory Control of Discrete-Event Systems. **Desing Software: TCT.** Dept. of Electrical & Computer Engineering, University of Toronto, 2014. Disponível em: < http://www.control.utoronto.ca/DES>. Acesso em: 07 de jun. de 2014.

WONG, K. C.; THISTLE, J. G.; HOANG, H.-H.; MALHAMÉ, R. P. Conflict resolution in modular control with application to feature interaction. In: **Proceedings of the 34th IEEE Conference on Decision and Control**, pp. 416-421, Dec. 1995.

WONG, K.C.; WONHAM W.M. Modular Control and Coordination of Discrete-Event Systems. **Discrete Event Dynamic Systems**, v. 8, n. 3, pp. 247-297, Oct. 1998.

WONHAM, W. M.; RAMADGE, P. J. Modular supervisory control of discrete event systems. **Mathematics of control of discrete event systems**, v. 1, n. 1, pp. 13-30, 1988.

ZHOU, M. C.; VENKATESH, K. Modeling, Simulation and Control of Manufacturing Systems: A Petri Net Approach. **World Scientific**, Singapore, 1999.

ZUKIN, M.; DALCOL, P. R. Um estudo empírico sobre a correlação entre automação flexível e flexibilidade de manufatura. **Rev. Produção**, Rio Grande do Sul, v. 10, n. 2, maio. 2001.

ÇINLAIR, E. **Introduction to Stochastic Processes.** Englewood Cliffs, N.J, USA: Prentice-Hall, 1975.

APÊNDICES

APÊNDICE A – Desenvolvimento do software de simulação

A linguagem de programação utilizada para o desenvolvimento do simular foi o C#, através da plataforma Visual C# 2013. O desenvolvimento seguiu as seguintes etapas:

- Criação da interface;
- Implementação de uma lista encadeada;
- Desenvolvimento do programa.

A.1 Criação da interface

A implementação da interface foi feita no Visual C#, modo Windows form aplication. Desta forma, a própria plataforma fornece um *Form* (janela), bastando o usuário adicionar os recursos (botões, figuras, comboxs, entre outros), que serão utilizados nesta janela, como exemplo na figura A.1.

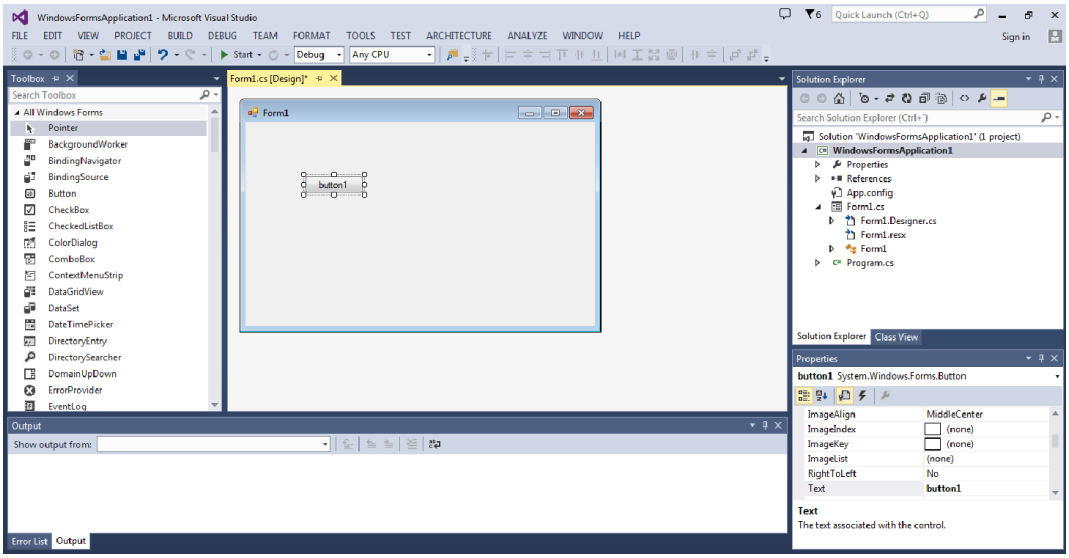

Figura A. 1: Interface do Visual C#

Utilizando-se das facilidades desta ferramenta, construiu-se toda a interface do programa. Um ponto importante é que cada componente da interface possui um conjuntos de propriedades, como cor, tamanho, visibilidade, entre outras, que podem ser facilmente alteradas nas configurações do componente em questão. Estas propriedades também podem ser alteradas via código de programação. Assim, os botões e os blocos relativos a cada equipamento podem ter as suas cores de fundo alteradas durante o funcionamento do programa. A figura A.2 representa a interface final para o simulador.

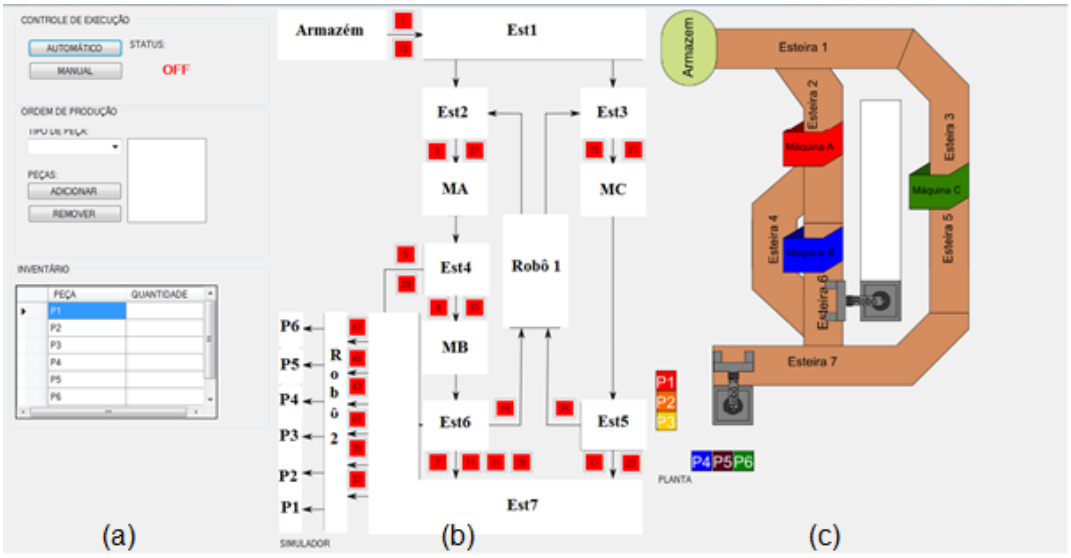

Figura A. 2: Interface do simulador

A.2 Lista Encadeada

Foi utilizada uma lista encadeada para representação dos equipamentos do sistema flexível de manufatura, de forma semelhante aos modelos de autômatos obtidos. Assim, cada equipamento é representado por um conjunto de atributos, definidos a seguir:

- Nome do equipamento (String);
- Estado de uso do equipamento (Boolean);
- Material operado pelo equipamento (String);
- Equipamento a direita (Object);
- Equipamento à esquerda (Object).

A lista foi implementada utilizando três classes. A primeira, chamada de esteira, onde se declara e inicializa os atributos. A segunda (fábrica) responsável pelo gerenciamento da lista, ou seja, adicionar, remover e pesquisar itens. Por fim, a terceira (esteiras), necessária para que a estrutura de dados funcione corretamente.

Desenvolvimento do programa

Utilizando as listas encadeadas, foi criada uma função responsável por observar os estados de cada equipamento. Esta função (condições()) é executada por um processo paralelo (Thread) a sequência do programa original, sendo inicializada no inicio do programa e opera durante todo o tempo.

Basicamente, verifica-se se determinado botão (evento) deve ser habilitado ou desabilitado. Isto se faz com a análise dos atributos da lista. Por exemplo, se houver um material na esteira 2, o atributo que indica que esta esteira está em uso estará verdadeiro. Isto evita que os botões

que enviam peça para este tapete fiquem habilitados. O atributo “material” indica qual botão deverá ficar habilitado para a saída de peças das esteiras. Ainda neste exemplo, se a esteira 2 estiver com matéria prima, este atributo será “MP” (matéria prima) e o botão 3 ficará habilitado. Se o tapete 2 estiver operando a peça 4, o atributo “material” será P4 e o botão 27 estará habilitado. Porém, antes de fazer essas habilitações de botões, é preciso verificar se os equipamentos posteriores aos que estão em uso estão ativos ou não, para assim poder liberar eventos (botões) que enviam peças para estes.

Cada botão possui um método, que será executado quando ocorrer um clique neste. Dentro de cada método associado ao clique do botão, atualizam-se os atributos da lista encadeada. Os atributos “equipamento à direita” ou “à esquerda” representam o componente subsequente ao equipamento em uso. Dentro desses métodos, ainda há a execução de processo em background (thread background). Este processo (setcolor()) executa uma função responsável por mudar a propriedade de cor do bloco relativo ao equipamento em uso, sendo verde para ativo e vermelho para inativo. Além disso, essa função atualiza a posição das peças na planta da figura A.2 (b).

O algoritmo responsável por todo o funcionamento do simulador da planta pode ser descrito pelo fluxograma da figura A.3.

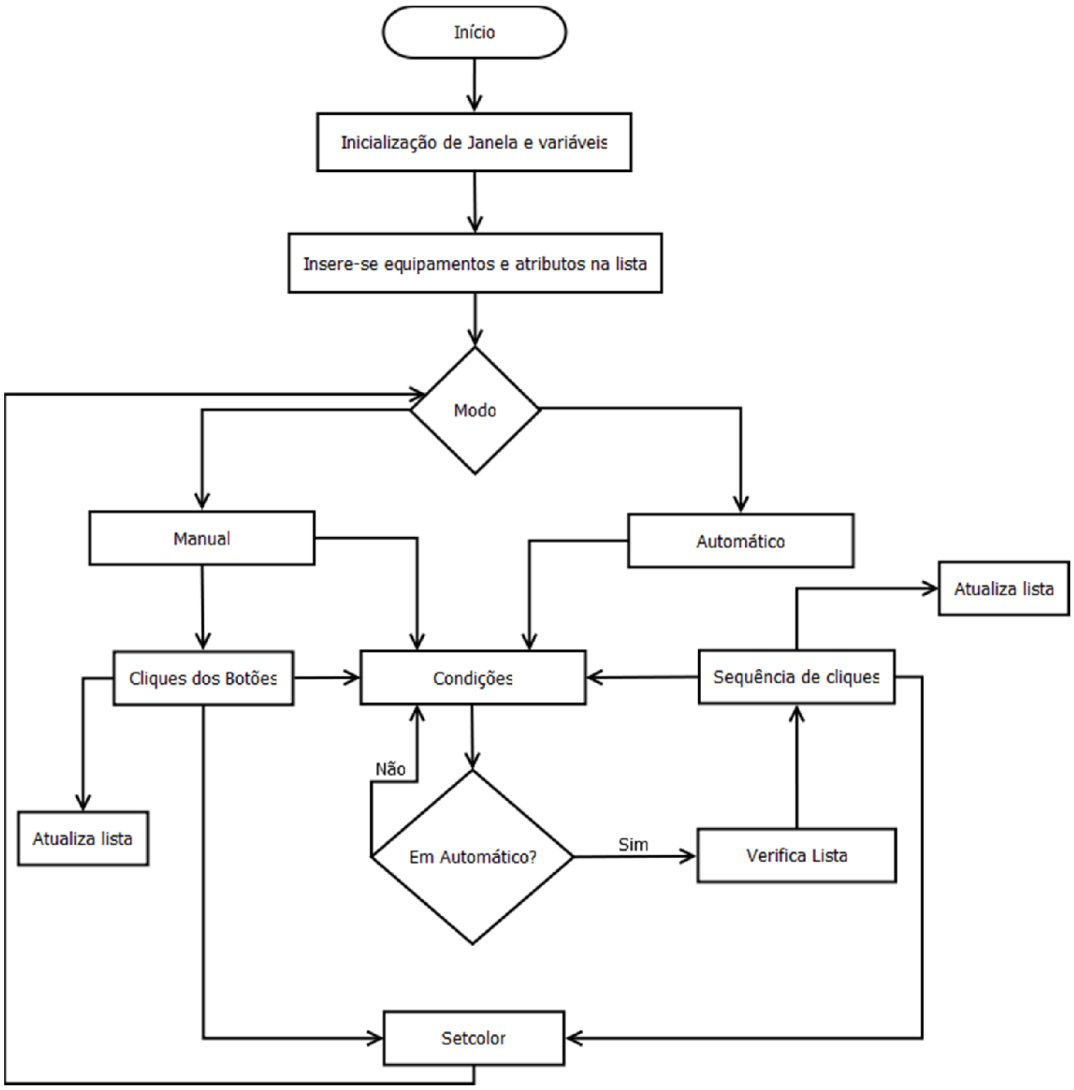

Figura A. 3: Fluxograma do algoritmo do simulador