UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

WEVERTON BUENO DA SILVA

TEORIA DOS JOGOS APLICADA À TOMADA DE DECISÕES EM PROCESSOS FLEXÍVEIS DE MANUFATURA

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2018

WEVERTON BUENO DA SILVA

TEORIA DOS JOGOS APLICADA À TOMADA DE DECISÕES EM PROCESSOS FLEXÍVEIS DE MANUFATURA

Trabalho de Conclusão de Curso de graduação, apresentado a disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia da Computação do Departamento Acadêmico de Informática - DAINF - da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro de Computação.

Orientador: Prof. Dr. Marcelo Teixeira

PATO BRANCO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 15 horas e 50 minutos do dia 22 de junho de 2018, na sala V108, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Marcelo Teixeira (orientador), Dalcimar Casanova e Jean Paulo Martins para avaliar o trabalho de conclusão de curso com o título **Teoria dos Jogos aplicada à tomada de decisões em processos flexíveis de manufatura**, do aluno **Weverton Bueno da Silva**, matrícula 01436813, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Marcelo Teixeira
Orientador (UTFPR)

_____
Dalcimar Casanova
(UTFPR)

_____
Jean Paulo Martins
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

SILVA, Weverton Bueno. Teoria dos Jogos Aplicada à Tomada de Decisões em Processos Flexíveis de Manufatura. 2018. 48f. Trabalho de Conclusão de Curso 2 - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

A *Teoria de Controle Supervisório* (TCS) é uma abordagem que permite a síntese automática de controladores para processos industriais, que se comportam de maneira robusta, sem bloqueios e de acordo com um conjunto de requisitos pré especificados. Apesar de sua solidez teórica, a TCS diretamente não atribui flexibilidade aos sistemas de controle resultantes. A TCS habilita, em cada estado, um conjunto de eventos, mas não permite decidir sobre as escolhas e prioridades dentro desse conjunto possível. Como essa característica predomina e dirige os sistemas modernos de manufatura flexível, passa a ser necessário enriquecer a TCS com novas estruturas que permitam lidar com tais características, tornando-a assim aplicável ao atual conceito de controle industrial. Este trabalho se propõe a abordar um processo flexível de manufatura como um jogo por turnos. Dessa forma, serão utilizadas técnicas e estruturas provenientes da *Teoria dos Jogos* como método para tomada de decisões sobre processos flexíveis, possibilitando a alteração da sequência de produção dinamicamente.

Palavras-chave: Teoria de Jogos, Controle Supervisório, Sistemas a eventos discretos, Sistema de manufatura flexível.

ABSTRACT

SILVA, Weverton Bueno. Game Theory Applied to Decision Making in Flexible Manufacturing Processes. 2018. 48f. Trabalho de Conclusão de Curso 2 - Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2018.

Supervisory Control Theory (SCT) is an approach that allows to automatically synthesize controllers for industrial processes, which behave robustly, without blockings and according to a set of pre-specified requirements. Despite its theoretical formulation, the SCT directly does not assign flexibility to the resulting control systems. The SCT enables, at each state, a set of possible events, but does not allows to decide about choices and priorities inside this set. As such characteristic predominates and guides modern flexible manufacturing systems, it is necessary to enrich the SCT with new structures that allow addressing it, thus making it applicable to the current concept of industrial control. This paper approaches a flexible manufacturing process as a turn-based game. In this way, techniques and structures from the *Game Theory* are used as a method for decision-making on flexible processes, allowing to change the production sequence dynamically.

Keywords: Game Theory, Supervisory Control, Discrete-Event Systems, Flexible Manufacturing System.

LISTA DE FIGURAS

Figura 1: Porta automática 14
Figura 2: Composição síncrona entre duas máquinas 18
Figura 3: Modelo de uma restrição 18
Figura 4: Modelo restringido por E 19
Figura 5: Pedra-Papel-Tesoura com informação completa 23
Figura 6: Sistema de Manufatura Flexível 29
Figura 7: Plantas do Sistema 1 30
Figura 8: Buffer de duas posições do Sistema 1 30
Figura 9: Especificação E_1 do Sistema 1 31
Figura 10: Arvore-Jogo do Sistema 1 32
Figura 11: Início do Sistema 1 sem tomada de decisão 33
Figura 12: Especificações do Sistema 1 após execução de A 33
Figura 13: S esperando M_1 retirar uma peça do *buffer* para realizar A 33
Figura 14: Árvore-jogo reduzida e com recompensas. 34
Figura 15: Plantas do Sistema 2 35
Figura 16: Buffer de duas posições do Sistema 2 35
Figura 17: Especificação para M_1 do Sistema 2 36
Figura 18: Arvore-Jogo do Sistema 2 37
Figura 19: Árvore-Jogo com recompensas para o Sistema 2 38
Figura 20: Supervisor sem TJ do Sistema 1 45

Figura 21: Supervisor com auxilio da TJ do Sistema 1 46

SUMÁRIO

1 INTRODUÇÃO 8

1.1 CONSIDERAÇÕES INICIAIS 8

1.2 PROBLEMÁTICA E JUSTIFICATIVA 10

1.3 OBJETIVOS 11

1.3.1 Objetivo Geral 11

1.3.2 Objetivos Específicos 11

2 REFERENCIAL TEÓRICO 12

2.1 SISTEMAS A EVENTOS DISCRETOS 12

2.1.1 Autômatos Finitos e Linguagens 13

2.1.2 Operações sobre Autômatos 16

2.1.3 Modelagem de SED por Autômatos 17

2.2 TEORIA DO CONTROLE SUPERVISÓRIO 19

2.3 TEORIA DE JOGOS 21

2.3.1 Tomada de decisão 24

3 APLICAÇÃO EM SISTEMA DE MANUFATURA FLEXÍVEL 26

3.1 MÉTODO UTILIZADO 27

3.2 SISTEMA 1 29

3.2.1 Abordagem sem Teoria dos Jogos 30

3.2.2 Abordagem utilizando Teoria dos Jogos 31

3.2.3 Simulação 32

3.3 SISTEMA 2 35

3.3.1 Simulação 37

3.4 COMPARATIVO ENTRE AMBAS ABORDAGENS 39

4 CONCLUSÃO 41

REFERÊNCIAS 43

APÊNDICE A - SUPERVISOR COM SEQUÊNCIA DE EVENTOS 45

APÊNDICE B - SUPERVISOR SEM SEQUÊNCIA DE EVENTOS 46

1 INTRODUÇÃO

Neste capítulo são apresentados os conceitos gerais que motivaram o desenvolvimento deste trabalho. Apresenta-se também os objetivos a serem alcançados e uma justificativa que demonstra a importância do assunto proposto.

1.1 CONSIDERAÇÕES INICIAIS

Ao avançar do tempo e com a automatização de tarefas antes atribuídas aos seres humanos, sistemas que utilizam multi-robôs e máquinas concorrentes começaram a surgir. Este avanço é visto principalmente em sistemas de manufatura, onde grande parte das linhas de produção possuem maquinário automatizado. Estes sistemas caracterizam uma classe chamada de *Sistemas a Eventos Discretos* (SEDs). Essa família de sistemas se diferencia por possuir uma dinâmica de evolução que é dirigida pela ocorrência de eventos assíncronos, em oposição a sistemas dirigidos pelo tempo.

Em um SED, algum critério precisa ser atingido (evento) para que a ação no sistema possa ser tomada (evolução). Em uma linha de produção de bebidas, por exemplo, o líquido só pode ser colocado ao se identificar um recipiente vazio. Sendo assim, o sistema evolui na ocorrência de eventos discretos, ou seja, que “ocorrem de forma abrupta e apenas em certos pontos no tempo” (CASSANDRAS; LAFORTUNE, 2007), gerando então o esquema de transição entre estados.

Para fins práticos, o comportamento de um SED, ou dos componentes individuais que compõem um SED, precisa ser restrito a um conjunto de especificações para que, na prática, eles atuem conforme o esperado. É então interessante que seja possível garantir que os eventos ocorram apenas quando os requisitos forem satisfeitos. Uma maneira de garantir tal comportamento é por meio da *Teoria de Controle Supervisório* (TCS) que, de acordo com Ra-

madge e Wonham (1989), “consiste em mudanças na entrada do controlador através de uma sequência de elementos em resposta a uma cadeia de eventos gerada previamente”, ou seja, através do conjunto de eventos possíveis, é adotado um conjunto de especificações que determinarão uma, ou várias, sequências de eventos válidas, garantindo dessa forma uma resposta ótima para SEDs.

Ao se utilizar a TCS o sistema é modelado através de autômatos, cuja composição recebe o nome de planta. Por sua vez, as restrições desejadas sobre as plantas do sistema são modeladas através de especificações. Sobre a composição da planta com as especificações, uma operação matemática é usada para sintetizar um supervisor ótimo, ou seja, minimamente restritivo e não bloqueante, que satisfaz as especificações. Apesar de muito útil para realizar o controle sobre o sistema, o supervisor não garante que apenas um evento será habilitado por estado. Dessa forma, em casos onde existe mais de um evento possível por estado, torna-se necessário decidir qual evento, entre o conjunto de eventos possíveis, será realizado primeiro, em função de alguma política de planejamento e aproveitamento do processo.

Por exemplo, em um *Sistema de Manufatura Flexível* (SMF), é esperado que as ações do sistema sejam influenciadas pelas mudanças do ambiente, podendo assumir ações e comportamentos diversos para diferentes tipos de situações. Pode ser de interesse que o próprio sistema analise a situação e decida por conta própria a melhor maneira de agir. Uma teoria bastante - conhecida e difundida para tomada de decisões em problemas matemáticos e probabilísticos é a *Teoria dos Jogos*.

A Teoria dos Jogos, assim como escrito por Griffin (2010), “é uma disciplina dentro da Matemática Aplicada que influencia e é influenciada por áreas de Pesquisas Operacionais, Economia, Teoria de Controle, Ciência da Computação, entre outras”. Ainda de acordo com Griffin (2010) a teoria pode ser classificada em quatro grandes subcategorias de estudo, que são: Clássica, Combinacional, Dinâmica e um conjunto de tópicos derivados das três primei-

ras, como por exemplo a Teoria dos Jogos Evolucionária.

A Teoria dos Jogos pode ser definida como o estudo de modelos matemáticos de conflito e cooperação entre tomadores de decisão inteligentes e racionais (MYERSON, 1997). Consiste em visualizar o problema como um jogo, onde existem as jogadas, que são as ações possíveis de serem realizadas, e os jogadores, que são aqueles que realizam as jogadas.

A Teoria dos Jogos Clássica é aquela que busca jogadas ótimas em situações onde os impactos de cada jogada são conhecidos, assim como as decisões daqueles envolvidos, utilizando métodos randômicos e probabilísticos para a tomada de decisões.

Diferentemente da Clássica, a Teoria dos Jogos Combinatória não utiliza de métodos randômicos, dessa forma, para se obter a jogada ótima, é realizada a análise de todas as combinações de jogadas possíveis. Funciona baseada em turnos, ou seja, cada jogador tem sua vez de agir, sendo então necessário um mínimo de dois jogadores, similar a um jogo de xadrez.

A Teoria em sua vertente Dinâmica utiliza de equações diferenciais para a análise de jogadas que devem ser feitas ao longo do tempo e que afetam as possibilidades do próximo instante de tempo.

1.2 PROBLEMÁTICA E JUSTIFICATIVA

Ao se utilizar da TCS, o supervisor obtido desabilita eventos que são controláveis para evitar a ocorrência de comportamentos ilegais pela planta. Contudo não é selecionado, dentre o conjunto de eventos habilitados, qual deve ser executado primeiro ou em qual ordem estes devem ser executados (VILELA; PENA, 2016). Para resolver essa situação normalmente é utilizada uma sequência de prioridades em que, caso haja mais de um evento possível, o sistema deve seguir a sequência de forma que não seja necessária a tomada de decisão. Porém, dessa maneira, restringem-se as opções possíveis sobre a planta.

Ao se trabalhar com máquinas flexíveis, que disputam ações em um mesmo ambiente para a realização de um (ou vários) objetivo, é interessante que haja cooperação, de forma que ambas trabalhem em conjunto para atingir um objetivo geral ou, então, seus próprios objetivos individuais. Esse tipo de situação pode ser facilmente visualizada como um sistema de jogo, em que as máquinas são os jogadores e os eventos são as jogadas a serem consideradas. Dessa forma, torna-se intuitivo pensar que a Teoria dos Jogos possa ser integrada a TCS e aplicada em problemas de controle de manufatura.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Este trabalho se propõe a integrar técnicas advindas da Teoria dos Jogos na Teoria do Controle Supervisório de Sistemas a Eventos Discretos, como forma de auxiliar o sistema de controle no problema de tomada de decisões em processos flexíveis de manufatura.

1.3.2 OBJETIVOS ESPECÍFICOS

- Fazer um levantamento das principais técnicas relacionadas à Teoria dos Jogos e suas aplicações em processos de tomada de decisões.
- Integrar técnicas da Teoria dos Jogos ao controle supervisório de SEDs.
- Realizar um estudo de caso sobre um processo flexível de manufatura onde é possível fabricar dois tipos de peças distintas em uma mesma linha de produção.
- Comparar a tomada de decisão realizada por um controlador sem e com o suporte da proposta.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos sobre *Sistemas a Eventos Discretos* (SEDs) e sua representação formal usando *Autômatos Finitos* e *Linguagens*. Em seguida é apresentada a *Teoria de Controle Supervisório* (TCS) e sua aplicação sobre SEDs. Por fim, apresentam-se conceitos relacionados à *Teoria dos Jogos*, bem como uma problemática que motiva o uso dessa teoria de modo integrado à TCS. Exemplos são apresentados para ilustrar cada abordagem.

2.1 SISTEMAS A EVENTOS DISCRETOS

De acordo com Ogata (1997), “um sistema é uma combinação de componentes que atuam em conjunto e realizam um certo objetivo”. Esta definição não envolve apenas sistemas físicos e elétricos, podendo também ser aplicada em sistemas biológicos e econômicos, por exemplo.

Todos os tipos de sistemas compartilham o fato de possuírem uma estrutura de *estados* que define a situação do sistema em determinado instante de tempo, e uma estrutura de *transição* de estados. É importante que essas características sejam observadas e identificadas pois através delas é possível modelar o sistema de forma a facilitar o seu posterior controle.

A estrutura de estados e transições de um sistema pode ser representada através do tempo basicamente de duas maneiras: em tempo *contínuo*, como por exemplo sistemas elétricos nos quais as transições se dão no domínio do tempo e, assim, são normalmente representados através de equações diferenciais; e em tempo *discreto*, nos quais as alterações de estados ocorrem de forma assíncrona.

À classe de sistemas que evoluem na ocorrência de eventos discretos no tempo, ou seja, nos quais as transições são consequência de eventos abruptos e que ocorrem apenas em certos pontos no tempo, dá-se o nome

de *Sistema a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2007). Um exemplo de fácil entendimento para um SED pode ser dado através de uma máquina simples que possui os estados *ligada* e *desligada*, porém a mesma só é ativada quando um botão de ligar é pressionado, dessa forma é fácil perceber que o ato de apertar o botão gera um evento, independente no tempo, que informa a máquina que deverá passar do estado *desligada* para o estado *ligada*. Essa mesma lógica pode ser aplicada para o desligamento da máquina.

Diferentemente de um sistema em tempo contínuo, modelar um SED através de equações diferenciais é inadequado, pois a ocorrência de eventos se dá em apenas alguns pontos no tempo e de forma irregular. Dessa forma, a representação deste tipo de sistema é, em sua grande maioria, realizada através de *Autômatos Finitos*.

2.1.1 AUTÔMATOS FINITOS E LINGUAGENS

Um *Autômato Finito* (AF), também conhecido por máquina de estados finitos, é um dos mais simples modelos computacionais existentes e muito utilizado para representar SEDs, pois o mesmo consegue mapear, de maneira natural e intuitiva, o avanço entre os estados do sistema de acordo com a ocorrência de eventos. Para fins práticos, tais eventos podem então ser associados a sinais do sistema real, tornando a abordagem diretamente implementável em sistemas reais.

Formalmente, um AF G pode ser representado por uma quíntupla:

$$G = \langle \Sigma, Q, q^{\circ}, Q^{\omega}, \rightarrow \rangle \tag{1}$$

em que:

- Σ é o alfabeto finito e não vazio de entradas (símbolos ou eventos);
- Q é o conjunto de estados possíveis;
- $q^{\circ} \in Q$ é o estado inicial;

- $Q^{\omega} \subseteq Q$ é o conjunto de estados finais;
- $\rightarrow$ é a função de transição dada por $\rightarrow: Q \times \Sigma \xrightarrow{\sigma} Q$, que define a transição de um estado q_i para um estado q_j, onde q_i, $q_j \in Q$, dado um evento $\sigma \in \Sigma$.

Para que seja possível construir o modelo de um SED por meio de um AF, é necessário definir quais são os estados que o sistema pode assumir, qual é o seu estado inicial, e em quais estados o sistema pode terminar sua operação. Outra ação fundamental é a definição do conjunto de símbolos que representarão os eventos que podem ocorrer no sistema, chamado de *alfabeto*. Por fim, ainda é necessário que os estados possíveis a partir do estado inicial sejam alcançados por meio de transições, perante a ocorrência de eventos.

Para exemplificar um AF vamos utilizar como exemplo uma porta automática, onde existem dois sensores, um na frente e outro atrás, para realizar a abertura/fechamento da porta. Quando qualquer um dos sensores detecta uma pessoa, a porta se abre, e quando nenhuma pessoa é detectada a mesma se fecha, conforme a figura abaixo.

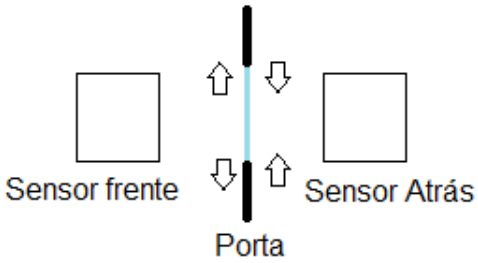


Figura 1: Porta automática.
Fonte: Autoria própria.

Analisando o exemplo nota-se que existem dois estados possíveis, que são: $porta\ aberta$ e $porta\ fechada$. Além disso também é possível visualizar quais são os eventos que realizam a mudança entre esses estados, que são os sinais enviados pelo sensor quando existe uma pessoa na $frente$ da porta, uma pessoa $atras$, quando há uma pessoa em $ambos$ os lados e quando não há $nenhuma$ pessoa. Também é importante notar que a porta estará fechada

quando o sistema for iniciado e que a mesma deverá permanecer fechada caso não haja nenhuma pessoa.

Dessa forma é obtido o seguinte AF:

$$G = \langle \{frente, atras, frente\_atras, nenhum\}, \{aberta, fechada\}, fechada, \{fechada\}, \rightarrow \rangle$$

Em G, $\rightarrow$ é a função que define as transições entre estados, sendo cada uma definida por um par de estados e o evento que faz a transição entre eles. Por exemplo, $fechada \overset{frente}{\rightarrow} aberta$ define a existência de uma transição com o evento $frente$ que leva o sistema da origem $fechada$ ao estado destino $aberta$. Essa representação também pode ser realizada da forma ($fechada$, $frente$, $aberta$), ou como Cassandras e Lafortune (2007) definem, $\rightarrow$($fechada$, $frente$) = $aberta$.

Do ponto de vista formal, algumas propriedades podem ser definidas para autômatos. Por exemplo, a partir do mapeamento entre os estados do modelo, nasce o conceito de *cadeias* finitas de eventos, tal que o conjunto de todas as cadeias possíveis no autômato é representado por Σ^*. Nesse trabalho, a notação $G \overset{s}{\rightarrow} q$ será usada para ilustrar que uma cadeia $s \in \Sigma^*$ é possível em G.

Uma linguagem L sobre Σ, é um subconjunto de cadeias em Σ^*, isto é, $L \subseteq \Sigma^*$, que determina quais são as cadeias possíveis de serem formadas com eventos em Σ em determinado contexto, incluindo a cadeia vazia denotada por ε, a qual representa a sequência com nenhum evento.

Existem dois tipos de linguagens que podem ser estabelecidas a partir de um autômato G, que são a *linguagem gerada* e a *linguagem marcada*, definidas respectivamente por:

$$L(G) = (s \in \Sigma^* \mid G \overset{s}{\rightarrow} q \in Q) \tag{2}$$

e

$$L^{\omega}(G) = (s \in \Sigma^* \mid G \overset{s}{\rightarrow} q \in Q^{\omega}) \tag{3}$$

A linguagem $L(G)$ é o conjunto de todas as cadeias possíveis sobre o autômato, ou seja, nesta linguagem também estão presentes cadeias que, apesar de serem possíveis, não são de interesse. Por exemplo, a cadeia ($frente$, $frente$) do exemplo anterior pertence a essa linguagem, porém a porta continua aberta.

Por sua vez, a linguagem marcada $L^{\omega}(G)$ é um subconjunto de $L(G)$, $L^{\omega}(G) \subseteq L(G)$, que possui apenas as cadeias que levam a um estado marcado $q \subseteq Q^{\omega}$ definido. No exemplo seriam todas as cadeias que terminam com o evento $nenhum$.

2.1.2 OPERAÇÕES SOBRE AUTÔMATOS

Em um SED é comum, e conveniente, se trabalhar com vários autômatos diferentes para representar partes de um sistema maior, possivelmente de maneira mais simples. Dessa forma é imprescindível fazer com que estes autômatos consigam sincronizar seus eventos para que o sistema funcione corretamente. A operação que realiza essa ação é chamada de *composição síncrona* sendo representada por $\|$ e funcionando da seguinte forma:

Sejam dois autômatos $G_1 = \langle \Sigma_1, Q_1, q_1^{\circ}, Q_1^{\omega}, \rightarrow_1 \rangle$ e $G_2 = \langle \Sigma_2, Q_2, q_2^{\circ}, Q_2^{\omega}, \rightarrow_2 \rangle$, a composição síncrona entre G_1 e G_2 é dada por

$$G_1 \| G_2 = \langle\ \Sigma_1 \cup \Sigma_2,\ Q_1 \times Q_2,\ \rightarrow,\ (q_1^{\circ},\ q_2^{\circ}),\ Q_1^{\omega} \times Q_2^{\omega} \rangle$$

tal que a função de transição é definida por:

- $(q_1, q_2) \xrightarrow{\sigma} (q_1^{'}, q'_2)$, se $\sigma \in \Sigma_1 \cap \Sigma_2$
- $(q_1, q_2) \xrightarrow{\sigma} (q_1^{'}, q_2)$, se $\sigma \in \Sigma_1 \backslash \Sigma_2$
- $(q_1, q_2) \xrightarrow{\sigma} (q_1, q'_2)$, se $\sigma \in \Sigma_2 \backslash \Sigma_1$

Ou seja, caso ambos os autômatos habilitem um mesmo evento, a transição entre estados ocorrerá de maneira síncrona nos dois autômatos, tal que as duas transições são fundidas no resultado composto. Caso ocorra um

evento que está habilitado em apenas um dos autômatos, a evolução ocorrerá de forma independente, havendo transição apenas no autômato que reconhece o evento (CASSANDRAS; LAFORTUNE, 2007; CURY, 2001).

Ainda segundo Cury (2001) é possível ressaltar algumas propriedades da composição síncrona, que são:

- $G_1 \| G_2 = G_2 \| G_1$
- $(G_1 \| G_2) \| G_3 = G_1 \| (G_2 \| G_3)$
- A definição pode ser estendida para *n* autômatos.

2.1.3 MODELAGEM DE SED POR AUTÔMATOS

Como já citado, em um SED é comum a representação de um sistema por meio de diversos subsistemas modelados por autômatos, cada um representando uma parcela do sistema total. O autômato que representa o sistema é chamado de *planta*, e analogamente a um sistema de controle em tempo contínuo, caso seja utilizada apenas a composição, o sistema geral terá o comportamento de um sistema em *malha aberta*, a não ser que sejam impostas restrições ou uma ação de controle.

O conjunto das restrições impostas sobre a planta é chamado de *especificação* e representado por E e, assim como a planta, também pode ser representado através de autômatos. Porém, como o próprio nome intuitivamente já impõe, sua função é limitar o funcionamento do sistema de forma que o mesmo haja da forma designada no projeto.

Tome como exemplo dois autômatos, M_1 e M_2, como na Figura 2, que modelam respectivamente o comportamento operacional de 2 máquinas industriais.

M_1 e M_2 possuem apenas dois estados cada e modelam o comportamento de início e fim de operação da respectiva máquina. Quando compostos, esses dois modelos levam a um comportamento que reflete a operação paralela das duas máquinas em $G = M_1 \| M_2$, livre de restrições.

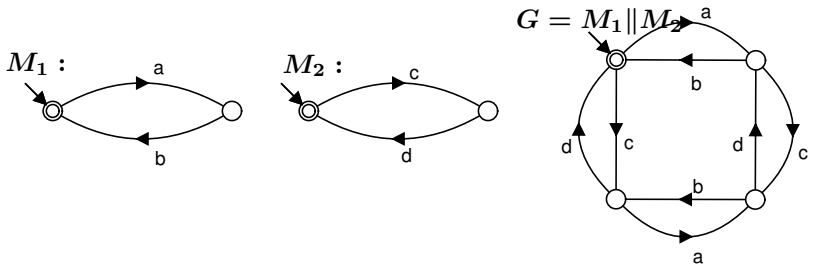


Figura 2: Composição síncrona entre duas máquinas.
Fonte: Adaptado de Teixeira (2013).

Seja então a especificação dada na Figura 3 utilizada para restringir o evento c da máquina M_2 de acontecer no estado inicial de G do exemplo demonstrado na Figura 2, devendo o mesmo acontecer apenas após a máquina M_1 realizar um evento b. Com a adição do auto-laço libera-se a execução de M_1 mesmo se não ocorrer um evento c.

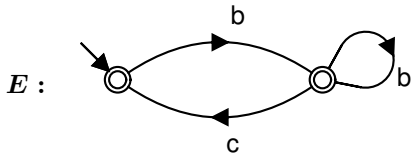


Figura 3: Modelo de uma restrição.
Fonte: Adaptado de Teixeira (2013).

Da mesma maneira que é realizada a modelagem do sistema por meio de subsistemas locais, a especificação também pode ser modelada de forma local através de vários autômatos E_i, que podem posteriormente ser compostos de forma a se obter a especificação geral, i.e.,

$$E = \|_{i=1}^{n} E_i. \tag{4}$$

Após obtida a especificação, o autômato R que representa o sistema restrito pelas especificações pode ser obtido através da composição síncrona de G com E, ou seja, $R = E \| G$. O autômato obtido para o exemplo anterior pode ser visto na Figura 4.

Na prática, R modela o comportamento que se espera de um SED

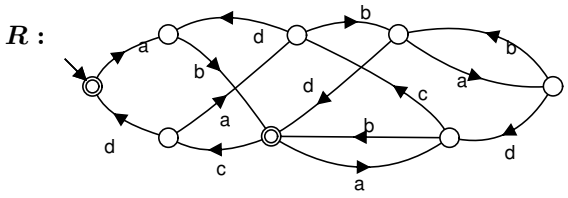


Figura 4: Modelo restringido por E.
Fonte: Autoria própria.

sob controle, partindo-se de modelos individuais, obtidos incrementalmente. Assim, R pode ser convertido em código e implementado como controlador real da planta. O grande empecilho para isso é que, na prática, alguns eventos de um SED podem não ser controláveis, ou seja, embora eles façam parte da planta, eles não podem ser diretamente desabilitados via controle. A recepção de um pacote via rede, ou de um sinal qualquer, por exemplo, ilustra a ideia de eventos sobre os quais não se tem controle direto. Assim, é necessário que, antes de ser implementado, o modelo R passe por uma operação, denominada *síntese*, que *sintetiza* ou extrai, de R uma versão que incorpora a impossibilidade de desabilitar um dado subconjunto de eventos. Uma abordagem formal que estrutura a operação de síntese é apresentada a seguir.

2.2 TEORIA DO CONTROLE SUPERVISÓRIO

A *Teoria de Controle Supervisório* (TCS) (RAMADGE; WONHAM, 1989) define um método matemático que permite incorporar a controlabilidade de eventos na síntese de controladores para SEDs. Para isso, o conjunto de eventos é particionado em $\Sigma = \Sigma_{\mathrm{c}} \cup \Sigma_{\mathrm{u}}$, em que

- Σ_{c} contém os *eventos controláveis*, cuja ocorrência pode ser evitada na planta; e
- Σ_{u} contém os *eventos não-controláveis*, os quais não são diretamente manipuláveis via controlador.

O elemento da TCS que efetivamente implementa as ações de con-

trole na planta, sintetizando a questão da controlabilidade, é o *supervisor*. Um supervisor S é uma estrutura associada à planta G que, após qualquer cadeia $s \in L(G)$, observa os eventos possíveis na planta e informa, dentre eles, quais devem ser habilitados.

O *Problema de Controle Supervisório* (PCS) consiste, então, em obter um supervisor S tal que sua ação sobre G, denotada S/G, satisfaça o conjunto de especificações K, ou seja, $L^{\omega}(S/G) \subseteq K$ e, para isso, desabilite apenas eventos controláveis.

A *controlabilidade* é uma condição necessária e suficiente para a existência de S. Uma linguagem $K \subseteq \Sigma^*$ é *controlável* em relação a uma linguagem prefixo-fechada $L(G)$ se $\overline{K}\Sigma_{\mathrm{u}} \cap L(G) \subseteq \overline{K}$. Ou seja, se após qualquer prefixo s, um evento $\mu \in \Sigma_{\mathrm{u}}$ é elegível em $L(G)$, então μ não é desabilitado, i.e., $s\mu \in \overline{K}$.

Se uma especificação $K \subseteq L(G)$ é controlável em relação a $L(G)$, então a solução de controle pode ser implementada por um autômato V que representa K, para $K = L^{\omega}(V) \cap L^{\omega}(G)$ e $\overline{K} = L(V) \cap L(G)$. Se K não for controlável, se faz necessário reduzi-la à sua *máxima sublinguagem controlável*.

$$\sup\mathcal{C}(K, G) = \bigcup\{K' \subseteq K \mid \text{K' é controlável em relação a } L(G)\}.$$

O processo de computar $\sup\mathcal{C}(K, G)$ é conhecido como *síntese* (CASSANDRAS; LAFORTUNE, 2007) e nada mais é do que uma operação matemática que implementa a noção de controlabilidade, i.e., o algoritmo de síntese extrai de K um sub-modelo K' que trata da impossibilidade de interferir diretamente em eventos de Σ_{u}.

Assim, $\sup\mathcal{C}(K, G)$ representa o comportamento menos restritivo possível de ser implementado por um supervisor S ao controlar G, respeitando a especificação K. Se $\sup\mathcal{C}(K, G)$ for ainda não-bloqueante, então $L^{\omega}(S/G) = \sup\mathcal{C}(K, G)$ e $L^{\omega}(S/G) = \overline{\sup\mathcal{C}(K, G)}$ é uma solução *ótima* para o PCS.

2.3 TEORIA DE JOGOS

A *Teoria de Jogos* (TJ) é uma área da matemática aplicada que estuda modelos de conflitos e cooperação entre decisores inteligentes e racionais (MYERSON, 1997). Essa teoria pode ser aplicada em diferentes situações e contextos, nas mais diversas áreas, para tomadas de decisões. Na linha desse trabalho, destaca-se a sua aplicabilidade na coordenação de sistemas multi-robôs (FILHO, 2012), como por exemplo robôs que fazem o transporte de produtos em um armazém. Também vale destacar a possibilidade de uso da mesma em *Sistemas de Manufatura Flexíveis*, onde uma linha de produção pode fabricar mais de um tipo de produto por vez, tornando interessante o uso de uma ferramenta que possa maximizar ou minimizar alguma variável da produção, como por exemplo o lucro final ou o custo de produção.

O modo convencional de se enxergar um sistema por meio da TJ envolve distinguir três atores fundamentais: os $jogadores$, as $jogadas$ e o $ambiente$, onde:

- O $jogador$ é o agente racional que analisa o sistema e escolhe a melhor jogada a ser feita;
- As $jogadas$ são os movimentos possíveis de serem feitos pelo jogador em determinado instante do tempo;
- O $ambiente$ é o local onde são feitas as jogadas, por exemplo, o tabuleiro de xadrez.

No modelo de um SED, uma característica predominante é a exatidão que costuma envolver os elementos: eventos, estados e transições. Ainda que em um SED a ocorrência de transições seja assíncrona, elas, em geral, ocorrem deterministicamente, ao menos em uma vasta classe de problemas de controle, os quais são de particular interesse nesse trabalho. Outra característica de interesse e relevante de ser mencionada aqui, é que os sistemas

reais de produção podem, em geral, ser manipulados em um domínio finito. De fato, estamos interessados na classe de problemas e de processos que possam ser convertidos em abordagens práticas o que, portanto, passa pela viabilidade de serem manipulados e armazenados computacionalmente. Esse raciocínio remete ao tamanho e escalabilidade do sistema. Em geral, pode ser assumido, e será assumido nesse trabalho, que o tamanho do sistema não extrapola os limites do tratamento computacional.

Nesse contexto, se torna natural pensar que a questão probabilística da TJ clássica se torna desnecessária. Alternativamente, pode-se optar por meios de modelagem que se adequem melhor ao tipo de sistema aqui abordado, como, por exemplo, o método combinatório, que parte do principio de analisar todas as combinações de jogadas possíveis de um jogador, de maneira a se identificar a jogada *ótima* (GRIFFIN, 2010). Essa abordagem é conveniente especialmente porque a TCS, a qual espera-se combinar neste trabalho com a TJ, é também uma abordagem determinística e que vislumbra sempre a otimidade de uma solução de controle.

Na teoria combinatória é comum o uso de *árvore*, i.e., um *grafo* conexo, orientado e acíclico, para a análise das jogadas, este tipo de representação é chamado de *árvore-jogo* e é definida da seguinte maneira.

Uma *árvore-jogo com informação completa e sem aleatoriedade* é uma tupla $G = \langle T, P, S, \upsilon, \mu, \pi \rangle$ (GRIFFIN, 2010), onde:

- T é uma árvore direcionada, $T = \langle V, A \rangle$, sendo V o conjunto de vértices e A o conjunto de arestas;
- P é o conjunto finito de jogadores;
- S é o conjunto finito de jogadas;
- υ é a função $D = V \backslash F \rightarrow P$ que atribui cada vértice não pertencente ao conjunto de vértices terminais F a um jogador;

- μ é a função que mapeia as arestas da árvore com as jogadas, i.e., $\mu : A \rightarrow S$;

- π é a função de recompensa que atribui a cada vértice terminal da árvore um valor de recompensa para cada jogador, $\pi : F \rightarrow \mathbb{R}^N$.

Nessa abordagem, é importante determinar quem são os agentes tomadores de decisão, i.e., os jogadores, e quais as jogadas que são possíveis em cada momento do jogo, além de definir uma ordem para os jogadores, ou seja, determinar quando será a vez de cada jogador fazer o seu movimento, através de υ. É importante lembrar a necessidade do conhecimento de todas as possibilidades de movimento que levam a um nó folha, ou seja, é necessário conhecer todas as ramificações da árvore para tomar a melhor decisão.

Para exemplificar, seja um jogo simples de Pedra-Papel-Tesoura com dois jogadores, onde as condições para alcançar a vitória são, $pedra$ vence $tesoura$, $papel$ vence $pedra$ e $tesoura$ vence $papel$.

Analisando este tipo de jogo, é facilmente identificado quem são os jogadores, $P = (P1, P2)$, e as jogadas possíveis, $S = (pedra, papel, tesoura)$. Como este tipo de jogo possui condições para se alcançar a vitória, é possível utilizá-las como base para atribuir os ganhos/perdas para cada condição de término do jogo através de π. Caso o jogador $P1$ seja o primeiro a jogar, obtém-se então a árvore-jogo T, mostrada na Figura 5.

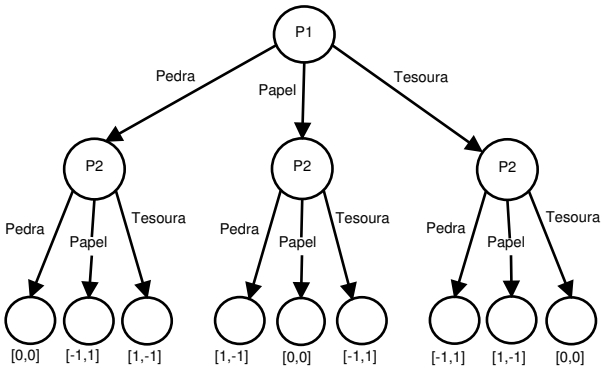


Figura 5: Pedra-Papel-Tesoura com informação completa.
Fonte: Adaptado de Griffin (2010).

Nesse caso, $P2$ possui três possibilidades de movimento para qualquer escolha de $P1$. Além disso, caso a jogada leve a uma vitória de $P1$, o ganho será de 1, caso leve a uma derrota o ganho será de -1 e caso leve a um empate, o ganho será 0. Asim, para garantir a vitória, o jogador $P2$ deve escolher a jogada que maximize o seu ganho.

2.3.1 TOMADA DE DECISÃO

Note que a Teoria dos Jogos analisa diversas situações e desfechos para um sistema, possibilitando que o jogador consiga o melhor resultado possível. Porém, para que isso seja viável computacionalmente, é necessária a utilização de métodos de busca e otimização sobre a árvore-jogo, de forma a encontrar o melhor ação a ser seguida. Existem diversos algoritmos e métodos que fornecem essa funcionalidade, sendo que o mais abordado nessa categoria é o algoritmo *minimax*, cuja ideia é sintetizada no Algoritmo 1.

Algorithm 1 Algoritmo minimax

Necessário: Árvore jogo T
Entrada: nó V, objetivo J
Saída: Melhor ganho A

se V *é uma* $folha$ **então**
    retorne $ganho[V]$
senão
    se J *é* $falso$ **então**
        $A \leftarrow \infty$
        para todo u *filho de* V **faça**
            $A = min(A, minimax(u, verdadeiro))$
        retorne A
    senão
        $A \leftarrow -\infty$
        para todo u *filho de* V **faça**
            $A = max(A, minimax(u, falso))$
        retorne A

O *minimax* é um algoritmo que, na sua forma convencional de implementação, possui dois modos de operação, buscando maximizar ou minimizar o

valor do ganho de acordo com o objetivo do jogador (nó) analisado. No caso do algoritmo apresentado, ele possui um funcionamento alternado em que, caso o nó atual esteja maximizando o valor, seu nó filho irá minimizá-lo. Sendo assim, se J for falso, o algoritmo irá buscar o menor ganho entre as possíveis jogadas (linha 4 à 7). Caso contrário, ele irá maximizar o ganho entre as possíveis jogadas (linha 9 à 12) retornando então o melhor valor possível encontrado (linha 8 e 13 respectivamente). Esse algoritmo utiliza da recursividade para a movimentação pela árvore até encontrar um nó folha, onde ele retorna o ganho obtido pelo respectivo resultado (linha 1 à 2).

3 APLICAÇÃO EM SISTEMA DE MANUFATURA FLEXÍVEL

O primeiro passo para a aplicação da Teoria dos Jogos sobre um processo flexível de manufatura é modelar os componentes do SED, a fim de facilitar a identificação dos componentes correspondentes a TJ, como os jogadores e jogadas.

Um fator importante a ser considerado durante a análise do sistema é a caracterização adequada da função de atribuição de recompensas nos nós folhas da árvore-jogo, pois é através delas que é definido o comportamento do sistema, ou seja, se o sistema buscará maximizar ou minimizar os ganhos ao buscar as melhores jogadas. Deve-se lembrar, ainda, que para o sistema manter a flexibilidade na quantidade de cada item produzidos, deve-se fazer com que os ganhos de cada nó variem após o término de cada jogada utilizando uma função de variação, pois do contrário o sistema irá sempre executar a mesma sequência de eventos.

Outro ponto importante de se notar é que a grande maioria dos jogos estão envolvidos em um ambiente de competição entre os jogadores, em que cada um busca o melhor resultado para si ou o pior resultado para o adversário. Porém, em um sistema de manufatura flexível, os jogadores precisam cooperar para atingir o melhor resultado possível para o jogo, ou seja, nesse contexto pode-se dizer que o objetivo deixa de ser individual e torna-se único para o jogo, sendo compartilhado por todos.

Dessa forma, diferentemente do exemplo exibido no tópico sobre Teoria de Jogos (2.3) onde cada jogador possui, nos nós folhas, uma recompensa específica para o seu movimento, aqui a recompensa final atribuída será igual para todos os envolvidos. É intuitivo pensar, então, que não é necessário um vetor de recompensas, afinal a recompensa não mais é atribuída ao jogador, e sim ao desfecho do jogo em si.

No entanto, note que, ao alterar a função de atribuição de recompensas (π) de forma a atribuir um valor único para as folhas, o sistema de manu-

fatura deixa de se comportar da maneira desejada, pois acaba sendo limitado por características advindas da árvore-jogo. Nesse caso, o sistema só poderia iniciar um jogo por vez, sendo necessário que todos os jogadores realizem seus movimentos antes do inicio de um novo jogo. Na prática, isso significa que o sistema de manufatura se limitaria a manipular apenas uma peça por vez na linha de produção, a cada jogo.

Essa limitação decorre do fato de que, por possuir um único valor como recompensa, este não poderá ser atualizado após uma jogada, pois, caso isso ocorra, a decisão do próximo jogador pode levar a um resultado diferente do encontrado no inicio do jogo. Assim, para evitar essa inconsistência, o peso deverá ser atualizado somente quando o último jogador realizar seu movimento.

3.1 MÉTODO UTILIZADO

Nesse trabalho, uma vez identificada essa limitação na integração da TJ com a TCS, propõe-se uma abordagem que contorna tal problema. Ao invés de realizar a alteração descrita acima para π, permite-se que ele continue atribuindo um vetor composto de recompensas para cada jogador do sistema, desde que as recompensas de um mesmo nó folha sejam iguais. Em outras palavras, existirá uma recompensa única para o nó folha, e essa será atribuída para todo o vetor, i.e., todas as posições do vetor de recompensas possuirão inicialmente o mesmo valor, que corresponderá a recompensa única daquele desfecho para o jogo. Dessa forma, apesar de cada jogador levar em consideração apenas sua respectiva recompensa no vetor, por serem recompensas iguais, ele estará simultaneamente operando para atingir o objetivo compartilhado no jogo. Com esse método será possível atualizar a recompensa correspondente ao jogador após este realizar seu movimento, sem alterar a decisão do próximo a jogar, permitindo então que se inicie um novo jogo, mesmo que o anterior ainda esteja em andamento.

Dito isso, para a realização dos testes e simulações buscou-se para o sistema um comportamento similar ao fornecido pelo sistema sem o auxilio

da TJ. Dessa forma, a função de variação de ganhos utilizada é tal que será reduzida uma unidade da recompensa correspondente ao jogador. Sendo assim, sempre que o jogador decidir sua jogada, o ganho que levou à decisão será decrementado seguindo a fórmula: $Ganho = Ganho - 1$, reiniciando os valores quando todos os ganhos correspondentes ao jogador forem iguais a zero.

Para a busca na árvore-jogo foi utilizado o Algoritmo 2 que é uma versão modificada do algoritmo apresentado em 2.3.1, em que o parâmetro J da função recursiva foi alterado de forma a manter um único objetivo para todos os jogadores durante a análise da árvore-jogo. Ou seja, sempre será buscado o máximo, ou o mínimo, como pode ser visto nas linhas 7 e 15. Além disso, também foi alterado o retorno do algoritmo, sendo que ao término de sua execução, ele retorna também o evento que leva ao melhor resultado encontrado (linhas 9 à 11 e linhas 17 à 19).

Algorithm 2 Algoritmo minimax com retorno de Evento

Necessário: Árvore jogo T
Entrada: Nó inicial v, Objectivo J
Saída: Melhor ganho N, Melhor movimento m

```
 1: se v é uma folha então
 2:     retorne ganho[v], null
 3: senão
 4:     se J é falso então
 5:         N ← ∞
 6:         para todo w filho de v faça
 7:             x, a = minimax(w, falso)
 8:             N = min(N, x)
 9:             se N foi alterado então
10:                 m = A[v, w]
11:         retorne N, m
12:     senão
13:         N ← −∞
14:         para todo w filho de v faça
15:             x, a = minimax(w, verdadeiro)
16:             N = max(N, x)
17:             se N foi alterado então
18:                 m = A[v, w]
19:         retorne N, m
```

É interessante notar que, dependendo da implementação das funções min (linha 8) e max (linha 16), o valor de N, para casos onde ambos os pesos são iguais, pode ser alterado ou não. Neste trabalho assumiu-se uma implementação onde N não será alterado neste tipo de situação.

A seguir serão apresentados duas situações diferentes onde foram construídas as respectivas estruturas para o funcionamento da TJ juntamente com a TCS, assim como seus resultados e um comparativo com uma abordagem sem a utilização da TJ. Também serão apresentadas simulações a fim de demonstrar a limitação operacional citada no paragrafo anterior, bem como a respectiva solução proposta neste trabalho.

3.2 SISTEMA 1

Considere um sistema de manufatura em que, em uma mesma linha de produção, possam ser fabricados dois tipos de produtos distintos, conforme a Figura 6.

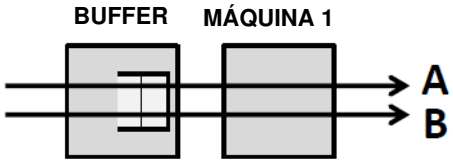
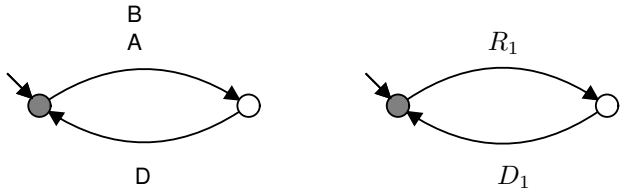


Figura 6: Sistema de Manufatura Flexível.
Fonte: Adaptado de Santos (2013).

Nesse contexto, uma Máquina S, localizada no inicio da linha de produção, deve decidir qual, entre dois tipos distintos de produtos (sendo eles A e B), será adicionado a um *buffer* de duas posições, de forma que possa ser processado posteriormente pela máquina M_1.

Modelando o sistema usando uma abordagem de SEDs, obtém-se os autômatos representados na Figura 7, em que S pode executar os eventos A e B em seu estado inicial e M_1 é uma máquina simples com apenas um modo de operação, representado por R_1. Ambos possuindo também eventos

de finalização, sendo eles D e D_1 respectivamente.



(a) S (b) M_1

Figura 7: Plantas do Sistema 1.
Fonte: Autoria própria.

Também obtém-se o modelo da especificação que controla o funcionamento do *buffer*, conforme apresentada na 8.

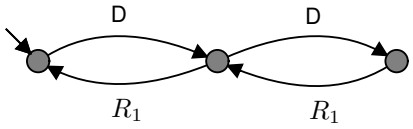


Figura 8: Buffer de duas posições do Sistema 1.
Fonte: Autoria própria.

Nesse exemplo, plantas e especificações são equivalentemente representadas, com ou sem a Teoria dos Jogos integrada à TCS. As diferenças entre as duas abordagens serão discutidas a seguir.

3.2.1 ABORDAGEM SEM TEORIA DOS JOGOS

Para que seja possível o funcionamento desse sistema é necessário definir qual evento ocorrerá quando *S* executar. Um dos modos de se implementar essa funcionalidade é acrescentando uma nova especificação que irá definir a prioridade e uma sequência de execução para os eventos. Nesta simulação, E_1 (Figura 9) é uma especificação com essa característica, onde os eventos *A* e *B* serão executados alternadamente e *A* irá ser executado primeiro, ou seja, possui maior prioridade.

Obtêm-se então, através da aplicação da TCS, o supervisor demonstrado na Figura 20, com as seguintes características:

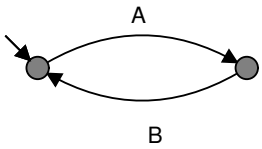


Figura 9: Especificação E_1 do Sistema 1.
Fonte: Autoria própria.

Tabela 1: Supervisor com definição de sequência do Sistema 1.

Supervisor com Sequência alternada		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
24	5	40

O supervisor construído possui 24 estados possíveis, com 5 eventos e 40 transições entre estados. É importante notar que, o número de estados e transições aumentará de acordo com a complexidade da sequência utilizada para *A* e *B* em E_1.

3.2.2 ABORDAGEM UTILIZANDO TEORIA DOS JOGOS

Ao utilizar-se da Árvore-Jogo obtida através da TJ elimina-se a necessidade de adicionar uma nova especificação ao sistema. Dessa forma é obtido o Supervisor representado na Figura 21 e que possui as seguintes características:

Tabela 2: Supervisor sem definição de sequência do Sistema 1.

Supervisor sem definição de sequência		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
12	5	26

Neste caso o supervisor possui apenas 12 estados, 5 eventos e 26 transições possíveis. Entretanto, ainda é necessária a criação da árvore-jogo, que irá controlar a sequência de eventos do supervisor. Dessa forma, utilizando a metodologia apresentada na Seção 2.3 para a TJ, é possível obter a seguinte árvore-jogo:

$$G = \langle T, \{S, M_1\}, \{A, B, R_1\}, \upsilon, \mu, \pi \rangle$$

Essa árvore é representada graficamente na Figura 10, onde *S* é o jogador inicial e P_1 e P_2 os pesos atribuídos as folhas da árvore-jogo a serem considerados durante a tomada de decisão.

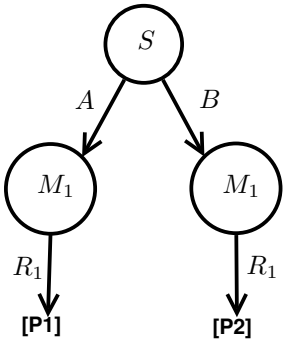


Figura 10: Arvore-Jogo do Sistema 1.
Fonte: Autoria própria.

É possível perceber neste problema que, por M_1 possuir apenas uma jogada possível, ele não influencia no resultado do jogo, tendo apenas uma função transitória. Dessa forma é possível reduzir a árvore-jogo, ocultando as jogadas de M_1, o que resulta em uma árvore de menor profundidade. Vale lembrar que a ação de M_1 influencia no produto final que será produzido, porém não influencia na busca de melhor jogada através da árvore. Acredita-se que isso seja válido de maneira genérica, para todos os jogadores que possuem menos que duas jogadas possíveis, sendo assim desnecessário representá-los no sistema-jogo.

3.2.3 SIMULAÇÃO

Primeiramente é interessante demonstrar o funcionamento do sistema sem o auxilio da Teoria dos Jogos. Para isso, foram utilizados os mesmos modelos já demonstrados nesta seção. Para iniciar a produção, o único evento possível de ocorrer deve ser a ação de S, ou seja, colocar uma peça no *buffer*. E, adicionalmente, pela especificação E_1, a única opção de inserção neste momento é A, assim como visto na Figura 11.

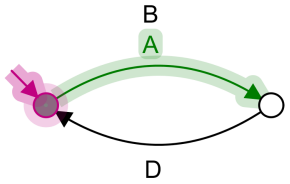
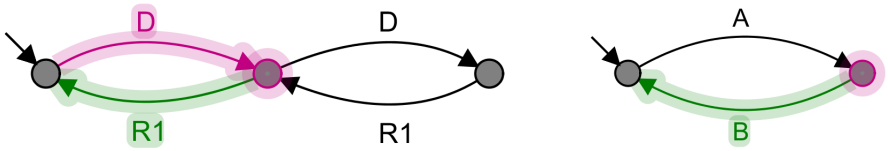


Figura 11: Início do Sistema 1 sem tomada de decisão.
Fonte: Autoria própria.

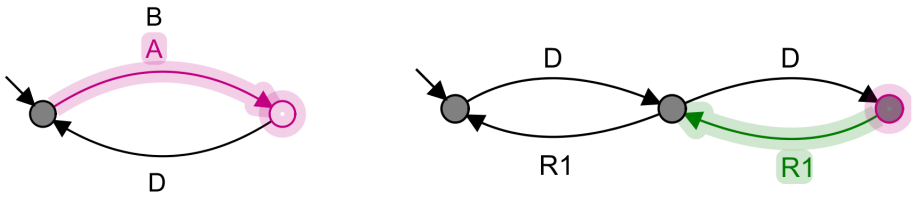
Ao realizar A, a máquina S deve então finalizar sua ação através de D, levando o sistema a situação onde existirá uma peça no *buffer*, liberando então M_1 para a manufatura. Ao mesmo tempo, E_1 irá impedir que S execute A novamente, mas irá permitir que se execute B, Figura 12.



(a) buffer (b) E1

Figura 12: Especificações do Sistema 1 após execução de A.
Fonte: Autoria própria.

Após S executar B, a sequência de produção reinicia, ou seja, A estará novamente habilitado para execução, porém só poderá ser adicionado ao *buffer* se M_1 processar ao menos uma das peças anteriores através de R_1.



(a) S (b) buffer

Figura 13: S esperando M_1 retirar uma peça do *buffer* para realizar A.
Fonte: Autoria própria.

Nessa situação o sistema funcionou exatamente como o esperado, seguindo a sequência de eventos pré estabelecida em E_1, porém sem flexibili-

dade, ou seja, o sistema sempre executará essa mesma sequência.

Agora será apresentada a simulação do sistema em conjunto com a Teoria dos Jogos. Para isso, como já explicado anteriormente, será utilizado um decréscimo unitário com o objetivo de simular a ação sequencial da especificação E_1. Além disso, para esse sistema foi utilizada a árvore-jogo da Figura 10 já reduzida.

Sendo assim, desejamos que o sistema opere de forma que os eventos A e B sejam executados de forma similar ao resultado anterior. Os pesos para que isso ocorra podem ser deduzidos do fato que a relação de execução destes eventos pode ser dita como de um para um, ou seja, ambos possuem a mesma prioridade e, consequentemente, para utilização da TJ os pesos atribuídos aos nós folhas do jogo também devem ser iguais.

Assumindo então pesos iguais a 1 para ambos os eventos, a função de recompensas π irá adicionar aos nós folhas um vetor de recompensas de tamanho igual a quantidade de jogadores, por possuir apenas um jogador (S), o vetor será de apenas uma posição. Sendo assim obtida a árvore-jogo representada na Figura 14.

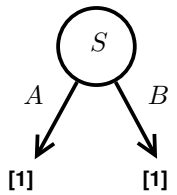


Figura 14: Árvore-jogo reduzida e com recompensas.
Fonte: Autoria própria.

Com a árvore-jogo construída e utilizando o Algoritmo 2 para a tomada de decisão, o sistema já está apto para execução. Sendo assim, ao iniciar o sistema, S irá realizar a chamada do algoritmo de decisão e, assumindo que seja utilizada a parcela do algoritmo para busca do maior valor, a função irá retornar o evento correspondente que, no estado inicial, por todas as recompensas serem iguais, irá retornar o primeiro a ser encontrado, ou seja, retornará o evento A. Então será realizada a atualização da recompensa de S

para o evento A que passará a ser igual a zero. Sendo assim, em uma próxima execução, o evento retornado será B que possui uma recompensa maior. Neste ponto, como todos os pesos são iguais a zero, o sistema irá reinicia-los, i.e., as recompensas irão voltar aos seus valores iniciais.

3.3 SISTEMA 2

Agora, considere um sistema similar ao apresentado pela Figura 6, porém com a máquina M_1 contendo dois eventos passíveis de execução, R_1 e R_2. As plantas desse sistema são apresentadas na Figura 15.

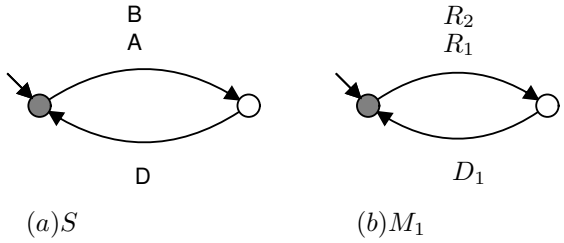


Figura 15: Plantas do Sistema 2.
Fonte: Autoria própria.

Analogamente ao caso anterior, também é obtida uma especificação para limitar o funcionamento do buffer de duas posições, como mostra a Figura 16.

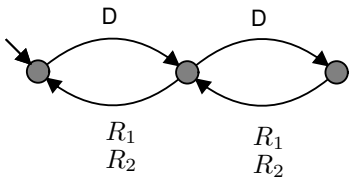


Figura 16: Buffer de duas posições do Sistema 2.
Fonte: Autoria própria.

Neste caso, para a utilização do sistema sem o auxílio da árvore-jogo, é necessário o acréscimo de uma ou mais especificações para controle dos eventos de S e de M_1. A construção dessas especificações pode se tornar

extremamente complexa de acordo com o comportamento desejado.

Para o controle e construção das especificações é preciso notar que existem quatro tipos de produtos que podem ser gerados, sendo eles: A_{R_1}, A_{R_2}, B_{R_1} e B_{R_2}. Assumindo que todos os resultados possuem a mesma prioridade, para simplificação, e com uma especificação E_1 igual ao da Figura 9, foi obtido a seguinte especificação para os eventos de M_1.

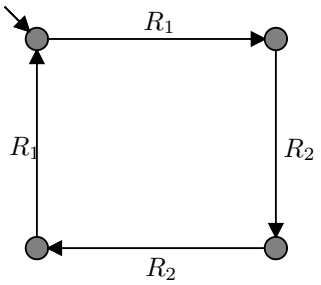


Figura 17: Especificação para M_1 do Sistema 2.
Fonte: Autoria própria.

Analisando de forma similar ao sistema anterior, é obtido um supervisor com as seguintes características:

Tabela 3: Supervisor com definição de sequência do Sistema 2.

Supervisor com Sequência alternada		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
48	6	80

Por sua vez, analisando o sistema integrado à TJ, remove-se a necessidade das novas especificações para coordenar a sequência de eventos e obtêm-se um supervisor estatisticamente definido como na Tabela 4.

Tabela 4: Supervisor sem definição de sequência do Sistema 2.

Supervisor sem Sequência alternada		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
12	6	30

Note que, de forma similar ao caso anterior, há uma redução na

quantidade de estados e de transições, como esperado.

Por sua vez, a árvore-jogo obtida para esse sistema é descrita por G, em que:

$$G = \langle T, \{S, M_1\}, \{A, B, R_1, R_2\}, \upsilon, \mu, \pi \rangle.$$

Esse modelo pode ser visualizado na Figura 18.

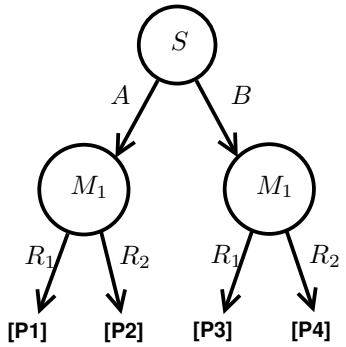


Figura 18: Arvore-Jogo do Sistema 2.
Fonte: Autoria própria.

Neste caso, não é possível uma simplificação que elimine as jogadas de M_1, pois ele inclui duas jogadas diferentes e isso pode influenciar no resultado do jogo.

3.3.1 SIMULAÇÃO

Como já visto para o sistema anterior, o funcionamento do sistema com a adição de novas especificações segue exatamente aquilo para o qual foi idealizado. Neste caso o sistema irá produzir resultados seguindo a sequencia, A_{R_1}, B_{R_2}, A_{R_2} e por fim B_{R_1}, voltando ao estado inicial onde o ciclo se repetirá.

Por sua vez, utilizando a árvore-jogo para a tomada de decisão, Figura 19, também com recompensas iguais a 1 para todos os resultados é obtido um comportamento similar.

Com o sistema em seu estado inicial, S irá executar o algoritmo de decisão que irá buscar o melhor resultado baseado nas recompensas atribuídas para si nos vetores (primeira posição nos vetores). Neste sistema, a máquina

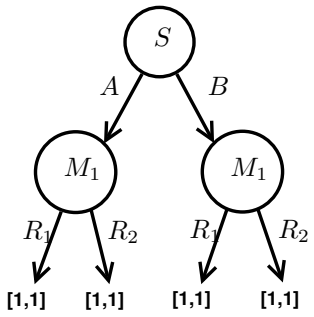


Figura 19: Árvore-Jogo com recompensas para o Sistema 2.
Fonte: Autoria própria.

M_1 também irá executar a tomada de decisão quando existir alguma peça no buffer, buscando o melhor resultado baseado em suas recompensas (segunda posição dos vetores). Dessa forma, a execução dos eventos seria conforme a tabela abaixo.

Tabela 5: Sequência de execução para S e M_1.

Primeira execução			
A_{R_1} $[0,1]$	A_{R_2} $[1,1]$	B_{R_1} $[1,1]$	B_{R_2} $[1,1]$
Segunda execução			
A_{R_1} $[0,1]$	A_{R_2} $[0,1]$	B_{R_1} $[1,1]$	B_{R_2} $[1,1]$
Terceira execução			
A_{R_1} $[0,0]$	A_{R_2} $[0,1]$	B_{R_1} $[0,1]$	B_{R_2} $[1,1]$
Quarta execução (reinício de S)			
A_{R_1} $[0,0]$	A_{R_2} $[0,0]$	B_{R_1} $[0,1]$	B_{R_2} $[0,1]$
Quinta execução			
A_{R_1} $[1,0]$	A_{R_2} $[1,0]$	B_{R_1} $[1,0]$	B_{R_2} $[1,1]$
Sexta execução (reinício de M_1)			
A_{R_1} $[0,0]$	A_{R_2} $[1,0]$	B_{R_1} $[1,0]$	B_{R_2} $[1,0]$
Sétima execução			
A_{R_1} $[0,1]$	A_{R_2} $[0,1]$	B_{R_1} $[1,1]$	B_{R_2} $[1,1]$

Em um jogo comum, S deveria esperar até que M_1 realizasse seu movimento para então poder realizar uma nova jogada. Porém, com a utilização dos vetores nos nós folhas é possível que S realize uma nova jogada antes de M_1, o que caracteriza a criação de um novo jogo, permitindo que S consiga fornecer peças continuamente e que o sistema não fique ocioso.

Da mesma forma que o sistema anterior, as recompensas de um jogador podem ser reiniciadas quando todos os valores correspondentes aquele jogador forem iguais a zero, garantindo, conforme exemplificado acima, uma execução continua do sistema.

Vale ressaltar que a execução acima pode ser diferente dependendo da ordem de execução do algoritmo de tomada de decisão por parte dos jogadores, pois, no sistema utilizado, existem estados onde tanto S quanto M_1 podem realizar jogadas, como por exemplo, no estado onde existe uma única peça no buffer.

3.4 COMPARATIVO ENTRE AMBAS ABORDAGENS

A Teoria dos Jogos elimina a necessidade de definir uma sequência de eventos em determinado estado ao acrescentar a estrutura da árvore-jogo, que é independente do supervisor, tornando desnecessária a criação de uma nova especificação para tal, o que por sua vez reduz a complexidade do sistema Supervisor. Essa característica é facilmente observável ao analisar o Supervisor resultante de ambas as abordagens.

Para o primeiro sistema, tem-se o seguinte comparativo:

Tabela 6: Comparativo Sistema 1.

Supervisor com nova especificação		
$\|Q\|$	$\|\Sigma\|$	$\|\rightarrow\|$
24	5	40
Supervisor sem definição de nova especificação		
$\|Q\|$	$\|\Sigma\|$	$\|\rightarrow\|$
12	5	26

Analogamente, para o segundo sistema tem-se:

Tabela 7: Comparativo Sistema 2.

Supervisor com novas especificações		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
48	6	80
Supervisor sem definição de novas especificações		
$\|Q\|$	$\|\Sigma\|$	$\| \rightarrow \|$
12	6	30

É fácil perceber que a complexidade do supervisor foi reduzida ao se eliminar a especificação que controla os eventos de S e de M_1. Essa complexidade foi transferida para árvore-jogo que irá realizar o processo de decisão entre os eventos necessários baseado nas métricas utilizadas na definição dos pesos das folhas da árvore.

Essa característica é interessante enquanto o sistema possuir poucos jogadores e/ou jogadas por jogador, afinal, por inserir uma busca durante o processo, haverá um atraso no funcionamento do sistema sempre que for necessário a tomada de decisão, se tornando crítico para sistemas muito grandes, pois como é necessária a análise de toda a árvore-jogo, o tempo de resposta será lento. Apesar disso com a adição da árvore-jogo ao sistema, é obtido a flexibilidade na quantidade de cada item produzido, sendo necessário apenas alterar as métricas da árvore-jogo, algo que sem o auxílio da TJ necessitaria de uma modificação nas respectivas especificações, o que causaria a modificação do supervisor, aumentando sua complexidade de acordo com a complexidade da sequência. Outra característica obtida é a modularidade do sistema, onde os componentes relacionados a decisão podem ser implementados de forma separada do sistema de controle.

4 CONCLUSÃO

Este trabalho apresentou um método capaz de atribuir flexibilidade ao controle de um Sistema a Eventos Discretos, usando Teoria dos Jogos. Nessa abordagem, o sistema é visto como um jogo, em que cada componente do sistema é visto como um jogador. Então, os jogadores e jogadas são integrados por algoritmos computacionais que calculam a tomada de decisão por meio da propagação de pesos sobre a árvore resultante da TJ.

A principal vantagem da abordagem proposta é que ela permite que se altere o comportamento do sistema, via controle automático, por meio de estímulos externos gerados por uma estrutura extra acoplada ao controlador. A manipulação computacional dessa estrutura é então executada a parte, sem a necessidade de uma remodelagem de todo o sistema de controle a cada cenário possível, o que em um processo flexível, por exemplo, pode ser numeroso.

Uma característica que se deve notar é que, utilizando a abordagem de jogos por turnos, como a utilizada neste trabalho, o funcionamento do sistema fica limitado a apenas um jogo simultâneo. Em outras palavras, para um sistema de manufatura por exemplo, apenas uma peça poderia ser fabricada em cada jogo, e apenas após a sua finalização o sistema ficaria livre para a produção de um novo produto, algo inconcebível em um situação real. Sendo assim, foi necessário propor e construir uma abordagem modificada da TJ para contornar essa limitação, o que remete à principal contribuição desse trabalho.

Como já mencionado, a utilização da TJ produz uma resposta ao sistema de acordo com os ganhos da árvore-jogo, possibilitando retirar parte da complexidade da estrutura do controlador lógico, movendo-a para os algoritmos de decisão. Sob o ponto de vista de implementação, isso é naturalmente visto como um aspecto positivo, principalmente pela possibilidade de modularização da implementação. Porém, essa abordagem pode se tornar bastante onerosa ao se tratar de sistemas com muitos jogadores, ou com mui-

tas jogadas possíveis para um mesmo jogador, de forma que a implementação pode necessitar de um *hardware* exclusivo para a execução da tomada de decisão.

Outro aspecto a se ressaltar é que este trabalho explorou uma classe de sistemas para os quais não existe a possibilidade de retrabalho de produtos. Como se evidencia na literatura, retrabalho possui peculiaridades complexas sob o ponto de vista de controle, as quais, estima-se, podem se beneficiar da abordagem aqui proposta. Dessa forma, uma das opções de trabalhos futuros seria analisar e desenvolver uma solução para sistemas com características de ciclo e reciclo, que seria o equivalente ao jogo possuir mais de uma rodada de jogadas.

REFERÊNCIAS

CASSANDRAS, Christos G.; LAFORTUNE, Stephane. **Introduction to discrete event systems**. 2nd ed. ed. [S.l.]: Springer, 2007.

CURY, Jose E. R. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos**. [S.l.]: V Simpósio Brasileiro de Automação Inteligente, 2001.

FILHO, José Gilmar Nunes de Carvalho. **Modelagem e Síntese para coordenação de sistemas multi-robôs baseada numa estrutura de jogo**. Universidade Federal de Santa Catarina - UFSC, Florianópolis: Tese de Mestrado - Mestrado em Engenharia de Automação e Sistemas - Programa de Pós-Graduação em Engenharia de Automação e Sistemas, 2012.

GRIFFIN, Christopher. **Game Theory: Penn State Math 486 Lecture Notes**. [S.l.: s.n.], 2010.

MYERSON, Roger B. **Game Theory: Analysis of Conflict**. [S.l.]: Harvard University Press, 1997.

OGATA, Katsuhiko. **Modern Control Engineering**. 3. ed. [S.l.]: Prentice-Hall International, Inc., 1997.

RAMADGE, P. J.; WONHAM, W. M. **The Control of Discrete Event Systems**. [S.l.]: Proceedings of IEEE, IEEE, 1989.

SANTOS, Octávio Francisco Paschoal Silva Ribeiro dos. **Modelagem de Sistemas Flexíveis de Manufatura para Tomada de Decisão em Tempo Real**. Universidade Federal de Itajubá - UNIFEI, Itajubá: Tese de Mestrado - Mestrado em Ciências em Engenharia Elétrica - Programa de Pós-Graduação em Engenharia Elétrica, 2013.

TEIXEIRA, Marcelo. **Explorando o uso de distinguidores e de autômatos finitos estendidos na teoria do controle supervisório de sistemas a eventos discretos**. Universidade Federal de Santa Catarina - UFSC, Florianópolis: Tese de Doutorado - Doutorado em Engenharia de Automação e Sistemas - Programa de Pós-Graduação em Engenharia de Automação e Sistemas, 2013.

VILELA, Juliana Nogueira; PENA, Patrícia Nascimento. **Abstração do Supervisor para Aplicação na Solução de Problemas de Planejamento em Sistemas de Manufatura**. Universidade Federal de Minas Gerais - UFMG, Belo Horizonte: Programa de Pós-Graduação em Engenharia Elétrica, 2016.

APÊNDICE A - SUPERVISOR COM SEQUÊNCIA DE EVENTOS

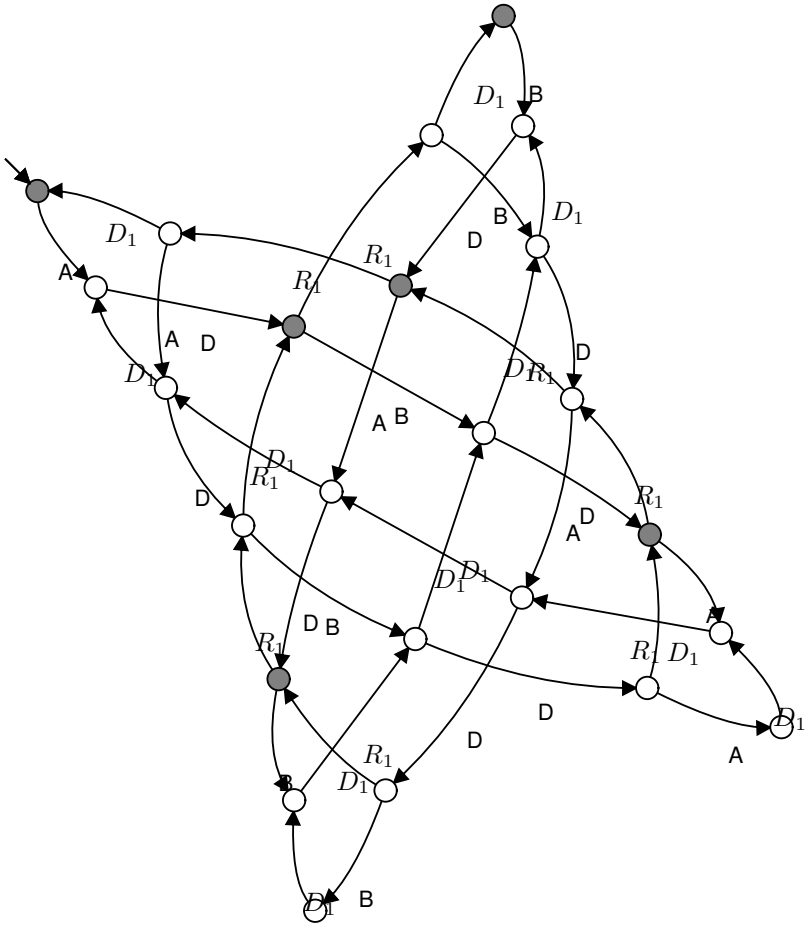


Figura 20: Supervisor sem TJ do Sistema 1.
Fonte: Autoria própria.

APÊNDICE B - SUPERVISOR SEM SEQUÊNCIA DE EVENTOS

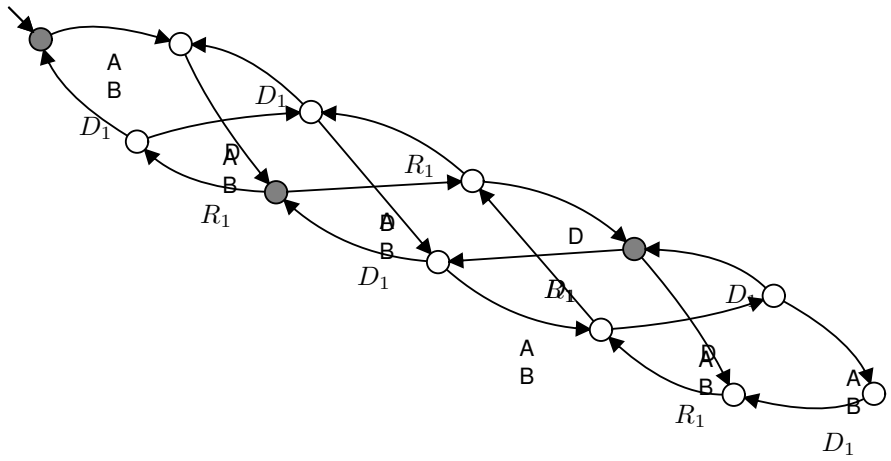


Figura 21: Supervisor com auxilio da TJ do Sistema 1.
Fonte: Autoria própria.