MIGUEL ANGEL DE MARCHI AMARILLA

UMA ABORDAGEM MATRICIAL PARA DESDOBRAMENTO DE REDES DE PETRI UTILIZANDO A FERRAMENTA MATLAB

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. Marcos Castilho.

Co-orientador: Prof. Dr. Luis Allan Künzle.

CURITIBA PR
2016

Amarilla, Miguel Angel de Marchi
Uma abordagem matricial para desdobramento de redes de petri utilizando a ferramenta Matlab / Miguel Angel de Marchi Amarilla. – Curitiba, 2016.
57 f. : il.

Dissertação (mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Marcos Castilho
Coorientador: Luis Allan Künzle

1. Redes de petri. 2. Algoritmos. 3. MATLAB (Programa de computador). I. Castilho, Marcos. II. Künzle, Luis Allan. III. Título

CDD 511.5

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós Graduação em INFORMÁTICA
Código CAPES: 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **MIGUEL ANGEL DE MARCHI AMARILLA**, intitulada: "**Uma abordagem matricial para desdobramento de Redes de Petri utilizando a ferramenta Matlab**", após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua aprovação.

Curitiba, 15 de Julho de 2016.

Prof MARCOS ALEXANDRE CASTILHO
Presidente da Banca Examinadora (UFPR)

Prof FRANCK CARLOS VELEZ BENITO
Avaliador Externo (UTFPR)

Prof LUIS ALLAN KÜNZLE
Coorientador - Avaliador Interno (UFPR)

Prof EDUARDO DE FREITAS ROCHA LOURES
Avaliador Externo (PUC/PR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Resumo

Nas últimas décadas, redes de Petri têm sido amplamente utilizadas como ferramenta para modelar, analisar, simular e avaliar o comportamento e desempenho de sistemas com peculiaridades de sincronização, concorrência e compartilhamento de recursos, como sistemas de manufatura, robótica, sistemas de tempo real entre outros. Um dos problemas enfrentados durante a modelagem desses sistemas está na explosão combinatória do número de estados, o que inviabiliza qualquer método de análise que tenha como base a enumeração desses estados, como o grafo de alcançabilidade. McMillan desenvolveu uma técnica conhecida como desdobramento - *unfolding*, a qual gera uma nova rede, de complexidade menor que a do grafo de alcançabilidade, que permite evitar análises enumerativas em sistemas modelados por redes de Petri. Em 2002, o algoritmo de McMillan foi aperfeiçoado por Esparza, Römer e Vogler, sendo criado o algoritmo ERV *Unfolding*, implementado nas ferramentas *CUnf, MOLE* e *PUnf*. Essas ferramentas são soluções específicas *stand-alone*, tendo como desvantagem a complexidade dos algoritmos e a consequente dificuldade de manutenção, prejudicando a alteração dos códigos para estudos e simulação de redes de Petri em ambientes distintos.

A abordagem matricial de uma rede de Petri permite com maior facilidade, o estudo das propriedades estruturais e comportamentais da rede, sendo inclusive um facilitador para a implementação de um algoritmo de desdobramento, uma vez que a técnica será implementada através de operações matriciais. A abordagem matricial permite também a simulação e validação de projetos, visando minimizar falhas ou interrupções (*deadlocks*) do sistema. Assim, a presente proposta de dissertação tem como objetivo construir uma abordagem matricial para o desdobramento em redes de Petri, implementando-a como um módulo da ferramenta *MATLAB*. As implementações serão validadas através de estudo de caso, sendo posteriormente avaliados os resultados obtidos e as limitações da ferramenta em relação aos algoritmos citados.

Palavras-chave: Redes de Petri, desdobramento, algoritmos, *MATLAB*.

Abstract

In recent decades, Petri nets have been widely used as a tool to model, analyze, simulate and evaluate the behavior and performance of systems with synchronization peculiarities, competition and resource sharing, as manufacturing systems, robotics, real-time systems among others. One of the problems faced during the modeling of these systems is the combinatorial explosion of the number of states, which prevents any method of analysis which is based on the enumeration of these states, as the states of reachability graph. McMillan created a technique known as split - unfolding, which generates a new network, complexity lower than the reachability graph, thus avoiding enumerative analysis systems modeled by Petri nets. In 2002, the algotirmo McMillan was perfected by Esparza, Römer and Vogler, being created the ERV Unfolding algorithm, implemented in *CUnf, MOLE* and *PUnf* tools. These tools are specific stand-alone solutions, with the disadvantage of the complexity of the algorithms and the consequent difficulty of maintenance, damaging the change of codes for studies and simulation of Petri nets in different environments.

The matrix approach of a Petri net with greater ease allows the study of the structural and behavioral properties of the network, including being an enabler for implementing an unfolding algorithm, since the technique is implemented using matrix operations. The matrix approach also allows the simulation and design validation to minimize failures or interruptions (deadlocks) system. The proposed thesis aims to build a matrix approach for deployment on Petri nets, implementing it as a module of *MATLAB* tool. The implementations will be validated through case studies, and later evaluated the results and the limitations of the tool relative to those cited algorithms.

Keywords: *Petri net, unfolding, algorithm, MATLAB*.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente, as organizações necessitam automatizar seus processos, com o objetivo de garantir a eficiência e a diminuição do desperdício dos recursos envolvidos. No decorrer dos anos, o avanço tecnológico tornou processos e sistemas cada vez mais complexos, sendo necessário a utilização de ferramentas de modelagem visando evitar erros durante a execução do sistema e incoerências entre os processos. Das várias ferramentas disponibilizadas para este propósito, pode-se citar as redes de Petri.

As redes de Petri (RdP) são um modelo matemático, proposto pela primeira vez por Carl Adam Petri [Petri, 1962] em sua tese de doutorado, na qual apresentou um tipo de grafo bipartido com o objetivo de estudar a comunicação entre sistemas autômatos. São utilizadas para modelar sistemas dinâmicos a eventos discretos caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não-paralelos, determinísticos ou estocásticos [Murata, 1989]. Uma rede de Petri permite analisar um sistema modelado, revelando propriedades importantes do sistema em relação a sua estrutura e comportamento dinâmico, facilitando sua modificação quando necessário. Na área de automação, por exemplo, elas se constituem em uma importante ferramenta pois permitem modelar sistemas representados por eventos discretos, visando analisar o seu comportamento [Cassandras, 1999].

Redes de Petri podem ser utilizadas tanto na forma gráfica quanto matricial, permitindo a modelagem, a análise e a validação de propriedades dos sistemas por ela modelados. Além disso, são adequadas e eficientes para modelar atividades que podem ser executadas independentemente uma da outra, de forma paralela ou concorrente.

A principal técnica que permite a análise de um modelo em redes de Petri se dá por meio do grafo de alcançabilidade, que apresenta todos os estados que o sistema modelado pode assumir. A desvantagem desta técnica está no custo computacional e na explosão do espaço de estados [Benito, 2010], pois a construção do grafo de alcançabilidade torna obrigatório enumerar todas as marcações alcançáveis, o que pode se tornar um problema intratável em sistemas reais, nos quais o número de eventos concorrentes é elevado. Porém, na última década, a comunidade científica tem focado seu interesse em uma nova técnica para estudo das redes de Petri, conhecida como desdobramento *(unfolding)*. McMillan [McMillan, 1995] propôs inicialmente um algoritmo de

desdobramento visando auxiliar na resolução do problema de explosão de estados de sistemas modelados com redes de Petri finitas. Em 2002, o algoritmo de McMillan foi aperfeiçoado por Esparza, Römer e Vogler [Esparza et al., 2002], sendo criado o algoritmo ERV *Unfolding*, servindo de base para implementação de diversas ferramentas, entre elas, *CUnf*, *MOLE* e *PUnf*.

A técnica de desdobramento se propõe a construir uma rede de ocorrências finita, na qual estão representadas todas as marcações alcançáveis. É sobre essa nova rede, finita e acíclica, que são analisadas as propriedades da rede, sem a necessidade de enumeração dos estados alcançáveis.

1.1 Problema e Motivação

As principais ferramentas disponíveis para geração de uma rede desdobrada a partir de uma rede de Petri segura utilizam abordagens diferentes de implementação, sendo todos programas *stand-alone*, em que para seu funcionamento, não necessitam de *software* auxiliar, como um interpretador, para serem executados. A ferramenta *MOLE* utiliza-se do algoritmo ERV *Unfolding*. O *PUnf* também utiliza como base o algoritmo ERV *Unfolding*, porém, adiciona uma técnica de paralelismo para geração do desdobramento. Já a ferramenta *CUnf* utiliza-se de uma abordagem conhecida como redes contextuais *(c-net)* para gerar o desdobramento da rede.

Do ponto de vista da implementação, todas as ferramentas foram desenvolvidas com a linguagem de programação C, utilizando-se de estruturas de listas e ponteiros. Apesar das três ferramentas gerarem desdobramento semelhante para uma mesma rede de Petri, conforme será visto no Capítulo 4, todas apresentam como desvantagem a complexidade dos algoritmos e a consequente dificuldade de manutenção, prejudicando a alteração dos códigos para estudos e simulação de redes de Petri em ambientes distintos. Outra desvantagem é que apesar de utilizarem o algoritmo ERV *Unfolding*, uma vez aplicadas em um cenário de grande porte, as ferramentas *CUnf* e *MOLE* apresentaram resultados divergentes em relação a ferramenta *PUnf*.

Assim, dada as limitações apresentadas acima, além da inexistência de uma abordagem matricial para a geração de uma rede de desdobramento a partir de uma rede de Petri Lugar/Transição, este trabalho tem como motivação estudar o processo de desdobramento e as ferramentas existentes, e desenvolver uma solução baseada em cálculo matricial. A abordagem matricial de uma rede de Petri permite com maior facilidade, o estudo das propriedades estruturais e comportamentais da rede, sendo inclusive um facilitador para a implementação de um algoritmo de desdobramento, uma vez que a técnica será implementada através de operações matriciais. A abordagem matricial permite também a simulação e validação de projetos, visando minimizar falhas ou interrupções (*deadlocks*) do sistema [Santos et al., 2014].

1.2 Trabalhos Disponíveis

A resolução do problema da explosão do espaço de estados através da técnica de desdobramento tem recebido grande atenção por parte da comunidade acadêmica e científica, tendo sido proposto diferentes metodologias de estudo e abordagens: [Esparza and Schröter, 2001], [Esparza et al., 2002], [Benaccio, 2008], [Benito, 2010], [Rodríguez and Schwoon, 2013], [Bonet et al., 2014], [Badouel et al., 2015] e [Benito, 2015]. No tocante a uma abordagem matricial visando a construção do desdobramento de uma rede de Petri Lugar/Transição, nenhum estudo foi encontrado abordando o tema. Para validar essa afirmação, foi utilizada a ferramenta de busca *Scopus*, que consiste em um banco de dados de resumos e citações de artigos para jornais e revistas acadêmicos, abrangendo aproximadamente 19.000 títulos de mais de 5.000 editoras internacionais, nos campos do conhecimento científico e tecnológico. Esta ferramenta é de propriedade da Editora Elsevier e está disponibilizada na Web para assinantes [Elsevier, 2016]. A ferramenta encontrou apenas um artigo publicado em revista, que faz uso da ferramenta *MATLAB* apenas para simulação de uma rede de Petri segura para análise do grafo de alcançabilidade em sistemas de manufatura [Cabasino et al., 2010].

O presente trabalho tem por objetivo propor uma abordagem matricial na construção da rede de desdobramento, tendo por base o algoritmo ERV *Unfolding*. O ambiente de implementação proposto é a ferramenta *MATLAB*. As demais implementações encontradas na bibliografia, como *CUnf*, *MOLE* e *PUnf*, são soluções específicas *stand-alone*, dedicadas exclusivamente à geração do desdobramento. Por outro lado, a ferramenta *MATLAB*, que possui alternativas de *software* livre, permitirá integrar a solução proposta a outras implementações, incorporando o desdobramento à análise da rede.

A presente dissertação está organizada da seguinte forma: no capítulo 2 é realizada uma revisão bibliográfica dos conceitos mais relevantes sobre redes de Petri, a técnica do grafo de alcançabilidade e seus problemas, os quais são necessários para a compreensão deste trabalho. No capítulo 3 são conceituados e descritos o processo de desdobramento, incluindo alguns conceitos fundamentais para o entendimento deste bem como o algoritmo ERV *Unfolding*. No capítulo 4 são apresentadas as ferramentas de desdobramento utilizadas normalmente para estudo das redes: *CUnf*, *MOLE* e *PUnf*. Já no capítulo 5, é descrita a proposta de trabalho, que consiste no desenvolvimento de uma abordagem matricial da rede de desdobramento, tendo por base os algoritmo ERV *Unfolding*, utilizando a ferramenta *MATLAB*. Por fim, no capítulo 6 são apresentadas as conclusões referentes a proposta de desdobramento, sob a ótica de uma abordagem matricial, sendo sugeridos trabalhos futuros que podem ser realizados a partir desta pesquisa.

Capítulo 2

Redes de Petri

No presente capítulo são apresentados os principais conceitos referente a redes de Petri (RdP) e sua dinâmica, sua representação gráfica e formalismo matemático, propriedades e classes. Com relação à análise de propriedades, são apresentadas a técnica do grafo de alcançabilidade e o problema da explosão de estados, os quais são necessários para a compreensão deste trabalho.

2.1 Conceitos Fundamentais

Redes de Petri são uma ferramenta de modelagem gráfica e matemática, sendo introduzidas em 1962 por Carl Adam Petri [Petri, 1962] em sua tese de doutorado, visando descrever a comunicação entre sistemas autômatos. A modelagem em rede de Petri permite representar características fundamentais do sistema modelado com relação a sua estrutura e seu comportamento dinâmico, permitindo a sua modificação de acordo com a necessidade do sistema.

A modelagem gráfica de uma rede de Petri consiste de um grafo bipartido e dirigido, contendo dois tipos de componentes chamados de lugares e transições, interligados por segmentos orientados denominados de arcos. Cada arco possui um valor inteiro e não negativo, denominado peso do arco. Basicamente, a representação gráfica de uma rede de Petri é composta pelos seguintes elementos:

a) Lugares (*places*): representados através de círculos, são conhecido como componentes passivos, representando condições, recursos ou estado parcial do sistema.

b) Transições (*transitions*): representada por meio de barras ou retângulos, conhecidas como ativos da rede e representam ocorrências ou eventos que modificam o estado da rede, adicionando ou removendo marcas dos lugares.

c) Marcas (*token*): representam um valor específico de um estado ou lugar, como por exemplo, a quantidade de recursos disponíveis nos estados do sistema. São identificados por pequenos circulos pretos dentro dos lugares, e sua quantidade e localização varia dinamicamente em função da dinâmica da rede.

d) Arcos: são respresentados através de setas e indicam as conexões entre os lugares e as transições, sendo que nunca podem ser conectados dois lugares ou duas transições de forma consecutiva. Os arcos determinam o fluxo das marcas na rede e são representados por um número inteiro e não negativo, que indica o peso do arco, ou seja, a quantidade de marcas consumidas ou produzidas pela transição. Caso o arco não possua valor, o peso será assumido como 1.

Uma transição possui um certo número de lugares de entradas e saídas, os quais são conectados através de arcos, representando respectivamente uma pré-condição e uma pós-condição do evento observado. Assim, a realização de uma ação está relacionada a algumas pré-condições, ou seja, existe uma relação entre os lugares e as transições, o que possibilita a execução de uma ação. Em contrapartida, após a realização de uma ação, alguns lugares sofrerão alterações em suas informações (pós-condições).

A figura 2.1 traz os elementos básicos que compõem uma rede de Petri:

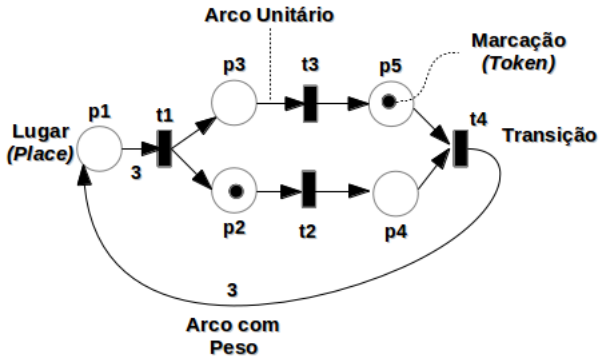

Figura 2.1: Elementos básicos que compõem uma rede de Petri.

2.2 Variações de redes de Petri

Diversas variantes ou classes de Redes de Petri surgiram após o trabalho apresentado por Carl Adam Petri [Petri, 1962]. Estas variações foram criadas para atender as necessidades de adaptação das Redes de Petri a especificidades de certas aplicações para as quais a sua utilização era necessária. O modelo original das redes de Petri é deficiente em dois elementos: funções complexas (fluxo de controle e de decisão) e temporização. Segundo [Dias, 2007], quanto ao grau de abstração, as redes de Petri podem ser classificadas em redes de Petri de alto nível e de baixo nível. Nas redes de Petri de alto nível, as marcas possuem semânticas como valores ou cores, viabilizando sua diferenciação, e os arcos podem fazer exigências em relação às marcas. Como exemplo de redes de Petri de alto nível podemos citar: redes de Petri Coloridas, redes de Petri Contínuas, redes de Petri estocásticas e redes de Petri Híbridas.

Já nas redes de baixo nível, as marcações não são diferenciáveis, dependendo apenas da estrutura da rede à qual estão associadas. Nestas redes, para modelar processos semelhantes ou

idênticos, torna-se necessário replicar as estruturas comuns aos processos. Entre as redes de baixo nível temos como exemplo as redes de Petri Elementares e as redes de Petri Lugar/Transição, conforme são apresentadas abaixo:

Definição 1 *Uma rede elementar é uma tripla N={P, T, F}, onde P e T são conjuntos disjuntos finitos de lugares e transições, respectivamente, e F* $\subseteq (P \times T) \cup (T \times P)$ *são o conjunto de fluxo de arcos, de tal forma que os seguintes requisitos devem ser satisfeitos:* ${}^\bullet x = \{y \mid (y,x) \in F\}$ *e* $x^\bullet = \{y \mid (x,y) \in F\}$ *para cada* $x \in P \cup T$*, sendo [Badouel et al., 2015]:*

a) Não existe auto-conflito: para cada $x \in P \cup T$, ${}^\bullet x \cap x^\bullet = \varnothing$.

b) Não existe transição isolada: para cada $t \in T$*, existe* $p \in P$*, tal que* $(p,t) \in F$ *ou* $(t,p) \in F$*, i.e.* ${}^\bullet t \cup {}^\bullet t \neq 0$.

c) Não existem transições equivalentes: para cada t, t' ∈ T, se ${}^\bullet t = {}^\bullet t'$ *e* ${}^\bullet t = t'^\bullet$*, então t = t'.*

d) A marcação de N é qualquer subconjunto de P.

Como as redes elementares são grafos bipartidos, a condição ${}^\bullet x \cap x^\bullet \neq 0$ que exclui auto-conflitos são realmente satisfeitos para cada elemento $x \in P \cup T$, desde que seja satisfeito para cada transição $x \in T$, ou para cada lugar $x \in P$.

Definição 2 *Uma rede de Petri Lugar/Transição marcada é um grafo direcionado e bipartido, permitindo que pesos sejam atribuídos aos arcos que conectam um lugar a uma transição ou vice-versa, sendo definida ainda como uma quíntupla [Murata, 1989]:*

$$N=(P,T,F,W,M)$$

onde:

a) $P=\{p_1, p_2, ..., p_m\}$*, que representa o conjunto finito de lugares;*

b) $T=\{t_1, t_2, ..., t_k\}$*, que corresponde ao conjunto finito de transições;*

c) $F \subseteq (P \times T) \cup (T \times P)$*, representa o conjunto de arcos da rede (relação de fluxo);*

d) $W: F \longrightarrow N$*, é o peso do arco;*

e) $M: P \longrightarrow N$*, corresponde ao número de marcas;*

f) $P \cap T = \oslash$ *e* $P \cup T \neq \oslash$*, em que P e T são conjuntos disjuntos.*

A relação de fluxo F esta contida no pré-conjunto e pós-conjunto de transições e lugares da rede. Sendo $x \in N$*, o pré-conjunto de x, denominado* ${}^\bullet x$*, e o pós-conjunto de x, denominado* $x^\bullet$*, podem ser definidos como:*

$$^{\bullet}x = \{y \in P \cup T \mid W(y,x) \geq 1\}$$
$$x^{\bullet} = \{y \in P \cup T \mid W(x,y) \geq 1\}$$

Cada lugar possui um quantidade k de marcas, onde k representa o número máximo de marcas que cada lugar possuirá. Os arcos ponderados possuem pesos os quais indicam a quantidade de marcas que serão ser retiradas ou inseridas em um dado lugar com o disparo da transição, sendo que o valor do peso é representado por um número inteiro positivo. Assim, para que uma transição possa disparar, se faz necessário que o número de marcas em cada lugar de entrada dessa transição seja igual ou superior ao peso do arco que liga o lugar a essa transição. Ocorrendo o disparo, é subtraído de cada lugar de entrada o número de marcas referente ao peso do arco que conecta o lugar a transição, bem como, em cada lugar de saída, será inserido o número de marcas igual ao peso do arco que liga a transição ao lugar.

2.3 Dinâmica da Rede

A alteração do estado de uma rede de Petri corresponde a evolução da marcação dos lugares, causada pelo disparo de transições. Assim, uma transição pode ou não estar habilitada, sendo que somente transições habilitadas podem disparar. O disparo de uma transição T consiste na retirada de uma ou mais marcações dos lugares de entrada da transição T, adicionando uma ou mais marcações nos lugares de saída da transição T, de acordo com o F e W. Segundo [Murata, 1989], uma transição t $\in$ T está habilitada por uma marcação M, se e somente se M $\geq$ W($^{\bullet}$t), ou seja, se o número de marcas nos lugares de entrada de t é maior ou igual ao peso dos arcos que liga os lugares a t. Em [Cardoso and Valette, 1997], quando uma transição sensibilizada for disparada, será obtida uma nova marcação M' tal que M'=M-$^{\bullet}$t+t$^{\bullet}$. O disparo de uma transição t, habilitada por uma marcação M gera uma marcação M' que pode ser representada por M[t>M'. Na figura 2.2 é mostrada a nova marcação resultante do disparo da transição t1.

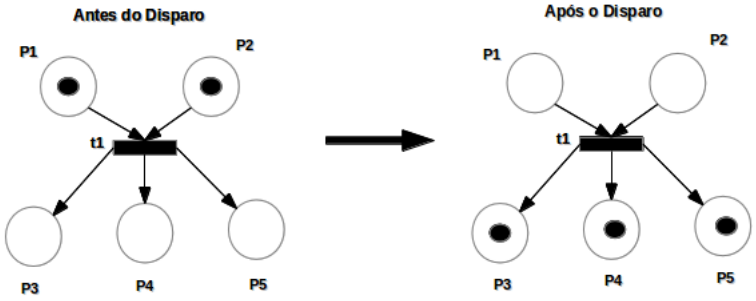

Figura 2.2: Dinâmica de funcionamento de uma rede de Petri.

2.4 Sequência de disparo

Definição 3 *Denomina-se sequência de disparo (s) quando uma sequência s=t_1,...,t_k, é disparável a partir de M_0, levando a uma marcação M_k.*

$$M_0 \xrightarrow{s} M_k$$

Observando a rede de Petri da figura 2.1, verifica-se que a transição t_2 está sensibilizada pela marcação M_0=[0 1 0 0 1] e pode disparar, levando a marcação M_1=[0 0 0 1 1]. Após o disparo de t_2, a transição t_4 por sua vez pode disparar, gerando a marcação M_2=[1 0 0 0 0]. Assim, a sequência s=t_2t_4, é disparável a partir de M_0.

$$M_0 \xrightarrow{t_2t_4} M_2$$

2.5 Equação Fundamental

A equação fundamental representa o estado que a rede de Petri atingirá após uma sequência de disparos das transições, permitindo a análise comportamental da rede. A equação é definida por:

$$M=M_0 + C.\overline{s}$$

Onde:

a) M_0: representa a marcação inicial da rede;

b) C: corresponde a matriz de incidência;

c) $\overline{s}$: para calcular uma nova marcação após o disparo de uma sequência de transições, define-se o vetor de Parikh *($\overline{s}$)* [Cardoso and Valette, 1997]. Esse vetor descreve a sequência de disparos de transições, isto é, sendo σ=t_1,...,t_k, uma sequência de disparos de transições, então $\overline{s}$=($\#_{t1}\sigma$,...,$\#_{tk}\sigma$), em que $\#_{ti}$ é o número de ocorrências de t_i em t_1,...,t_k.

A equação fundamental permite a análise da acessibilidade das marcações e a representação dos aspectos comportamentais da rede, uma vez que esta descreve a dinâmica da inserção e retirada de marcas nos lugares e a sequência de disparos de transições.

Apesar da equação fundamental ser condição necessária para estudo do comportamento da rede, torna-se limitada no que diz respeito a preservação das marcações alcançáveis da rede original, uma vez que o vetor $\overline{s}$ poderá ser positivo, porém não disparável.

2.6 Notação Matricial de uma rede de Petri

Uma rede de Petri pode ser representada algébricamente por uma matriz de incidência, representando a relação entre os lugares e as transições. O comportamento dinâmico destas matrizes podem ser descritas por um sistema linear [Cardoso and Valette, 1997].

O fluxo F e o peso W dos arcos podem ser representados por duas matrizes que estabelecem as transições da rede e sua conexão com os lugares da rede. A matriz de entrada Pré ($^\bullet$x) representa os arcos orientados que conectam lugares a transições, sendo que Pre(p,t) é o peso do arco (p,t). Caso Pré(p,t)=0, então não existirá um arco que ligue um lugar p a uma transição t. Já a matriz de saída Pós (x$^\bullet$) mostra os arcos orientados que ligam as transições com lugares, sendo que Pós(p,t) é o peso do arco (t,p). Se Pós(p,t)=0, não existirá um arco entre a transição t e o lugar p. Para a figura 2.1, as matrizes Pré e Pós serão:

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad Pos = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 3 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

A matriz de incidência define todas as possíveis interconexões entre os lugares e as transições de uma rede de Petri. A matriz de incidência é uma matriz $C = [a_{ij}]$ de tamanho n x m, onde n é o número de lugares e m é o número de transições. A matriz C será definida pela diferença entre a matriz pós-conjunto e a matriz pré-conjunto:

$$C = x^\bullet - {}^\bullet x$$

A seguir é exemplificada a matriz de incidência da rede da Figura 2.1:

$$C = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -3 & 0 & 0 & 3 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array}$$

Na matriz de incidência, cada coluna corresponde à modificação da marcação através do disparo da transição associada. Assim, da matriz C pode-se concluir que quando do disparo da transição t_2, por exemplo, será retirada uma ficha do lugar p_2 e será adicionada uma ficha no lugar p_4.

2.6.1 Limitação da Matriz de Incidência

A matriz de incidência apresenta limitações quando uma rede de Petri possui auto-laços *(self-loops)* em sua estrutura.

Definição 4 *Um auto-laço é um par formado por um lugar p e um transição t, tal que p é, ao mesmo tempo, entrada e saída de t.*

Antes de apresentar essa limitação, se faz necessário definir os conceitos de não reflexividade *(nonreflexivity)* e reflexividade *(reflexive)*:

Definição 5 *A propriedade de não reflexividade, é uma característica que, para qualquer par (p,t),* $p \in {}^\bullet t \Longrightarrow p \notin t^\bullet$ *e* $p \in t^\bullet \Longrightarrow p \notin {}^\bullet t$*, ou seja,* ${}^\bullet t \cap t^\bullet = \varnothing$.

A rede na qual todos os pares *(p,t)* são não reflexivos são chamados de redes livres de laços *(self-loop-free)* ou redes de Petri puras. Na figura 2.3 é apresentado uma rede de Petri não reflexiva:

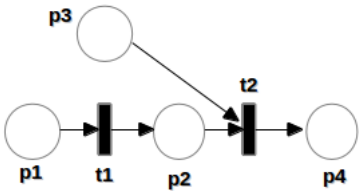

Figura 2.3: Rede de Petri pura.

Da rede de Petri não reflexiva, apresentada na figura 2.3 é extraído a sua matriz de incidência, conforme se vê abaixo:

$$C = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \end{array}$$

Definição 6 *A propriedade da reflexividade, é quando um lugar p é ao mesmo tempo, uma entrada e uma saída de uma transição t, ou seja,* ${}^\bullet t \cap t^\bullet \neq \varnothing$.

Quando as redes de Peti são reflexivas, serão também chamados de redes de Petri impuras. Na figura 2.4 é representado uma rede de Petri reflexiva:

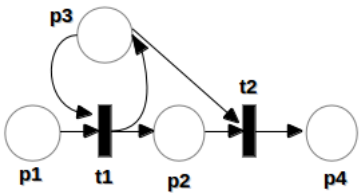

Figura 2.4: Rede de Petri impura.

Da rede de Petri reflexiva, apresentada na figura 2.4 é extraído a sua matriz de incidência, conforme se vê abaixo:

$$C = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \end{array}$$

Verifica-se que, embora as redes das figuras 2.3 e 2.4 sejam diferentes, as suas matrizes de incidência correspondentes são idênticas. Assim, o estudo de uma rede de Petri através de uma matriz de incidência, torna-se limitado e ineficiente dado a possibilidade dessa rede possuir a propriedade reflexiva na estrutura, levando a conclusões errôneas da rede, em análises a partir da matriz de incidência.

2.7 Propriedades da rede de Petri

Segundo [Murata, 1989], existem dois tipos de propriedades estudadas nos modelos de rede de Petri: comportamentais e estruturais.

2.7.1 Propriedades Comportamentais

As propriedades comportamentais ou dinâmicas, se referem a elementos relacionados à dinâmica da rede e, portanto, dependem da marcação inicial dos lugares. Entre as propriedades comportamentais podemos destacar:

Definição 7 *(Alcançabilidade): o conjunto de marcações acessíveis de uma rede de Petri (N) marcada é o conjunto das marcações alcançáveis (M_k) a partir da marcação inicial (M_0), por meio de uma sequência de disparos s.*

$$N = \{\, M_0, \exists s\ M_k \xrightarrow{s} M_0 \,\}$$

Se todas as marcações iniciais de uma rede de Petri forem decorrente da marcação inicial, então a rede é dita alcançável. Verificar a alcançabilidade de uma marcação é determinar se uma marcação M_k é alcançável a partir de uma marcação inicial M_0.

Apesar de não ser objeto do presente estudo, saber se uma determinada marcação é alcançável a partir da marcação inicial da rede é uma questão relevante, sendo definido como problema de alcançabilidade em redes de Petri.

Definição 8 *(Limitabilidade): a rede é dita k-limitada se o número de fichas de cada lugar não ultrapassar um número finito k, ou seja, se para todo lugar p_i, a restrição $M(p_i) \leq k$ é respeitada em todas as marcações acessíveis da rede. Assim, a rede será k-limitada, onde k representa a capacidade máxima de cada lugar.*

$$\forall p \in P \; e \; \forall M \in N, \; M(p_i) \leq k$$

Na figura 2.5, a rede é limitada, pois observa-se que as fichas nos lugares da rede nunca excedem ao inteiro 2, portanto, é uma rede 2-limitada. Se *k=1*, a rede 1-limitada é chamada de rede segura *(safe net)*.

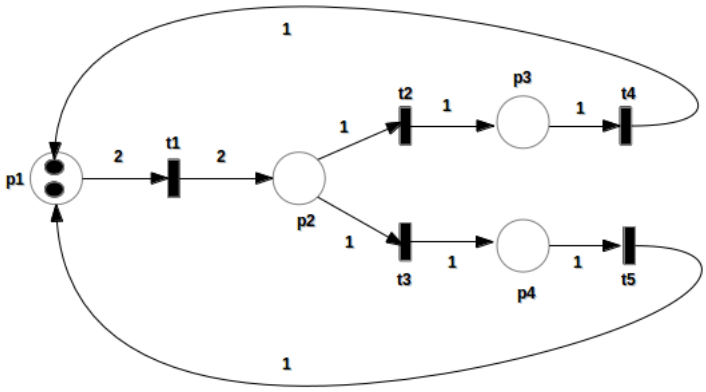

Figura 2.5: Exemplo de uma rede K-limitada.

Definição 9 *(Vivacidade): uma rede de Petri será viva se e somente se para cada transição $t \in T$, t pode ser disparada a partir de qualquer marcação M' do conjunto de marcações acessíveis por meio de uma sequência s de disparo.*

$$\forall M' \in N, \; \exists s \mid M' \xrightarrow{st}$$

O *deadlock* é uma característica que pode ocorrer na rede de Petri quando em um dos estados alcançáveis não possuir transições disparáveis, impedindo a evolução da rede. Na figura 2.6 é apresentado uma rede não viva, devido a transição *t1* não estar habilitada para o disparo, pois nem todos os lugares de entrada, neste caso *p2*, possui ficha, caracterizando um *deadlock*. A vivacidade de uma rede, depende da garantia de uma rede livre de bloqueios, evitando prejuízos ao rendimento do sistema.

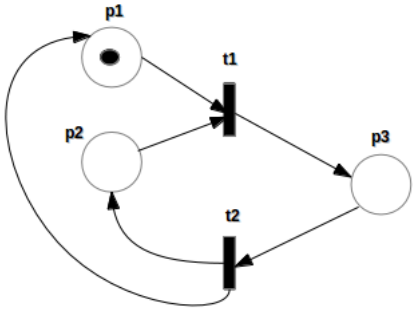

Figura 2.6: Exemplo de uma rede Não Viva.

Definição 10 *(Reversibilidade ou Reinicialização): se dá quando uma marcação inicial volta a ocorrer ao longo de uma sequência de disparos, ou seja, permite verificar se é possível, dado uma sequência de disparos, retornar à marcação inicial da rede.*

$$\forall M' \in N, \exists s \mid M' \xrightarrow{s} M_0$$

A figura 2.7 mostra uma rede de Petri reversível, pois através da sequência de disparos *t2, t4* e *t5* ou *t1, t3* e *t5*, é possível retornar à marcação inicial (M_0).

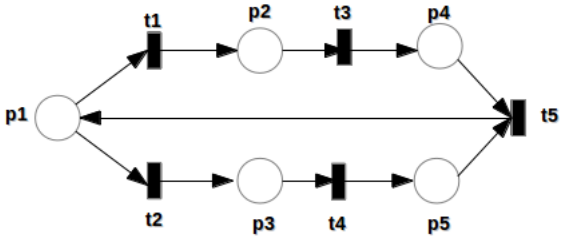

Figura 2.7: Exemplo de Reversibilidade.

É importante destacar que as propriedades comportamentais têm alto custo computacional, uma vez que dependem da enumeração do espaço de estados, a partir da marcação inicial (M_0).

2.7.2 Propriedades Estruturais

As propriedades estruturais são independentes da marcação, estando atreladas a estrutura da rede, mas que podem fornecer informações adicionais sobre a dinâmica da rede. Assim, cita-se duas propriedades estruturais:

Definição 11 *(Componentes Conservativos - Invariantes de Lugar): de acordo com [Cardoso and Valette, 1997], um componente conservativo de uma rede de Petri é o conjunto de lugares $p \in P$ correspondentes aos elementos não nulos do vetor coluna $\overrightarrow{f}$, solução da equação:*

$$C \,.\, \overrightarrow{f}^T = 0, \; \overrightarrow{f} > 0$$

Sendo $\overrightarrow{f}$ a solução da equação anterior, o invariante de lugar será dado por:

$$M \,.\, \overrightarrow{f}^T = M_0 \,.\, \overrightarrow{f}^T, \forall M \in N$$

Uma invariante de lugar é uma função linear de marcação dos lugares em que o valor é uma constante que depende apenas da marcação inicial da rede. Esta propriedade permite verificar se há uma restrição no que se refere a marcação de alguns ou todos os lugares em todos os estados alcançáveis da rede, citando como exemplo, a conservação ponderada do número total de marcas para um conjunto determinado de lugares.

Na figura 2.8 observa-se uma rede formada pelos lugares p_1 e p_2, e pelas transições t_1 e t_2, sendo que a soma das marcações $\mathrm{M}(\mathrm{p}_1)$ com $\mathrm{M}(\mathrm{p}_2)$ é sempre 1 para a $M_0 = [1\ 0\ 3\ 0\ 1]^T$. O disparo de t_1 não altera o valor da soma bem como o disparo de t_2, apesar da marcação dos lugares serem modificado nos disparos das transições. Da mesma forma, o disparo das transições t_3 e t_4 não alteram o resultado da soma, pois as marcações p_1 e p_2 não se modificam:

$$\forall \mathrm{M} \in \mathrm{N},\ \mathrm{M}(\mathrm{p}_1) + \mathrm{M}(\mathrm{p}_2) = \mathrm{M}_0(\mathrm{p}_1) + \mathrm{M}_0(\mathrm{p}_2)$$

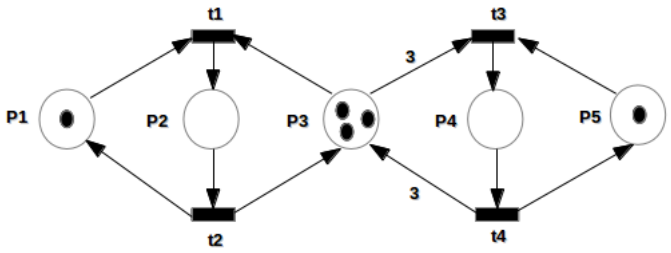

Figura 2.8: Propriedade Estrutural - Invariante de Lugar.

A forma linear *$M(p_1) + M(p_2)$* é chamada de invariante de lugar, pois a soma das marcas se conserva para estes lugares, e os lugares *p_1 e p_2* formam um componente conservativo da rede. Assim, uma rede de Petri será conservativa se todos os seus lugares pertencem a um componente conservativo.

Definição 12 *(Componentes Repetitivos - Invariantes de Transição): segundo [Cardoso and Valette, 1997], um componente repetitivo de uma rede de Petri, é dado pelo vetor coluna $\overrightarrow{s}$, correspondente a solução da equação:*

$$C \,.\, \overrightarrow{s}^T = 0,\ \overrightarrow{s} > 0$$

Uma rede de Petri é repetitiva se todas as transições $t \in T$ pertencem a um componente repetitivo. A sequência de transições t associadas ao vetor $\overrightarrow{s}$ é um invariante de transição da rede. Um invariante de transição é definido pelo vetor $\overrightarrow{s}$ de dimensão igual ao número de transições da rede, onde cada componente indica o número de vezes que a transição correspondente deve ser disparada.

Assim, uma invariante de transição corresponde um conjunto de transições cuja sequência de disparos resulte em um ciclo, ou seja, uma sequência de eventos que podem se repetir indefinidamente.

Na figura 2.8, tendo como $M_0 = [1\ 0\ 3\ 0\ 1]^T$, o disparo de t_3 e t_4 retorna para a marcação inicial, ou seja, t_3 e t_4 é uma invariante de transição dado que após o disparo da sequência, a rede retorna ao estado anterior, gerando um comportamento cíclico das marcações, não alterando a marcação da rede. As transições t_3 e t_4 formam um componente repetitivo estacionário. Os invariantes de transição dependem da marcação atual da rede, pois os disparos dependem diretamente das pré-condições das transições.

2.8 Métodos de Análise

Segundo [Murata, 1989], os métodos de análise de redes de Petri podem ser classificados em três grupos: grafo de alcançabilidade e grafo de cobertura; matriz de incidência e equações de estado; e técnicas de redução ou decomposição.

Em uma rede de Petri, as marcações alcançáveis podem ser organizadas em uma estrutura de grafo, em que os nós estão associados a uma marcação e cada arco de uma transição transforma a marcação do nó de origem na marcação do nó destino. Dada uma marcação inicial M_0 de uma rede de Petri, serão obtidas novas marcações de acordo com o número de transições habilitadas. A cada nova marcação pode-se alcançar mais marcações, resultando numa representação de marcações em forma de árvore, em que as folhas são as marcações geradas de M_0 e suas sucessoras. Cada ramo da árvore simboliza o disparo de uma transição, resultando na mudança de uma marcação para outra. Assim, a árvore possuirá todas as marcações alcançáveis e as sequências de disparo das transições. No que diz respeito a enumeração de espaços de estado, o conjunto de marcações de uma rede de Petri pode ser finito ou limitado e infinito ou ilimitado, sendo que para o primeiro caso, será construído um grafo de alcançabilidade e para o segundo, um grafo de cobertura.

O grafo de alcançabilidade representa o conjunto de estados ou marcações alcançáveis de uma rede de Petri a partir de uma marcação inicial M_0. Cada vértice é uma marcação e cada arco representa o disparo da transição que leva a uma nova marcação. A marcação inicial é a raiz do grafo de alcançabilidade.

Para construir um grafo de alcançabilidade utiliza-se o algoritmo apresentado em [Karp and Miller, 1969]:

Algoritmo 1: Algoritmo para construção do grafo de alcançabilidade:

1 Rotular a marcação raiz M_0 como nova;
2 **enquanto** *existirem marcações novas* **faça**
3 Selecionar uma nova marcação M;
4 **se** *nenhuma transição está habilitada para M* **então**
5 Identifique-a como morta ou final;
6 **senão**
7 M é idêntica a uma marcação existente no caminho da raiz até M, rotule-a como antigo e selecione uma outra marcação nova;
8 **fim**
9 **enquanto** *existirem transições habilitadas em M* **faça**
10 Obtenha a marcação M' que resulta do disparo da transição;
11 **se** *no caminho desde a raiz até M existe uma marcação M" tal que, para todo lugar p ∈ P, M'(p) ≥ M"(p) e M' ≠ M"* **então**
12 Substitua M'(p) por ω para cada p tal que M'(p) ≥ M"(p);
13 **senão**
14 Volte ao passo 9;
15 **fim**
16 **fim**
17 **fim**

Na figura 2.9 (a) é apresentado uma rede de Petri e na figura 2.9 (b) é mostrado o resultado do grafo de alcançabilidade dessa rede aplicando o algoritmo de [Karp and Miller, 1969].

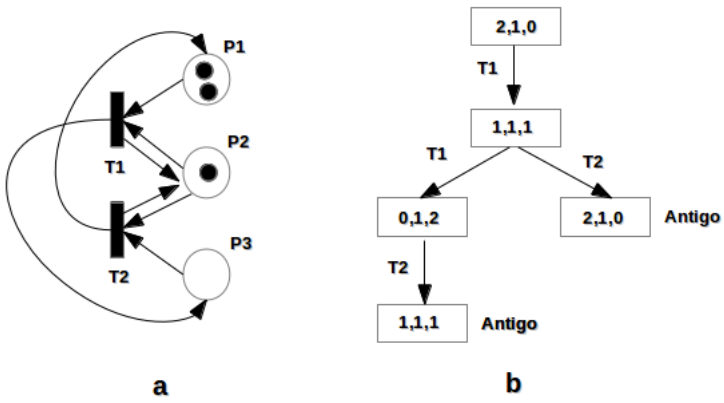

Figura 2.9: Grafo de alcançabilidade (b) de uma rede de Petri (a).

O grafo de cobertura é um gráfico que busca representar finitamente um número infinito de marcações. Este grafo é obtido a partir do grafo de alcançabilidade o qual pode crescer infinitamente. Para tornar o grafo finito, utilizamos o símbolo especial ω para representar o número de marcações que pode ser considerado arbitrariamente grande e que esta sujeito a quatro regras aritméticas para todo inteiro α:

$$\alpha < \omega$$
$$\omega \leq \omega$$

$$\omega + \alpha = \omega$$
$$\omega - \alpha = \omega$$

Na figura 2.10 (a) é representado uma rede de Petri e na figura 2.10 (b) é apresentado um grafo de cobertura, também aplicando o algoritmo de [Karp and Miller, 1969].

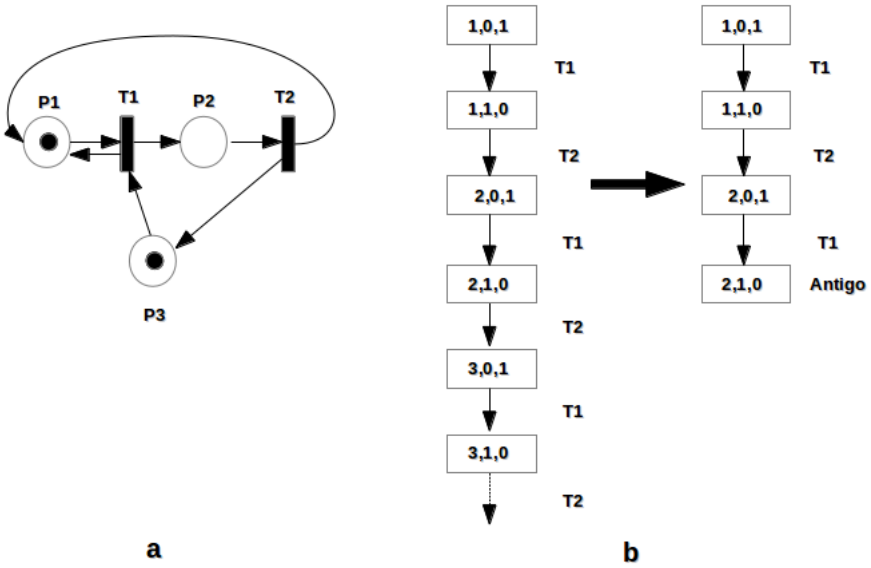

Figura 2.10: Grafo de cobertura (b) de uma rede de Petri (a).

2.9 O problema da técnica do grafo de alcançabilidade

Em sistemas complexos, a construção do grafo de alcançabilidade pode ser inviável em virtude do seu tamanho. Isso ocorre devido a explosão dos espaços de estados causados pela estrutura da rede, pela concorrência e pelo número de nós que podem habilitar as transições. O grafo de alcançabilidade lista todas as marcações alcançáveis, o que pode se tornar um problema em virtude da explosão dos espaços de estados além do que nem sempre é necessário conhecer todas as marcações alcançáveis.

O problema do jantar dos filósofos chineses, desenvolvido por [Dijkstra, 1971], é frequentemente utilizado para ilustrar o uso ineficiente dos recursos compartilhados bem como para mostrar o aumento do espaço de estados. O problema original descreve cinco filósofos sentados em uma mesa redonda, sendo que ao centro existe um prato de espaguete e apenas um palito, o qual é colocado entre cada par de filósofos adjacentes. Os filósofos possuem apenas dois estados, o de meditar e o de comer. Inicialmente, todos os filósofos estão no estado de meditação. Cada filósofo deve pegar os palitos do lado esquerdo e direito para entrar no estado de comer, podendo pegar os palitos somente quando eles estiverem disponíveis. Após o filósofo terminar de comer, ele devolverá os palitos a mesa e retornará para o estado de meditação. A medida que aumentamos o número de filósofos sentados a mesa, teremos um aumento nas marcações

alcançáveis. Por exemplo, se partirmos de cinco filósofos sentados a mesa, e formos adicionando filósofos, notaremos um aumento nas marcações alcançáveis, conforme segue na tabela 2.1:

Tabela 2.1: Marcações alcançáveis para um número variável de filósofos.

Número de Filósofos	Marcações Alcançáveis
5	82
6	198
7	478
8	1152

Assim, verifica-se que com o aumento do número de elementos da rede, neste caso o de filósofos, o grafo de alcançabilidade crescerá exponencialmente, produzindo informações intratáveis, pois nem sempre será necessário conhecer todas as marcações alcançáveis.

2.10 Conclusão

As redes de Petri possuem um eficiente formalismo matemático que pode ser utilizado na representação de sistemas dinâmicos. O presente capítulo apresentou os conceitos fundamentais sobre redes de Petri, a sua dinâmica e a equação fundamental, destacando-se ainda as variações das redes de Petri, sendo de interesse do presente trabalho as redes de Petri Lugar/Transição segura.

Foi apresentado também a notação matricial de uma rede de Petri, atravé da matriz de incidência, sendo ainda demonstrado a sua desvantagem como ferramenta de análise, em virtude da possibilidade de existência de auto-laço em sua estrutura. Em seguida, foi destacado as propriedades comportamentais e estruturais e sua importância no estudo das redes de Petri.

Por fim, foi apresentado os métodos de análise existentes para análise das redes de Petri, destacando-se a técnica do grafo de alcançabilidade e a sua problemática quando utilizada como ferramenta de análise.

No capítulo a seguir, será estudado todo o processo de desdobramento de uma rede de Petri, ou seja, o processo através da qual uma rede cíclica é transformada em uma rede acíclica, que possui todas as marcações alcançáveis da rede original.

Capítulo 3

Desdobramento

Neste capítulo serão apresentadas definições relacionadas ao processo do desdobramento: processo de ramificação, prefixo finito completo, configurações, eventos de corte e algoritmo de desdobramento.

3.1 Conceitos Fundamentais

O processo de desdobramento *(unfolding)*, apresentado em [McMillan and Probst, 1995], tem como objetivo oferecer uma alternativa de análise sem gerar (enumerar) o espaço de estados, quando comparado com o grafo de alcançabilidade. A técnica de desdobramento, a partir de uma rede de Petri marcada e segura, gera uma rede de ocorrências. Portanto segundo [McMillan and Probst, 1995], a técnica de desdobramento cria uma nova rede, de complexidade menor que o grafo de alcançabilidade, permitindo evitar a explosão de estados do sistema modelado com redes de Petri. A técnica de desdobramento descreve a dinâmica da execução e simulação de uma rede de Petri Lugar/Transição segura a partir de uma determinada marcação inicial. Esse processo é realizado através do mapeamento da rede original segura para uma rede de Ocorrência, preservando as informações de concorrência, gerando uma rede acíclica, finita e que possui todas as marcações alcançáveis da rede original. Essa dinâmica é realizada por meio de uma rede de ocorrência, cuja estrutura final dependerá da rede original e da marcação inicial.

3.1.1 Relações de Ordem

Definição 13 *Segundo [Taubin et al., 1998], sendo N=(P,T,F,W,M) uma rede acíclica e* x_1, $x_2 \in P \cup T$, *teremos as seguintes relações de ordem:*

a) O nó x_1 *precede o nó* x_2, *ou seja,* $x_1 \preceq x_2$, *se existe um caminho que conduz de* x_1 *para* x_2. *Na figura 3.1 (a), temos um exemplo de precedência entre os nós* $p_1 \preceq p_2 \preceq p_3$.

b) Dois nós (lugar ou transição), x_1 e x_2, estão em conflito, ou seja, $x_1 \# x_2$, se existem transições distintas t_1, $t_2 \in T$ tal que ${}^\bullet t_1 \cap {}^\bullet t_2 \neq 0$, $t_1 \preceq x_1$ e $t_2 \preceq x_2$. A figura 3.1 (b) mostra o conflito entre as transições $t_1 \# t_2 \# t_3$, pois ambas as transições tem p_1 como um dos elementos do pré-conjunto.

c) Dois nós (lugar ou transição), x_1 e x_2, são concorrentes, ou seja, $x_1 \parallel x_2$, se não são precedentes e não estiverem em conflito. Na figura 3.1 (c), temos a concorrência entre os lugares $p_2 \parallel p_3 \parallel p_4$.

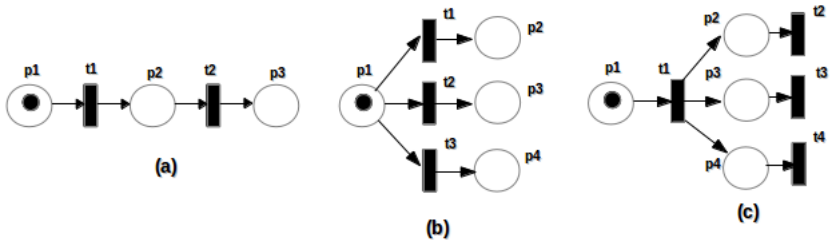

Figura 3.1: Relações de Ordem: (a) Precedência, (b) Conflito e (c) Concorrência.

As redes de Petri acíclicas são especificadas pelas relações de ordem, facilitando sua análise.

3.1.2 Rede de Ocorrência *(Occurrence Net - ON)*

Definição 14 *Segundo [Taubin et al., 1998], uma rede condição evento N=(B,E,F) será uma rede de ocorrências desde que satisfaça o que segue abaixo:*

a) Cada lugar tem no máximo uma transição de entrada, ou seja, $\forall b \in B$, ${}^\bullet b \leq 1$.

b) F é acíclico.

c) Nenhum evento $e \in E$ está em autoconflito.

Uma rede de ocorrência permite, a partir de qualquer nó, percorrer o caminho inverso até o nó de origem. Na rede de ocorrência apresentada na figura 3.2, as relações de ordem entre os nós são mutuamente exclusivas.

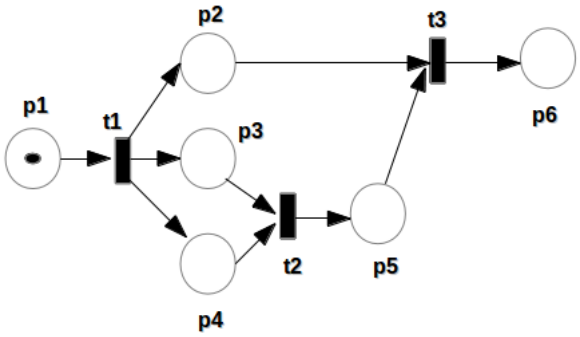

Figura 3.2: Rede de Ocorrência.

Na figura 3.3, os lugares p_2 e p_3 encontram-se em conflito, pois o lugar p_1 permite disparar uma das transições, t_1 ou t_3. Nesta rede, podem ocorrer dois ciclos, o primeiro p_1, t_1, p_2, t_2 e p_1; e o segundo ciclo p_1, t_3, p_3, t_4 e p_1. A presença desses dois ciclos faz com que seja impossível determinar a relação $x_1 \preceq x_2$ ou $x_2 \preceq x_1$ entre os dois nós x_1 e x_2. Em uma rede de ocorrências, devido a aciclicidade este problema não acontece e as relações de ordem não ocorrem ao mesmo tempo.

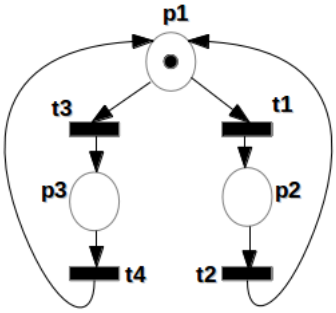

Figura 3.3: Rede de Petri Lugar-Transição.

A vantagem principal da rede de ocorrências está na representação de todos os comportamentos que a rede pode gerar, evitando a enumeração do espaço de estados que muitas vezes pode levar a uma explosão combinatória do espaço de estados. Execuções de redes cíclicas podem levar à geração de redes de ocorrência infinitas. As mesmas poderão ser truncadas em subredes chamadas de prefixo finito completo, as quais detêm todas as informações referentes a marcações e sequências de disparos de transições da rede original. Esse prefixo finito completo é também conhecido como rede desdobrada *(unfolding)*. Segundo [Esparza and Schröter, 2001], para truncar uma rede de ocorrência é fundamental a noção de configuração local e de marcação básica.

3.1.3 Homomorfismo *(Homomorphism)*

Definição 15 *Conforme [Bonet et al., 2014], o homomorfismo é uma função de rotulamento de uma rede de ocorrência dada uma rede de Petri, ou, de acordo com [Benito, 2015], o*

homomorfismo é uma função de rotulamento entre dois grupos, onde as propriedades do grupo domínio são preservadas no grupo da imagem. Assim, sendo (D, ·) e (E,∗) dois grupos e φ uma função de D em E, diz-se que φ é um homomorfismo se:

$$(\forall\, x,y \in D)\colon \varphi(x \cdot y) = \varphi(x) * \varphi(y)$$

3.1.4 Processo de Ramificação *(Branching Processes)*

Definição 16 *Segundo [Taubin et al., 1998], o processo de ramificação consiste de etiquetamentos de uma rede que mostram todos os caminhos possíveis, mantendo as informações sobre conflito e concorrência. A ramificação de uma rede N é representada pelo par β=(ON,φ), onde ON=(B,E,F) é uma rede de ocorrência e φ representa o homomorfismo da rede de ocorrência tal que [Bonet et al., 2014]:*

a) φ(B)⊆P e φ(E)⊆T (mantém a natureza dos nós).

b) ∀e∈E, a restrição de φ para $^\bullet$e é uma função bijetora entre $^\bullet$e (em N) e $^\bullet$φ(e) (em β), e simetricamente para e$^\bullet$ e φ(e)$^\bullet$ (mantém o ambiente das transições).

c) O conjunto de elementos mínimos de B∪E que possuem seu pré-conjunto vazio é denominado Min(ON). A restrição de φ para Min(ON) é uma função bijetora entre Min(ON) e M_0. Considerando redes em que não existem transições com pré-conjunto vazio, Min(ON) só podem ser condições (B), então φ{|Min(ON)|}=M_0.

d) ∀e_1,e_2∈E, se $^\bullet e_1 = {}^\bullet e_2$ e φ(e_1)=φ(e_2) então e_1=e_2 (não existe redundância nas transições).

O processo de ramificação (β) pode ser infinito, porém é possível trunco-la em várias sub-redes denominadas de prefixo finito, as quais possuem todas as marcações alcançáveis da rede.

Definição 17 *Segundo [Bonet et al., 2014], um processo de ramificação β'=(ON',φ') de uma rede Petri é um prefixo finito de β=(ON,φ), denotado por $\beta' \subseteq \beta$ se ON'=(B',E',F') é uma subrede de ON=(B,E,F), tal que:*

a) Min(ON) ∈ ON' (contém todos os mínimos elementos).

b) Se e ∈ E' e (b,e) ∈ F ou (e,b) ∈ F, então b ∈ B'.

c) Se b ∈ B' e (e,b) ∈ F, então e ∈ E'.

d) φ' é uma restrição de φ para B' ∪ E'.

A figura 3.4 mostra um exemplo de uma rede de Petri segura com um processo de ramificação, onde φ dos lugares e transições são indicados pelos etiquetamentos internos dos nós.

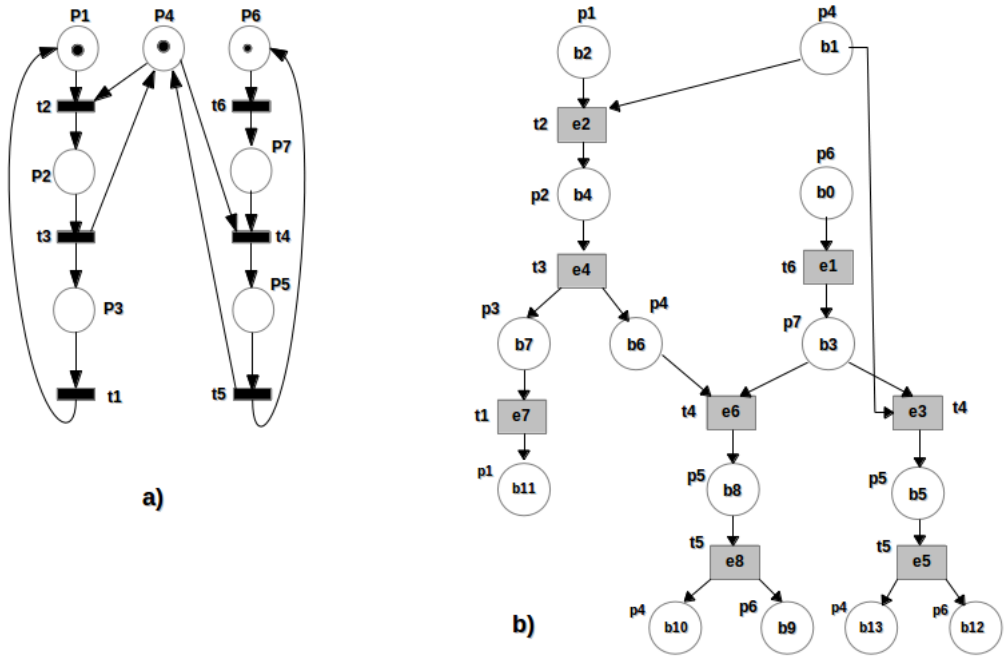

Figura 3.4: Rede de Petri segura (a) e Processo de ramificação (b).

3.1.5 Configuração *(Configuration)*

Definição 18 *A configuração C de uma rede de Ocorrência é um conjunto de transições T representando uma possível execução parcial da rede [McMillan and Probst, 1995]. A configuração deve ser causal, ou seja, se começar com um evento que está na configuração e traçar um caminho de retorno, todos os eventos que encontrarmos estarão na configuração. Também deve ser livre de conflito, não podendo dois eventos serem originários do mesmo lugar. Assim, a configuração deve ser causalmente fechada e livre de conflito, tal que [Bonet et al., 2014]:*

a) para cada $e \in C$*, a configuração C contém e, e todos os seus precedentes.*

b) $\forall e_1 e_2 \in C$, $e_1 \neq e_2$, *sendo* ${}^\bullet e_1 \cap {}^\bullet e_2 = 0$.

O prefixo mostrado na figura 3.4 (b), $\{e_2, e_4, e_1, e_6\}$ é uma configuração, enquanto que $\{e_2, e_4, e_8\}$ e $\{e_1, e_6, e_3\}$, não são configurações poi $e_6 \preceq e_8$, e $e_1 \# e_6$. Segundo [McMillan and Probst, 1995], uma configuração descreve um multiconjunto parcialmente ordenado de transições da rede inicial.

Segundo [Benito, 2015], para cada evento $e \in$ E, a configuração $[e] = \{e' \mid e' \preceq e\}$ é denominada de configuração local de e e $<e> = [e] \setminus \{e\}$, representa o conjunto de predecessores causais de e. Na figura 3.4 (b), a configuração local para o evento e_8 é $[e_8]=\{e_2, e_4, e_1, e_6, e_8\}$.

Uma marcação *Mark* (C) pode ser associada a uma configuração, que corresponde a uma marcação alcançável a partir de M_0, depois que todas as transições de C forem disparadas, sendo *Mark* (C) = φ ((Min(ON) $\cup$ C$^\bullet$)$\setminus$ $^\bullet$C) . A sequência de eventos $\{e_2, e_4, e_1, e_6\}$ da figura 3.4 (b), representa uma configuração, sendo a marcação alcançável a partir dessa configuração $\varphi(c_7,c_8)=p_3p_5$ ou $Mark([e_6])=p_3p_5$, conforme figura 3.4 (a).

3.1.6 Prefixo Finito Completo

Definição 19 *Segundo [Bonet et al., 2014], para cada rede existe um único processo de ramificação máximo, que possui todas as marcações alcançáveis, preservando a concorrência e o conflito da rede, sendo este processo de ramificação denominado de prefixo finito completo (β'). O prefixo β' de β será completo se, para cada marcação alcançável M, existe uma configuração $C \in \beta$', tal que:*

a) Mark (C) = M.

b) para cada transição t habilitada para M, existe uma configuração $C \cup \{e\}$, tal que $e \notin C$ e $\varphi(e)=t$.

Na figura 3.4 (b) temos um prefixo finito completo, que foi cortado do desdobramento sem perder informações. O corte é realizado identificando eventos que levam a uma marcação já representada, sendo esses eventos denominados de eventos ou transição de corte.

3.1.7 Evento de Corte *(Cut-off)*

Definição 20 *Em uma rede de ocorrência, dado um evento e, ele será um evento de corte (cut-off) de β, se β contém um outro evento e', que satisfaça o seguinte [Bonet et al., 2014]:*

a) Mark[e] = Mark[e']

b) [e'] < [e].

Uma ordem parcial $\prec$ sobre as configurações de um processo de ramificação é uma ordem adequada desde que a escolha do evento a ser disparado seja aquele cuja configuração local seja mínimia. Sendo [e'] < [e], teremos que [e'] será menor que [e], desde que [e'] tenha um número menor de eventos, ou seja, [e']<[e]. Assim, nota-se que essa ordem parcial é uma ordem adequada para a rede, ou seja, [e'] $\prec$ [e]. Ainda na figura 3.4 (b), os eventos e_5 e e_7, tratam-se de eventos de corte, pois levam à marcação inicial M_0=(p_1,p_4,p_6). No tocante ao evento e_8, é também um evento de corte, uma vez que a marcação alcançada p_1,p_4,p_6, é a mesma alcançada pelo evento e_4, sendo a $Mark[e_8]$=$Mark[e_4]$, porém [e_4] $\prec$ [e_8], ou seja, e_4 será executado primeiro em virtude da ordem adequada. Removendo todos os eventos transições e lugares que sucedem os eventos transições de corte da rede de ocorrência, resultará em uma rede de desdobramento.

3.2 Algoritmos de Desdobramento

Em 1995, McMillan em [McMillan and Probst, 1995], desenvolveu uma técnica para evitar o problema de explosão de estados em sistemas modelados com redes de Petri de estado

finito. A técnica baseou-se no conceito de desdobramento da rede, através do processo de ramificação da rede. McMillan propôs um algoritmo para a construção da parte inicial finita do processo de desdobramento contendo todas as informações sobre os estados alcançáveis, tendo sido denominado de prefixo finito completo.

A desvantagem do algoritmo de McMillan está na geração de prefixos de tamanhos muitas vezes maiores do que o necessário. Assim, em 2002, conforme [Esparza et al., 2002], foi proposto um algoritmo melhorado de Esparza, Römer e Vogler, denomidado algoritmo ERV *Unfolding*, que gera um prefixo completo mínimo em relação ao algoritmo de McMillan. O algoritmo ERV *Unfolding* é apresentado a seguir:

Algoritmo 2: Construção do Prefixo Finito Completo:

Dados: Uma rede N, onde $M_0=\{p_1,...,p_k\}$
Resultado: Um Prefixo Finito Completo do *Unfolding* Unf

```
1  Unf ← lugares de M0;
2  pe ← transições habilitadas por M0;
3  cut-off ← 0;
4  enquanto pe ≠ 0 faça
5      Escolha evento e=(t,X) de pe tal que [e] seja mínima respeitando a ordem ≺;
6      se [e] ∩ cut-off = 0 então
7          Adicione e e seus pós-conjunto em Unf;
8          pe ← PE(Unf) //Atualiza as transições habilitadas;
9          se e é um evento cut-off então
10             cut-off ← cut-off ∪ e;
11         fim
12     senão
13         pe ← pe \ {e};
14     fim
15 fim
```

O algoritmo ERV utiliza os conceitos anteriormente descritos, gerando o prefixo finito completo e truncando o processo de ramificação através do evento de corte. Como entrada, o algoritmo recebe uma rede de Petri segura, a partir da qual será construído o desdobramento. A rede de ocorrências **Unf** é inicializada com os lugares da marcação inicial M_0 da rede (linha 1). A lista de eventos **pe** é inicializada com as transições habilitadas a partir de M_0 (linha 2). O evento de corte **cut-off** armazenará todos os eventos de corte e será iniciado como vazio (linha 3). Enquanto existirem eventos, será executado o laço contendo o desdobramento propriamente dito (linha 4). Através da ordem adequada, será escolhida uma transição habilitada pela marcação inicial (linha 5). A linha 6 trata sobre o truncamento do processo de ramificação, uma vez identificado o evento de corte. Na linha 7, a transição selecionada é disparada, sendo gerado e inserido junto a rede desdobrada, seu respectivo homomorfismo e as condições referentes ao seu pós-conjunto. Esta linha representa a nova marcação após o disparo da transição escolhida. As transições habilitadas pela nova marcação são inseridas na lista de eventos **pe** (linha 8). Na linha 9 é verificado se o evento escolhido é um evento de corte, e que caso seja, será adicionado na

lista de eventos de corte (linha 10). Caso as transições não sejam disparadas nem façam parte da rede desdobrada, serão inseridas na lista **pe** (linha 13).

3.3 Conclusão

Este capítulo abordou o conceito de desdobramento criado por McMillan e aperfeiçoado posteriormente por Esparza, como sendo uma técnica de análise e de complexidade menor em relação ao grafo de alcançabilidade, sendo uma rede finita com todas as marcações alcançáveis da rede original.

No decorrer do capítulo foram apresentados conceitos fundamentais que envolvem o processo de desdobramento como: rede de ocorrência, homomorfismo, processo de ramificação, configuração, prefixo finito completo e evento de corte. Posteriormente, foi apresentado o algortimo de desdobramento de Esparza, Römer e Vogler, denomidado algoritmo ERV *Unfolding*, que gera um prefixo finito mínimo em relação ao algoritmo de McMillan.

Estes conceitos foram fundamentais para o entendimento do processo de desdobramento e do aprendizado das ferramentas utilizadas para o desdobramento de uma rede de Petri, conforme será visto no capítulo seguinte.

Capítulo 4

Ferramentas

Neste capítulo são apresentadas as principais ferramentas utilizadas para gerar o desdobramento de redes de Petri, tendo como base o algoritmo de ERV *Unfolding*. Além de conceituá-las e demonstrá-las, serão destacadas as suas desvantagens dado a complexidade do código envolvido bem como serão geradas simulações utilizando cenários de redes de Petri de pequeno e grande porte, visando analisar os resultados obtidos.

4.1 Ferramentas de Desdobramento

O algoritmo de ERV *Unfolding* foi utilizado como base para o desenvolvimento de diversas ferramentas que geram o desdobramento de redes de Petri, entre as quais podemos destacar o *MOLE* e o *PUnf*. O *software PUnf* foi desenvolvido por Victor Khomenko [Khomenko, 2016], com a finalidade de trabalhar com conceitos de *slice* e paralelismo no desdobramento de uma rede de Petri. Outra ferramenta utilizada para o desdobramento de uma rede é *CUnf*, que utiliza o conceito de arcos de leitura que permite a uma transição, ao invés de consumir ou produzir, a leitura das marcações de um lugar, visando a geração do prefixo finito completo da rede.

Todas as ferramentas citadas possuem uma mesma estrutura básica para geração do desdobramento: arquivo de entrada em formato texto descrevendo a estrutura da rede de Petri implementada; estrutura de processamento com parâmetros pré-definidos; e arquivo de saída em formato texto. Para representação gráfica do prefixo finito, será utilizado o *software graphiviz*, sendo uma ferramenta *open source* e compatível com as ferramentas de desdobramento já citadas.

4.1.1 Ferramenta *MOLE*

A ferramenta *MOLE* é um *software* sob licença pública geral (GNU GPL), utilizada para gerar o desdobramento de redes de Petri seguras, tendo sido desenvolvida por Stefan Römer e Stefan Schwoon [Schwoon and Romer, 2016]. O *MOLE* foi escrito na linguagem C, utilizando ponteiros e estruturas de listas, e é compatível com o ambiente do Projeto PEP (*Programming*

Environment based on Petri Nets), mantido pelo *Parallel Systems Group* do Departamento de Ciência da Computação da Universidade de Oldenburg na Alemanha [Best and Stehno, 2016].

A estrutura de entrada possui a descrição da rede no formato próprio do ambiente, gerando como saída, após processamento, um arquivo com o desdobramento da rede. O arquivo de saída, pode ser transformado em um formato gráfico, utilizando uma ferramenta gráfica compatível. A descrição da estrutura de entrada, será realizada utilizando a rede de Petri da figura 4.1

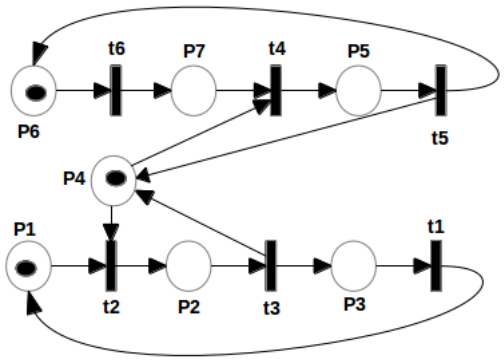

Figura 4.1: Rede de Petri segura.

O arquivo de entrada que descreve a rede de Petri possui uma estrutura obrigatória, que contém em ordem, os seguintes elementos [PEP, 2016]:

1. Cabeçalho;
2. Lista dos lugares;
3. Lista das transiçõess;
4. Lista dos arcos de transição para lugar;
5. Lista dos arcos de lugar para transição.

O cabeçalho de deve ser iniciado com a palavra-chave **PEP**. Na segunda linha deve ser especificado o tipo de rede que será desdobrada, sendo que para uma rede de Petri Lugar/-Transição será utilizado a abrevitura **PTNet**. A última linha do cabeçalho, conterá o formato da rede utilizada, sendo utilizada a palavra-chave **FORMAT_N**. Assim, o cabeçalho terá a seguinte estrutura:

PEP
PTNet
FORMAT_N

Logo abaixo do cabeçalho, estarão dispostas as listas contendo lugares, transições, arcos de transição para lugar e arcos de lugar para transição, indicados pelas respectivas palavras-chave **PL**, **TR**, **TP** e **PT**. Todas devem estar presentes no arquivo mesmo no caso de a lista a ser descrita estar vazia.

Como o formato do arquivo foi desenvolvido para um ambiente gráfico, é necessário especificar a posição dos lugares e das transições através das coordenadas **num@num**, sendo

num um número inteiro positivo ou negativo. É necessário também, especificar um nome e um identificador único para os lugares e transições. No tocante aos lugares pode ser definido a marcação **M num m num**, onde **M** representa a marcação incial, **m** a marcação corrente e **num** ao número de marcas .

Para identificar os arcos de transição para lugar **(TP)**, é necessário seguir a sequência **identificador_da_transição < identificador_do_lugar**, e para os arcos de lugar para transição **(PT)**, a sequência **identificador_do_lugar > identificador_da_transição**.

A figura 4.2 representa o arquivo de entrada referente à rede de Petri da figura 4.1, utilizado como entrada no *MOLE* para gerar o desdobramento da rede apresentado na figura 4.3.

```
PEP
PTNet
FORMAT\_N
PL
1"p1"0@0M1m1
2"p2"0@0M0m0
3"p3"0@0M0m0
4"p4"0@0M1m1
5"p5"0@0M0m0
6"p6"0@0M1m1
7"p7"0@0M0m0
TR
1"t1"0@0
2"t2"0@0
3"t3"0@0
4"t4"0@0
5"t5"0@0
6"t6"0@0
TP
1<1
2<2
3<3
3<4
4<5
5<4
5<6
6<7
PT
1>2
2>3
3>1
4>2
4>4
5>5
6>6
7>4
```

Figura 4.2: Arquivo de Entrada do Mole para a rede da figura 4.1.

Para geração do desdobramento utilizando a ferramenta *MOLE*, versão 1.1.0 e o *software graphviz*, foi utilizada a plataforma *Linux - Ubuntu, versão 14.04*, sendo adotados os seguintes passos:

1. O arquivo de entrada foi criado com a extensão *".ll_net"*;

2. O comando do *MOLE* utilizado no shell do Sistema Operacional para gerar o desdobramento foi: **./mole <nome do arquivo>.ll_net**;

3. Para gerar o gráfico da rede de desdobramento utilizando o *software graphviz* se faz necessário formatar os dados gerados no item 2, utilizando o seguinte comando do *MOLE*: **./mci2dot <nome do arquivo>.mci > <nome do arquivo>.dot**;

4. O comando do *graphiviz* utilizado no shell do Sistema Operacional para gerar o desdobramento no formato gráfico foi: **dot <nome do arquivo>.dot -o <nome do arquivo>.png -Tpng -Grankdir=LR**. A representação gráfica do desdobramento pode ser verificado na figura 4.3.

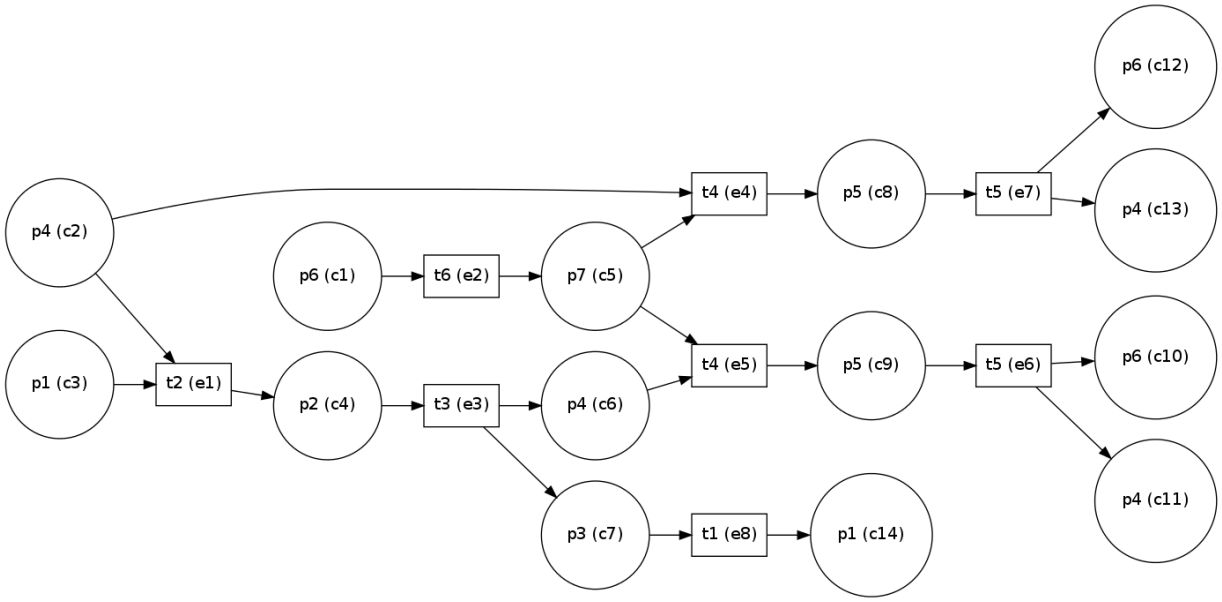

Figura 4.3: Representação gráfica do desdobramento gerado pelo *MOLE*.

Já as estruturas de dados são compostas de vetores e variáveis, as quais têm como função a criação da estrutura da rede de entrada e da rede de ocorrência, com a finalidade de auxiliar o processo de desdobramento. Dentre as várias estruturas e variáveis existentes, podemos citar as seguintes:

a) *nodelist_t*: estrutura responsável pelas conexões dos lugares com as transições e vice-versa, do homomorfismo e de outras conexões importantes para a geração do desdobramento.

b) *place_t*: estrutura que armazena as informações dos lugares da rede, como nome do lugar, se o lugar possui ou não uma marca, entre outros.

c) *trans_t*: armazena as informações das transições da rede, como nome da transição, quantidade de lugares do pré-conjunto e pós-conjunto da transição, etc.

d) *contingent_t*: armazena as estruturas encarregadas de fazer as conexões.

e) *net_t*: responsável por armazenar as informações gerais da rede Lugar/Transição do arquivo de entrada, como a quantidade de lugares e transições, armazena o maior valor da quantidade de elementos dos pré e pós-conjuntos das transições, entre outros.

f) *cond_t*: armazena as informações das condições, como identificador da marcação que auxiliará no disparo das condições, identificador da condição, homomorfismo da condição, etc.

g) *event_t*: estrutura que armazena as informações dos eventos, como identificador da marcação que serve para verificar se o evento está habilitado, identificador do evento, quantidade de condições do pré-conjunto e pós-conjunto do evento, homomorfismo do evento, entre outros.

h) *unf_t*: armazena as informações gerais da rede de ocorrência.

i) *hashcell_t*: estrutura responsável por armazenar informações de uma marcação.

j) *events*: é um vetor de ponteiros com 2000 posições em que cada posição aponta para uma estrutura *event_t*. Serve para auxiliar o disparo de um evento.

k) *hash*: é um vetor de ponteiros onde cada posição aponta para uma estrutura *hashcell_t*. Serve para auxiliar o evento de corte, armazenando as marcações da rede.

l) *parikh_t*: armazena as informações de uma transição pertencente a uma configuração.

m) *parikh*: é um vetor de estruturas *parikh_t*. Serve para armazenar a configuração mínima do evento, sendo esta sempre em ordem crescente.

n) *pe_queue_t*: estrutura responsável por armazenar as informações das transições habilitadas por uma determinada marcação.

o) *pe_queue*: é um vetor de ponteiros com 1024 posições, que apontam para as estruturas *pe_queue_t*. Serve para armazenar as transições habilitadas.

p) *cutoff*: variável que controla a existência de um evento de corte.

q) *cutoff_list*: ponteiro para os eventos que produzem o *cut-off*, formando uma lista encadeada com estes.

r) *colists*: vetor de ponteiros, que armazenam as configurações da rede.

s) *ev_mark*: variável que auxilia no controle das condições e eventos habilitados.

t) *pe_qsize*: variável auxiliar que armazena a quantidade de transições habilitadas presentes na lista *pe_queue*.

Cada estrutura de dados citada acima, possui um conjunto de campos e ponteiros, que executam determinadas tarefas dentro da estrutura. A título de conhecimento, será descrito os campos e ponteiros que compõe a estrutura de dados *place_t*, conforme segue abaixo:

a) *name*: campo que representa o nome do lugar, como por exemplo p_1.

b) *id*: identificar do lugar, exemplo *1*.

c) *marked*: campo que indica se o lugar possui ou não uma marca.

d) *conds*: é um ponteiro que auxilia no homomorfismo do lugar, apontando para a condição ou condições derivadas deste lugar.

e) *preset*: é um ponteiro para uma estrutura *nodelist_t*, que faz a conexão do lugar com o seu pré-conjunto (transição).

f) *postset*: ponteiro para uma estrturua *nodelist_t*, que faz a conexão do lugar com o seu pós-conjunto (transição).

g) *next*: é um ponteiro para outra estrutura *placa_t*, serve para associar os lugares da rede.

Na figura figura 4.4, temos a representação *place_t*:

Figura 4.4: Campos e ponteiros da estrutura *place_t*.

4.1.2 Ferramenta *PUnf*

A ferramenta *PUnf - Petri Net Unfolder*, constrói um prefixo finito completo de uma rede de Petri segura. É uma eficiente implementação de paralelismo, podendo ser utilizado isoladamente ou em conjunto no ambiente do Projeto PEP. Os prefixos gerados pelo *PUnf* podem ser utilizados como entrada para o *CLP - Checker base on Linear Programming*, para identificação de *deadlock-freeness* [Esparza and Heljanko, 2008].

O *PUnf* suporta as seguintes classes de redes:

- Redes de baixo nível: no formato *FORMAT_N (.ll_net)*, os quais são suportados pelo ambiente PEP.

- Redes de alto nível: no formato *FORMAT_2N (.hl_net)*, os quais também são suportados no ambiente PEP.

- *Signal Transition Graphs*: no formato *STG (.stg)*.

Atualmente, a ferramenta *PUnf* não possui um *solver* implementado, sendo que ao invés disso, processa uma sequência de tarefas e predicados que permitem a construção da rede desdobrada.

A estrutura de entrada da rede de Petri da figura 3.5, é a mesma utilizada na ferramenta *PUnf*. Para geração do desdobramento utilizando a ferramenta *PUnf*, versão 8.51, foram adotados os seguintes procedimentos:

1. O arquivo de entrada foi criado com a extensão *".ll_net"*.

2. O comando do *PUnf* utilizado no shell do Sistema Operacional para gerar o desdobramento foi: **./punf <nome do arquivo>.ll_net -m=<nome do arquivo>.mci**.

3. Para gerar o gráfico da rede de desdobramento utilizando o *software graphviz* se faz necessário formatar os dados gerados no item 2, utilizando o seguinte comando do *PUnf*: **./mp2dot <nome do arquivo>.mci**. O utilitário **mp2dot** gera automaticamente um arquivo com a extensão **.dot**.

4. O comando do *graphviz* utilizado no shell do Sistema Operacional para gerar o desdobramento no formato gráfico foi: **dot <nome do arquivo>.dot -o <nome do arquivo>.png -Tpng -Grankdir=LR**.

A figura 4.5 apresenta graficamente a rede desdobrada utilizando a ferramenta *PUnf*.

4.1.3 Ferramenta *CUnf*

Recentemente, a construção de desdobramento foi extendido para redes de Petri, através de arcos de leitura, conhecido como redes Contextuais ou *c-nets* [Rodríguez and Schwoon, 2013]. As redes Contextuais são redes de Petri extendidas, onde além das conexões entre os lugares e as transições, pode-se fazer a leitura dos arcos. Essa técnica permite verificar a existência de marcações em um lugar, antes da transição ser disparada. Assim, a transição pode ser considerada de leitura de um contexto requerido para o disparo, por isso chamada de redes Contextuais [Rodríguez and Schwoon, 2013]. As redes contextuais permitem uma melhor modelagem de sistemas que fazem vários acessos simultâneos de um recurso em comum, como acesso simultâneo a banco de dados e circuitos assíncronos. O desdobramento de uma rede Contextual é outra rede *c-nets* que representa totalmente o comportamento (marcações alcançáveis) da rede. A ferramenta *CUnf* constrói um prefixo finito completo apenas de redes seguras.

A ferramenta *CUnf* foi desenvolvida na linguagem C, utilizando ponteiros e estruturas de listas, sendo compatível com o ambiente PEP. O *CUnf* suporta as seguintes classes de redes:

- Redes de baixo nível: no formato *FORMAT_N (.ll_net)*, os quais são suportados pelo ambiente PEP.

- Redes de alto nível: no formato *FORMAT_2N (.hl_net)*, os quais também são suportados no ambiente PEP.

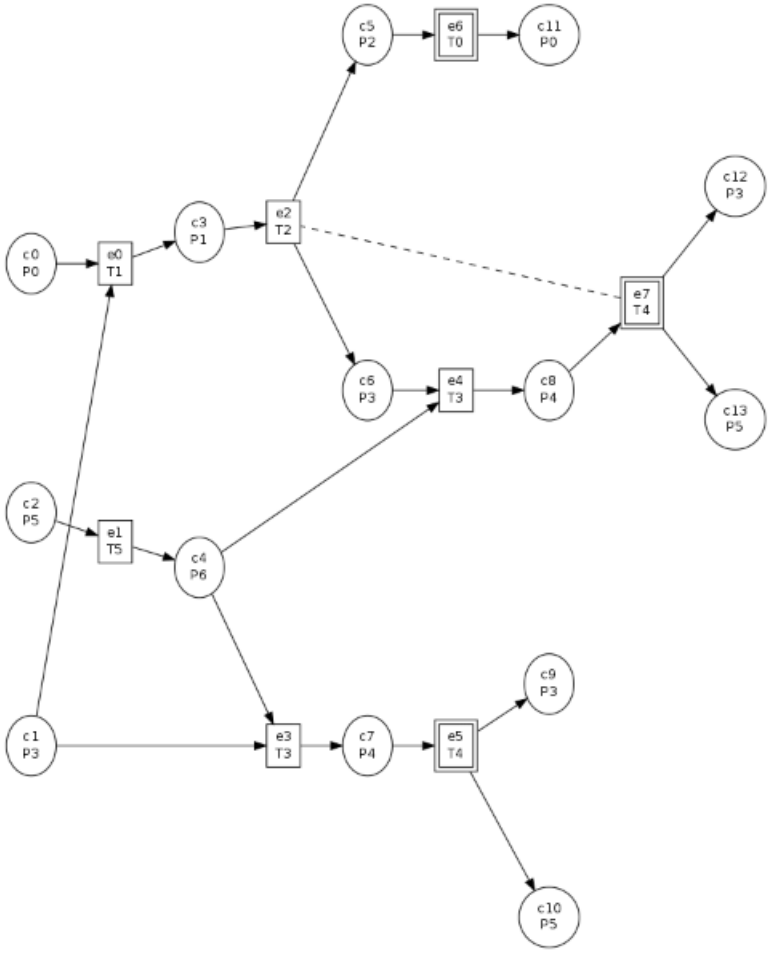

Figura 4.5: Representação gráfica do desdobramento gerado pelo *PUnf*.

A estrutura de entrada da rede de Petri da figura 3.5, é também a mesma utilizada na ferramenta *CUnf*. Para geração do desdobramento utilizando a ferramenta *CUnf*, versão 1.6, foram adotados os seguintes passos:

1. O arquivo de entrada foi criado com a extensão *".ll_net"*.

2. O comando do *CUnf* utilizado no shell do Sistema Operacional para gerar o desdobramento foi: **./cunf -o u.cuf <nome do arquivo>.ll_net**.

3. Para gerar o desdobramento no ambiente gráfico, se faz necessário formatar os dados gerados no item 2, utilizando o seguinte comando do *CUnf*: **./cuf2pep.py <u.cuf> <nome do arquivo>.ll_net**.

4. O comando utilizado no shell do Sistema Operacional para gerar o desdobramento no formato gráfico foi: **./pep2dot <nome do arquivo>.ll_net | dot -T pdf > <nome do arquivo>.pdf**.

Na figura 4.6 é apresentada graficamente a rede desdobrada, resultado da utilização da ferramenta *CUnf*.

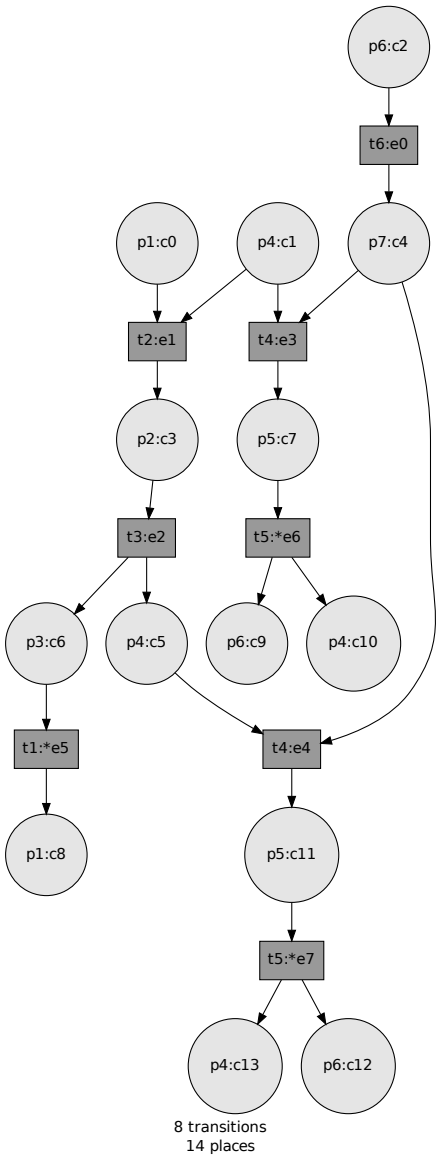

Figura 4.6: Representação gráfica do desdobramento gerado pelo *CUnf*.

4.1.4 Análise entre as ferramentas

As três ferramentas apresentadas utilizam como base o algoritmo *ERV Unfolding* para gerar o prefixo finito completo, sendo que:

a) O *PUnf* utiliza-se do conceito de paralelismo e *slice* para gerar o desdobramento.

b) O *CUnf* vale-se da abordagem de redes Contextuais para gerar o prefixo finito completo.

No tocante a linguagem de programação, todas foram desenvolvidas utilizando a linguagem *C*, com ponteiros e listas como de dados. Isto pode ser considerado uma desvantagem em virtude da complexidade dos algoritmos e a consequente dificuldade de manutenção, prejudicando a alteração dos códigos visando a simulação de redes em ambientes distintos. A

ferramenta *CUnf* utiliza-se de códigos implementados em *Python* para auxiliar no processo de desdobramento da rede.

No que diz respeito ao cenário de testes, quando da simulação de uma rede de pequeno porte, como o da figura 4.1, com 7 lugares e 6 transições, ambas as ferramentas apresentaram resultados idênticos quanto ao número de condições, de eventos e de *cut-off* para a rede desdobrada, conforme já foram apresentados através da figuras 4.3, 4.5 e 4.6, e conforme a tabela 4.1 abaixo.

Tabela 4.1: Resultados obtidos para uma rede de pequeno porte.

	CUnf	MOLE	PUnf
Condições	14	14	14
Eventos	8	8	8
Cut-off	3	3	3

Já quando o cenário simulado ocorreu em uma rede de grande porte, utilizando-se como exemplo um problema de planificação descrito pela linguagem PDDL - *Planning Domain Definition Language* e traduzido para uma rede de Petri com 48 lugares e 273 transições, os resultados obtidos foram idênticos entre as ferramentas *MOLE* e *CUnf*, porém diferentes com a ferramenta *PUnf*.

Na figura 4.7, pode-se verificar que a ferramenta *CUnf* gerou uma rede desdobrada com 2941 condições, 725 eventos e 577 eventos de corte, resultado idêntico para ferramenta *MOLE*:

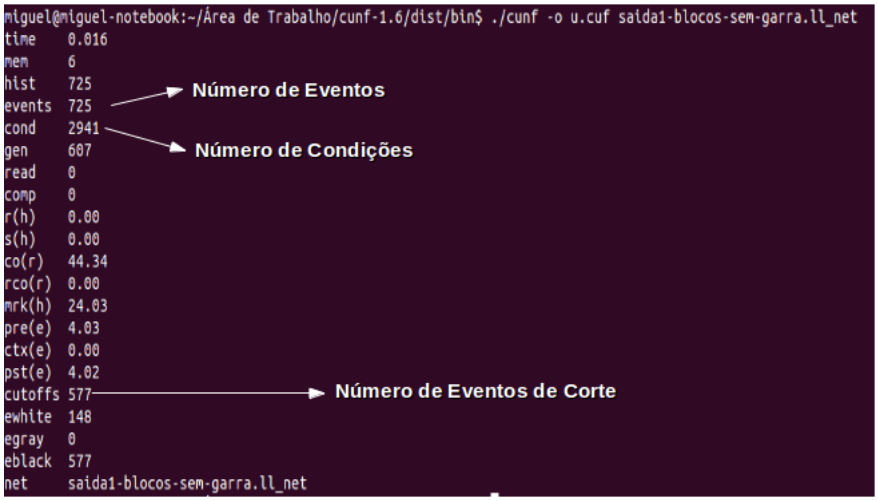

Figura 4.7: Resultados do desdobramento gerado pelo *CUnf*.

Na figura 4.8, verifica-se a divergência de resultados gerados pela ferramenta *PUnf*, para o mesmo exemplo, gerando uma rede desdobrada com 2916 condições, 717 eventos e 569 eventos de corte:

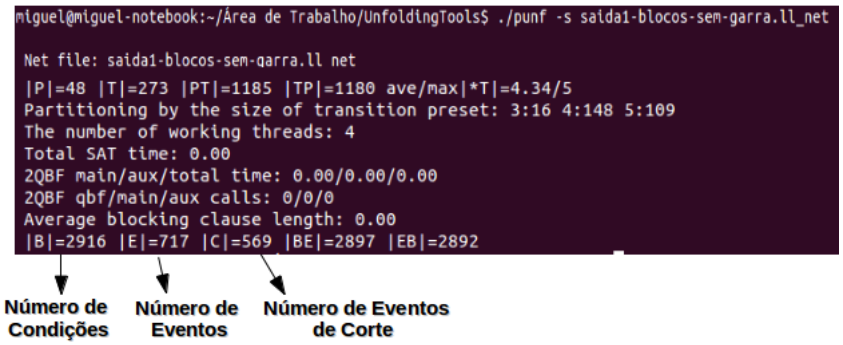

Figura 4.8: Resultados do desdobramento gerado pelo *PUnf*.

Assim, para uma rede de grande porte, foram encontrados resultados diferentes quando comparadas as ferramentas *MOLE* e *CUnf* com a ferramenta *PUnf*, conforme resumido na tabela 4.2

Tabela 4.2: Resultados obtidos para uma rede de grande porte.

	CUnf	MOLE	PUnf
Condições	2941	2941	2916
Eventos	725	725	717
Cut-off	577	577	569

As diferenças observadas quando da simulação de rede de grande porte, conforme tabela 4.2, pode estar sendo ocasionada pelo conceito de guarda de trânsição *(transition guard)*, que é uma sequência de tarefas condicionadas a certas restrições, durante o processo de desdobramento. Estas restrições são expressões e operadores lógicos. Durante o processo de desdobramento, o guarda de transição analisa as restrições e rotula as transições e lugares da rede desdobrada, de acordo com certos padrões estabelecidos, os quais são apenas citados no manual da ferramenta *PUnf* [Khomenko, 2016], não sendo possível analisar os detalhes desses padrões, além do que, não é possível realizar engenharia reversa sobre a ferramenta, uma vez que ela esta disponível apenas como executável.

4.2 Conclusão

O capítulo apresentou as principais ferramentas utilizadas no processo de desdobramento, sendo: *CUnf, MOLE* e *PUnf*. Ambas foram implementadas com a linguagem *C*, utilizando ponteiros e estruturas de dados, verificando-se ser uma desvantagem em virtude da complexidade dos algoritmos e a consequente dificuldade de manutenção.

Apesar de utilizarem como base o algoritmo ERV *Unfolding*, ambas apresentam implementações diferenciadas em seus códigos, o que também tornou-se uma desvantagem, pois uma vez aplicadas em um cenário de grande porte, houve divergência nos resultados obtidos quando

comparada o *CUnf/MOLE* com a ferramenta *PUnf*. Essa divergência de resultados pode estar sendo gerada pela forma que o processo de desdobramento é realizado pela ferramenta *PUnf*, através do uso do conceito de guarda de transição, realizando operações de através de certos padrões estabelecidos, os quais não são disponibilizados para a referida análise.

Capítulo 5

Implementação Matricial

Cálculos matriciais fazem parte de muitos algoritmos computacionais e científicos, incluindo sistemas de comunicação, processamento digital de sinais, computação gráfica e operações financeiras. Dada a importância, as matrizes são consideradas também uma forma conveniente de armazenar e manipular dados [Han and Kamber, 2011]. As operações matriciais são tarefas computacionais complexas, exigindo tempo e esforço na implementação [Irturk, 2009]. Porém, nas últimas décadas, foram aprofundadas pesquisas no sentido de desenvolver ferramentas e algoritmos que permitam cálculos mais rápidos e eficientes utilizando matrizes [Irturk, 2009]. A abordagem matricial de uma rede de Petri permite com maior facilidade, o estudo das propriedades estruturais e comportamentais da rede, sendo inclusive um facilitador para a implementação de um algoritmo de desdobramento, uma vez que a técnica será implementada através de operações matriciais. A abordagem matricial permite também a simulação e validação de projetos, visando minimizar falhas ou interrupções (*deadlocks*) do sistema [Santos et al., 2014].

Neste capítulo serão apresentadas as principais funções que compõem o algoritmo de desdobramento implementados no *MATLAB*, com base no algorimo ERV, utilizando vetores e operações matriciais.

5.1 *Matrix Laboratory - MATLAB*

MATLAB, abreviação de *Matrix Laboratory* é uma linguagem de programação de alto nível, desenvolvida pela *MathWorks*, sendo utilizada por milhões de usuários no mundo inteiro [MathWorks, 2016]. Utiliza matrizes para cálculos e aplicações científicas, sendo apreciada pelos desenvolvedores pelo fato da diversidade de bibliotecas e funções disponíveis, a não necessidade de declaração do tipo de variáveis, e a interatividade da ferramenta [Dubrau, 2012]. Algoritmos desenvolvidos em *MATLAB* tem um excelente desempenho para cálculos matriciais complexos, podendo ser considerado por vários pesquisadores como um padrão de ferramenta para o desenvolvimento de algorimos para operações matriciais [Irturk, 2009]. Tradicionalmente, o *MATLAB* é utilizado na realização de simulações em várias áreas como engenharia de controle,

processamento de imagens e comunicações, e recentes pesquisas tem apontado a sua eficiência no campo da programação paralela [Prasad, 2012].

A interação com a ferramenta é realizada mediante instruções (comandos), funções e *scripts*, utilizando como matéria prima "matrizes" [Scappini, 2010].

5.2 Análise do Algoritmo ERV com a ferramenta *MATLAB*

O algoritmo ERV *Unfolding*, conforme visto no capítulo 3, seção 3.2, gera o prefixo finito completo, truncando o processo de ramificação através de eventos de corte. O algoritmo ERV utiliza como entrada uma rede de Petri segura e sua marcação inicial (M_0), gerando como saída o desdobramento da rede representado por *Unf*, conforme figura 5.1.

Algoritmo 2 Construção do Prefixo Finito Completo

Require: Uma rede N, onde M_0=$\{p_1, ..., p_k\}$

Ensure: Um Prefixo Finito Completo do *Unfolding* Unf

Figura 5.1: Dados de entrada e saída do algoritmo ERV.

Para gerar o desdobramento de uma rede de Petri segura, o *MATLAB* utiliza matrizes, sendo necessário três matrizes de entrada para iniciar o processo de desdobramento. As matrizes de entrada são identificadas de *Pre, Pos e* M_0, representando a matriz lugar-transição, a matriz transição-lugar e a matriz de marcação inicial.

O processo de desdobramento, gerará como saída *pre_unfolding* que representa a matriz lugar-transição do desdobramento; *pos_unfolding* que traz a matriz transição-lugar do desdobramento; o vetor *marcacoes_alcancadas* do desdobramento, o vetor *transicoes_disparadas* durante o processo de desdobramento e o vetor *cut_off* que contém os eventos de corte do desdobramento.

No tocante a inicialização das variáveis, *Unf* e *pe*, o algoritmo ERV se utiliza dos lugares de M_0 da rede de Petri e das transições habilitadas a partir de M_0, respectivamente. Já a lista de eventos de corte, representado por *cut-off*, será inicializado sem qualquer evento no seu conjunto, conforme figura 5.2.

1: Unf $\leftarrow$ lugares de M_0
2: pe $\leftarrow$ transições habilitadas por M_0
3: cut - off $\leftarrow$ 0

Figura 5.2: Inicialização de variáveis do algoritmo ERV.

As variáveis *Unf, pe* e *cut-off* do algoritmo ERV, são representadas no *MATLAB* pelos vetores *marcacoes_alcancadas*, *transicoes_disparadas* e *cut_off*. O vetor *marcacoes_alcancadas* será inicializado com o conteúdo de M_0, onde será identificado os lugares no vetor que possuem

valores diferentes de zero e armazenados no vetor *P0* para uso no processo de desdobramento. O vetor *transicoes_disparadas* será inicializado sem qualque evento, uma vez que será necessário identificar, quais são os lugares com marcações que habilitam uma dada transição. Já o vetor *cut-off* será inicializado vazio, uma vez que durante o desdobramento, os eventos de corte serão armazenados no vetor.

O processo de desdobramento propriamente dito no algoritmo ERV encontra-se contido no laço *while - end while* (linha 4 a linha 15). O laço será executado enquanto houverem eventos na lista de eventos *pe*, conforme Figura 5.3.

```
4:  while pe ≠ 0 do
5:     Escolha um evento e=(t,X) de pe tal que [e] seja mínima respeitando a ordem ≺
6:     if [e] ∩ cut-off = 0 then
7:        Adicione e e seus pós-conjunto em Unf
8:        pe ← PE(Unf)   // Atualiza as transições habilitadas
9:        if e é um evento cut-off then
10:          cut-off ← cut-off ∪ e
11:       end if
12:    else
13:       pe ← pe \ {e}
14:    end if
15: end while
```

Figura 5.3: Laço de repetição do processo de desdobramento do algoritmo ERV.

No *MATLAB*, o laço de repetição do processo de desdobramento é representado de maneira similar, pois enquanto houverem eventos na lista de eventos, representado pela variável *transicoes_disparadas*, o processo de desdobramento será executado.

Na linha 5 do algoritmo ERV é escolhida uma transição habilitada (evento) pela marcação inicial, respeitando a ordem adequada. Na proposta apresentada no *MATLAB*, primeiramente será verificada as transições habilitadas para o disparo, através da comparação de cada coluna da matriz *Pre* com a matriz de marcação inicial (M_0). Definida a transição (evento) de menor configuração local, será a mesma armazenada em um vetor para construção das matrizes *pre_unfolding* e *pos_unfolding*.

Na linha 6 do algoritmo ERV existe uma condição em que será verificado se a transição escolhida encontra-se na lista de eventos *cut-off*, e em sendo verdadeiro, a transição e seu pós-conjunto de lugares será adicionada na rede de ocorrência *Unf* (linha 7). Caso a condição seja falso, ou seja, caso as transições não sejam disparadas nem façam parte da rede desdobrada, serão adicionadas na lista *pe* (linha 13). Estes mesmos procedimentos são adotados no *MATLAB*, pois estando a transição escolhida na lista de eventos *cut_off*, será adicionada juntamente com o pós-conjunto de lugares na rede de ocorrência *(marcacoes_alcancadas)*, gerando um lista expandida de lugares e transições *(transicoes_disparadas)* os quais definirão as matrizes *pre_unfolding* e *pos_unfolding*.

As transições habilitadas pela nova marcação são inseridas na lista de eventos *pe*, gerando um processo de atualização das transições habilitadas (linha 8). No *MATLAB*, essas transições são inseridas no vetor de transições habilitadas *(transicoes_habilitadas)*.

Na linha 9 do algoritmo ERV, será executada outra condição em que será verificado se a transição escolhida é um evento de corte. Se o evento pertencer a uma ordem adequada para a rede ou caso o evento leve a uma marcação já representada anteriormente, será considerado um evento de corte. Em sendo um evento de corte, será inserido na lista de eventos de corte *(cut-off)*. No *MATLAB*, para ser verificado se uma transição é um evento de corte, é verificado a ordem adequada no vetor de eventos de corte *(cut-off)* bem como é realizado a verificação dessas transições se elas levam a marcações anteriormente alcançadas, comparando o vetor *marcacoes_alcancada*.

5.3 Algoritmo de Desdobramento

O algoritmo de desdobramento foi desenvolvido utilizando uma função, onde se faz necessário inserir os valores de entrada, sendo a matriz Pre, a matriz Pos e o vetor de marcação inicial. O algoritmo de desdobramento será operacionalizado através de operações matriciais, funções lógicas e funções de controle de fluxo, gerando o desdobramento da rede tendo como resultado as matrizes Pre e Pos de desdobramento, as matrizes de expansão contendo os lugares e transições do desdobramento e os eventos de corte do desdobramento.

Neste trabalho serão utilizadas matrizes celulares, para armazenamento temporário de dados da rede de desdobramento. Uma matriz celular é uma matriz especial cujos elementos são células, contendo outras matrizes MATLAB [Chapman, 2010]. Por exemplo, uma célula de uma matriz celular pode conter uma matriz de números reais, outra, uma matriz de caracteres. Cada elemento de uma matriz celular é um apontador para outra estrutura de dados, e essas estruturas de dados podem ser diferentes tipos [Chapman, 2010]. As matrizes celulares usam chaves para selecionar e apresentar o conteúdo das células. As matrizes celulares são flexíveis, uma vez que qualquer quantidade de qualquer tipo de dados pode ser armazenada em cada célula. Assim, a diferença entre uma matriz celular e uma matriz é que, em uma matriz, todas as entradas devem ter o mesmo tipo de dados; já em uma matriz celular, pode-se mesclar tipos diferentes de dados. Neste caso, utilizou-se a matriz celular para armazenar os níveis de desdobramento, e os lugares e transições que fazem parte desse nível.

A execução da função se dá através da seguinte sintaxe: *[pre_unfolding,pos_unfolding,marcacoes_alcancadas,transicoes_disparadas,cut_off]=Desdobramento(Pre,Pos,M0)*

onde os parâmetros de entrada são representados por:

- *Pre*: matriz contendo os arcos orientados que ligam os lugares às transições;
- *Pos*: matriz contendo os arcos orientados que conectam as transições aos lugares;
- M_0: contém a marcação inicial da rede.

Referente a rede de Petri da Figura 4.1, as matrizes Pre, Pos e M_0, correspondem aos seguintes valores:

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \quad Pos = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad M0 = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \end{array} \begin{array}{c} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array}$$

e os parâmetros de saída são:

- *pre_unfolding*: contém a matriz Pre da rede de desdobramento;

- *pos_unfolding*: contém a matriz Pos da rede de desdobramento;

As matrizes *pre_unfolding* e *pos_unfolding* serão utilizadas para a construção da rede de ocorrência do desdobramento, sendo compostas por 0 e 1, sendo que para valores iguais a 1, representará a ligação do lugar com a transição ou da transição com o lugar, e para valores iguais a 0, a ausência de conexão entre eles.

Ao final do desdobramento da rede de Petri da Figura 4.1, as matrizes *pre_unfolding* e *pos_unfolding* serão representadas pelas seguintes estruturas matriciais:

$$pre\_unfolding = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{bmatrix} \quad pos\_unfolding = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & b_{2,3} & \cdots & b_{2,n} \\ b_{3,1} & b_{3,2} & b_{3,3} & \cdots & b_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & b_{m,3} & \cdots & b_{m,n} \end{bmatrix}$$

- *marcacoes_alcancadas*: é um vetor P x $\mathbb{N}$, na qual cada coluna representa uma marcação alcançada, conforme segue abaixo:

$$\begin{array}{c} \\ 1 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & \cdots & n \end{array} \\ \begin{bmatrix} P_{\mathbb{N}} & P_{\mathbb{N}} & P_{\mathbb{N}} & P_{\mathbb{N}} & \cdots & P_{\mathbb{N}} \end{bmatrix} \end{array}$$

Por exemplo, o vetor *marcacoes_alcancadas*={p_1,p_4,p_6}, mostra as marcações alcançáveis que estão armazenadas no vetor, com a seguinte estrutura de saída no *MATLAB*:

marcacoes_alcancadas = [1,4,6]

- *transicoes_disparadas*: é um vetor T x $\mathbb{N}$, na qual cada coluna representa uma transição disparada, conforme abaixo:

$$\begin{array}{c} \quad 1 \quad 2 \quad 3 \quad 4 \quad \cdots \quad n \\ 1 \quad \left[\begin{array}{cccccc} T_{\mathbb{N}} & T_{\mathbb{N}} & T_{\mathbb{N}} & T_{\mathbb{N}} & \cdots & T_{\mathbb{N}} \end{array} \right] \end{array}$$

Por exemplo, o vetor *transicoes_disparadas*={t_1,t_2}, mostra as transições disparas que estão armazenadas no vetor, tendo como resultado no *MATLAB* o que segue:

transicoes_disparadas = [1,2]

- *cut_off*: é um vetor de $\mathbb{N}$ que contém os eventos de corte da rede desdobrada.

$$cut\_off = \left[\begin{array}{cccccc} cut_{1,1} & cut_{1,2} & cut_{1,3} & \cdots & cut_{1,n} \end{array} \right]$$

Na rede de Petri da Figura 4.1, os eventos de corte gerados pelo *MATLAB* correspondem aos seguintes valores:

cut_off = [1,5,5]

5.4 Descrição do Algoritmo - Estudo de Caso

Para análise e validação do algoritmo, será realizado um estudo de caso utilizando a figura 3.5, anteriormente desdobrada pela ferramenta *MOLE*.

5.4.1 Definições Iniciais

O algoritmo é iniciado com a definição das variáveis que serão utilizadas durante a execução do código. As variáveis terão funções distintas, como variáveis auxiliares, células e matrizes de armazenamento. Dado a extensão do algoritmo, serão apresentados as principais variáveis utilizadas:

```
1 M0=M0';
2 [m,n]=size(Pre);
3 P0=find(M0);
4 marcacoes_alcancaveis(1)={P0};
5 transicoes_disparadas(1)={};
6 pre_unfolding=[];
7 pos_unfolding=[];
8 cut_off=[];
```

Na linha 1, é definido que a variável M_0 receberá o conteúdo da variável M_0, neste caso, a transposta da marcação inicial. Da matriz Pre será necessário extrair a quantidade de linhas e colunas, visando realizar as devidas comparações, inicialmente com a matriz de M_0 e posteriormente, com a matriz de marcação alcançada, com o objetivo de identificar as transições habilitadas para o disparo (linhas 2). Na linha 3, serão identificados as posições dos lugares

com marcações de M_0, sendo armazendos em um vetor chamado *P0*. O vetor *P0* será utilizado posteriormente e em conjunto com a matriz *Pre*, visando identificar quais serão as transições habilitadas pelos lugares de M_0. O vetor *marcacoes_alcancadas* foi definido como sendo uma célula, onde serão armazenados as marcações alcançáveis decorrentes do desdobramento (linha 4). O vetor *transicoes_disparadas* também definido como uma célula, armazenará os eventos disparados durante o processo de desdobramento da rede (linha 5). Nas linhas 6 e 7, foram definidas as variáveis *pre_unfolding* e *pos_unfolding* as quais armazenarão as matrizes da rede de desdobramento, não sendo definido o tamanho em virtude do crescimento dessas matrizes. O vetor *cut_off* armazenará os eventos de corte do processo de desdobrmento.

Na M_0 da rede de Petri da figura 3.5, foram identificados com *tokens* os lugares p_1, p_4 e p_6, através do comando *find* do *MATLAB*, o qual retorna as posições de valores diferentes de 0, sendo esses valores armazenados no vetor *P0*:

```
1 P0=find(M0);
```

A saída gerada pelo comando *find* foi: *P0=[1,4,6]*, sendo que cada valor representa a posição do lugar com *token* de M_0. Na figura 5.4 é apresentado o nível inicial do desdobramento da rede.

Figura 5.4: Nível inicial da rede de desdobramento.

Os lugares de *P0* precisam ser ordenados de forma a ser identificado a quantidade de lugares marcados, uma vez que será necessário criar um laço para fazer uma varredura na matriz *Pre* e compará-la com *P0*, visando identificar os lugares aptos a habilitar a transição correspondente. Os lugares ordenados serão armazenados em *P0ordenado*:

```
1 for k=1:length (P0)
2    P0ordenado=[P0ordenado k];
3 end
```

Os lugares ordenados de *P0*, ficarão assim armazenados: *P0ordenado=[1,2,3]*. A figura 5.5, mostra o ordenamento dos lugares.

Figura 5.5: Ordenamento dos lugares de P0.

5.4.2 Processo de desdobramento

No MATLAB, o laço de repetição do processo de desdobramento é representado de maneira similar ao algortimo ERV *Unfolding*, pois enquanto houverem eventos na lista de eventos, representado pela variável *transicoes_disparadas*, o processo de desdobramento será executado.

Inicialmente será realizada a identificação de cada transição contida na coluna da matriz *Pre*, sendo que a quantidade de colunas estará armazenada na variável *n*. No interior do laço, serão verificadas as transiçoes de *Pre* e comparadas com M_0 para identificar, conforme já citado, as transições habilitadas para o disparo, com o objetivo de criar as matrizes *pre_unfolding* e *pos_unfolding*. Para analise de cada transição, será utilizada a seguinte expressão do *MATLAB*:

```
Precoluna=Pre(:,i)
```

Em relação a Figura 4.1, a saída gerada da expressão acima, sendo *i=1* será:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A posição dos lugares pertencentes a cada coluna da transição da matriz *Pre*, serão armazenados na variável *ind*, da seguinte forma:

```
ind=find(Precoluna)
```

A saída gerada será *ind=[3]*, indicando que apenas na terceira posição *(p_3)*, se conecta a transição t_1, conforme a Figura 5.6.

Dado a Figura 5.6, verifica-se que a transição t_1 não estará habilitada ao disparo, uma vez que o lugar p_3 não possui marcação quando comparado a matriz de marcação inicial M_0. Quando i=2, o vetor gerado em relação a Figura 4.1 é:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

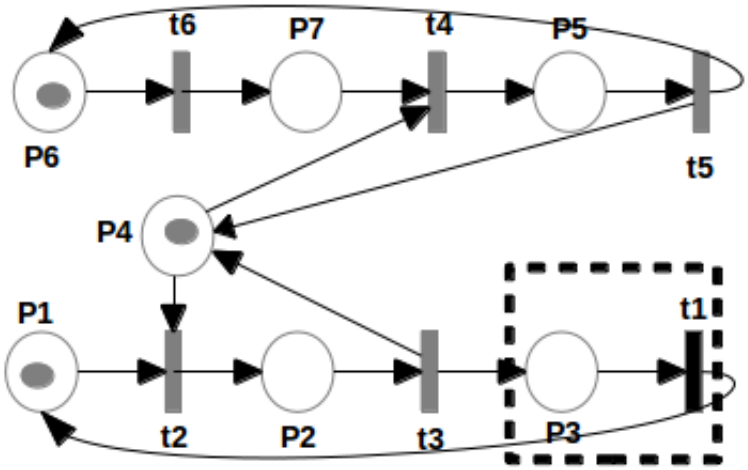

Figura 5.6: Lugar p_3 conectada a transição t_1.

Assim, pelos valores do vetor acima, *ind=[1,4]*, indica que a primeira e quarta posição do vetor *(p_1 e p_4)*, se conectam a transição t_2, conforme Figura 5.7.

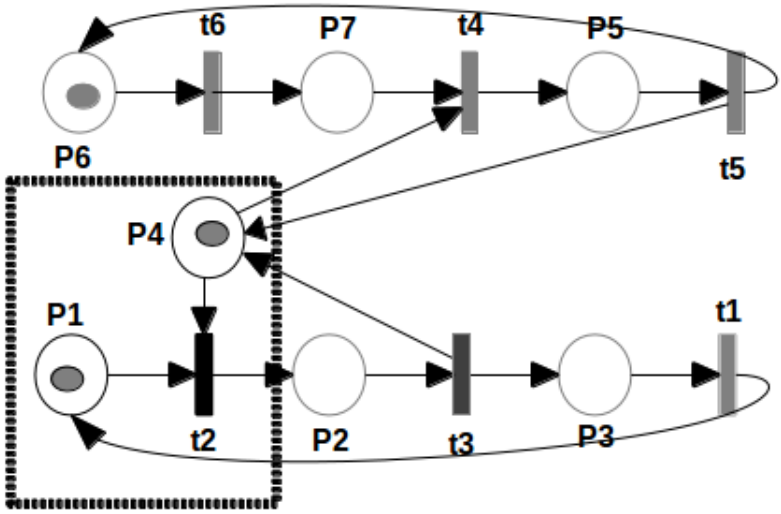

Figura 5.7: Lugares p_1 e p_4 se conectadam a transição t_2.

Verificando a Figura 5.7, identifica-se que a transição t_2 está habilitada para o disparo, uma vez que os lugares p_1 e p_4 possuem marcação quando comparado a matriz de marcação inicial M_0. A t_6 também esta habilitada para o disparo, dado que o lugar p_6 possui marcação. Todo esse processo de verificação será sempre realizado enquanto ocorrer o desdobramento da rede. O conjunto de lugares qua habilitam uma transição bem como o seu pós conjunto de lugares, serão armazenados em vetores auxiliares, visando a construção das matrizes *pre_unfolding* e *pos_unfolding*. A escolha da transição habilitada (evento) entre t_2 e t_6, se dará respeitando a ordem adequada, sendo que neste caso, será escolhido o evento t_2, conforme Figura 5.8.

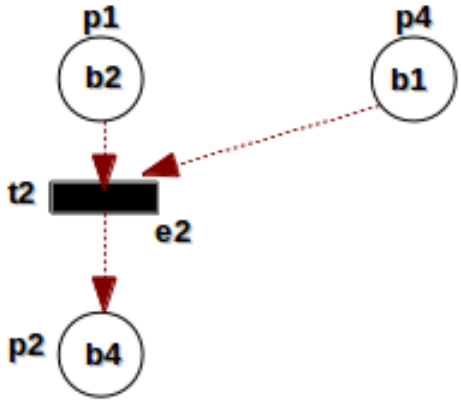

Figura 5.8: Inicio do processo de desdobramento em relação a figura 4.1.

Dando continuidade ao processo de desdobramento, a próxima transição a ser disparada, será t_6 gerando o pós conjunto de lugares da referida rede, conforme Figura 5.9.

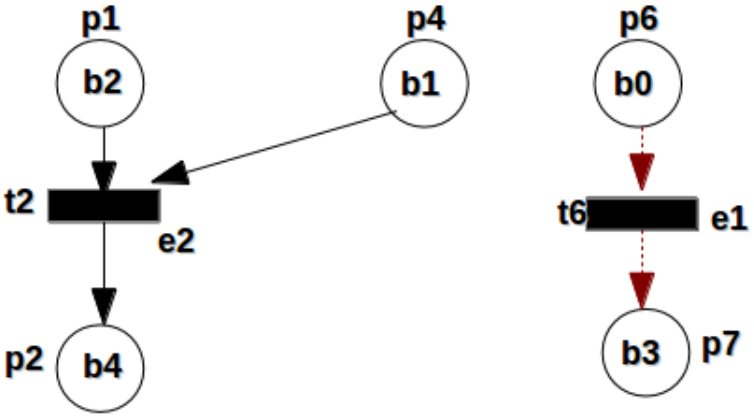

Figura 5.9: Processo de desdobramento em relação a figura 4.1.

Até o presente momento, o *MATLAB* gerou as seguintes informações, as quais são:

marcacoes_alcancadas = [1,4,6,2,7]
transicoes_disparadas = [2,6]

Como as transições (eventos) não são eventos de corte, as mesmas não serão adicionadas ao vetor *cut-off*. Dando continuidade ao processo de desdobramento, os lugares p_2 e p_7, encontram-se interligados às transições t_3 e t_4, respectivamente. Novamente pela ordem adequada, a transição t_3 estará habilitada, gerando o conjunto de pós conjunto de lugares p_3 e $p_{4.1}$, e posteriormente a transição t_4, com o seguinte lugar p_5, conforme Figura 5.10.

As transições habilitadas pela nova marcação são inseridas no *transicoes_disparadas*, gerando um processo de atualização das transições habilitadas. Assim, analisando os eventos

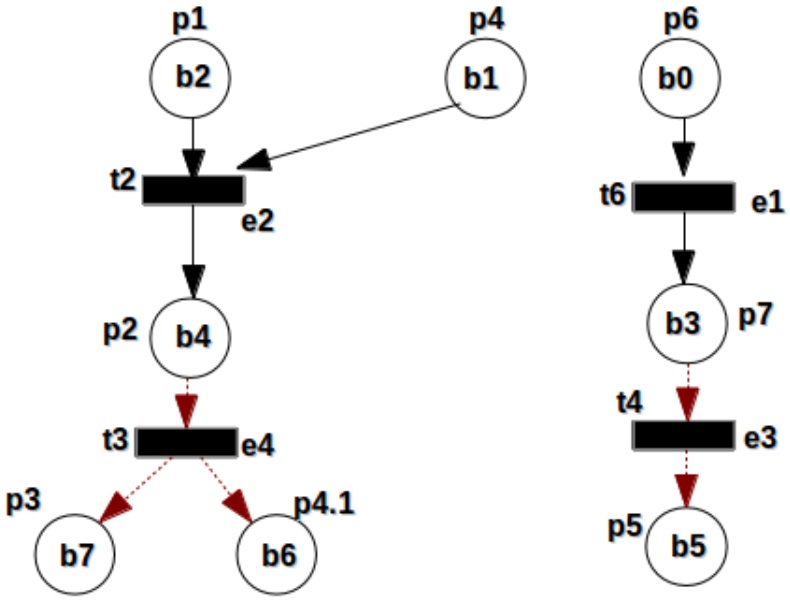

Figura 5.10: Desdobramento da rede em relação a figura 4.1.

presentes, os quais ainda não se caracterizam como eventos de corte, teremos os seguintes vetores gerados pelo *MATLAB*:

marcacoes_alcancadas = [1,4,6,2,7,3,4,5]
transicoes_disparadas = [2,6,3,4]

Durante o processo acima, ao ser encontrada um evento de corte, em que são verificados se o evento pertence a uma ordem adequada da rede ou caso o evento leve a uma marcação já alcançada anteriormente, o mesmo será adicionado na lista de eventos de corte *cut-off*. No *MATLAB*, para ser verificado se uma transição é um evento de corte, é verificado a ordem adequada no vetor da lista de transições disparadas *(transicoes_disparadas)* bem como é realizado a verificação dessas transições se elas levam a marcações anteriormente alcançadas, comparando o vetor *marcacoes_alcancada*. Assim, no *MATLAB* teremos os seguintes eventos *cut-off* gerados pelo código:

cut-off = [1,5,5]

Na rede da Figura 4.1, os eventos de corte *cut-off* gerados pelo *MATLAB* estão representados da seguinte forma na Figura 5.11:

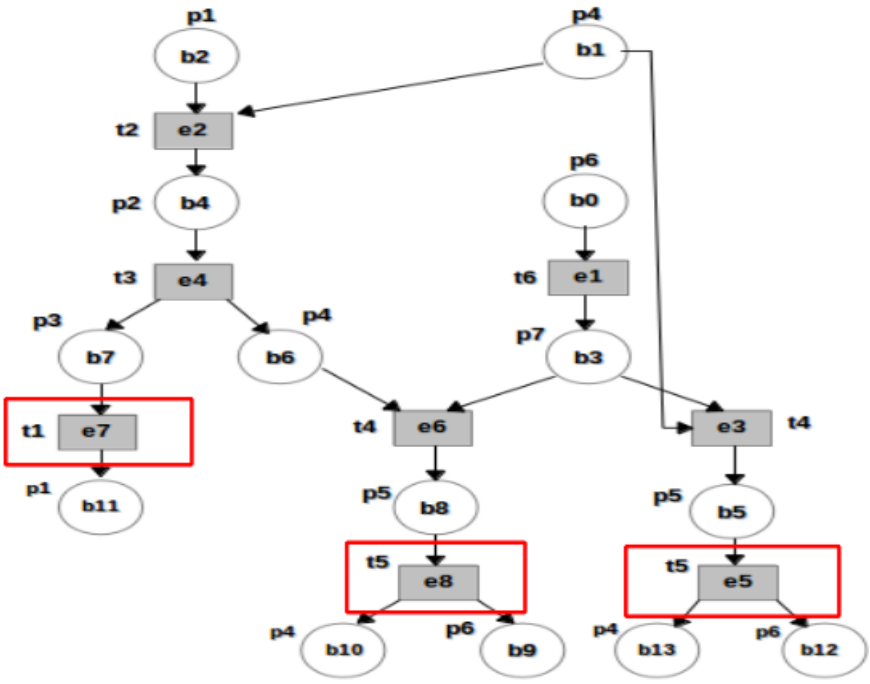

Figura 5.11: Eventos de corte gerados pelo *MATLAB* em relação a Figura 4.1.

As matrizes *pre_unfolding* e *pos_unfolding* serão geradas a partir do conjunto de lugares na rede de ocorrências do vetor *marcacoes_alcancadas* com o vetor *(transicoes_disparadas)*, sendo composta por valores de 0 e 1, sendo que para valores iguais a 1, representará a ligação do lugar com a transição ou da transição com o lugar, e para valores iguais a 0, a ausência de conexão entre eles. Da Figura 4.1, teremos as seguintes matrizes *pre_unfolding* e *pos_unfolding* gerados do processo de desdobramento no *MATLAB*:

$$pre\_unfolding = \begin{array}{c} \\ p_1 \\ p_4 \\ p_6 \\ p_2 \\ p_7 \\ p_3 \\ p_{4.1} \\ p_5 \\ p_{1.1} \\ p_{5.1} \\ p_{4.2} \\ p_{6.1} \\ p_{4.3} \end{array} \begin{array}{c} \begin{array}{ccccccc} t_2 & t_6 & t_3 & t_4 & t_1 & t_{4.1} & t_5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$pos\_unfolding = \begin{array}{c} \\ p_1 \\ p_4 \\ p_6 \\ p_2 \\ p_7 \\ p_3 \\ p_{4.1} \\ p_5 \\ p_{1.1} \\ p_{5.1} \\ p_{4.2} \\ p_{6.1} \\ p_{4.3} \end{array} \begin{array}{c} \begin{array}{ccccccc} t_2 & t_6 & t_3 & t_4 & t_1 & t_{4.1} & t_5 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

No tocante a lista expandida de lugares *(marcacoes_alcancadas)* e transições *(transicoes_disparadas)* da rede de ocorrências, o *MATLAB* gerou os seguintes resultados:

marcacoes_alcancadas = [1,4,6,2,7,3,4,5,1,5,4,6,4,6]

transicoes_disparadas = [2,6,3,4,1,4,5,5]

5.4.3 Análise de cenário de grande porte

No capítulo 4.1.4 deste trabalho, foi discutido e analisado as três ferramentas utilizadas para desdobramento, sendo utilizado um cenário de grande porte através de um exemplo de planificação descrito pela linguagem PDDL. Os resultados encontrados foram idênticos entre as ferramentas *MOLE* e *CUnf*, porém diferentes quando comparado com a ferramenta *PUnf*.

Esse cenário de grande porte da rede de Petri, sendo composto por 48 lugares e 272 transições, foi simulado utilizando o algoritmo proposto em *MATLAB*, sendo gerado os seguintes resultados:

numero_marcacoes_alcancadas = 2941
numero_transicoes_disparadas = 725
numero_eventos_corte = 577

Assim para uma rede de grande porte, os resultados gerados pelo algoritmo proposto foram idênticos às ferramentas *MOLE* e *CUnf*, porém diferentes aos resultados gerados pela ferramenta *PUnf*, conforme resumo da tabela 5.1.

Tabela 5.1: Comparação de resultados entre as ferramentas analisadas.

	Algoritmo	CUnf	MOLE	PUnf
Condições	2941	2941	2941	2916
Eventos	725	725	725	717
Cut-off	577	577	577	569

Conforme já foi citado, as diferenças observadas nos resultados, pode estar sendo ocasionado pelo conceito de guarda de transição, o qual faz parte da estrutura da ferramenta *PUnf*, não sendo possível validar essa afirmação em virtude da ferramenta estar disponível apenas no formato executável. Foi realizado contato através de *E-mail* como o desenvolvedor da ferramenta, o Prof. Dr. Victor Khomenko, visando compreender a diferença de resultados, porém não foram obtidas informações que auxiliassem de maneira conclusiva.

5.5 Conclusão

Conforme já foi visto, o algoritmo ERV Unfolding, serviu de base para implementação de diversas ferramentas, entre elas, *CUnf, MOLE e PUnf*. Para execução dessas ferramentas, se faz necessário criar um arquivo de entrada com formato da descrição da rede, devendo conter em ordem, elementos que estrturam ou definem a rede, sem os quais impede a operabilidade da ferramenta. Essas ferramentas utilizam estruturas e variáveis já definidas impedindo ao desenvolvedor de fazer qualquer alteração. Já a ferramenta *MATLAB*, o principal elemento de entrada é uma matriz que viabiliza a realização de cálculos necessários para geraração do

desdobramento da rede. O *MATLAB* não se prende a estruturas de entrada pré-definidas, cabendo ao desenvolvedor criá-las de acordo com a sua necessidade, podendo utilizar funções já existentes na ferramenta, apenas para auxiliá-lo no processo de desdobramento.

Para validação do algoritmo proposto, foi realizado um estudo de caso utilizando a rede de Petri da figura 3.5, anteriormente desdobrada pela ferramenta *MOLE*, sendo obtido os mesmos resultados no tocante aos eventos de corte. Ainda como meio de validação, foi simulado o algoritmo utilizando um cenário de grande porte através de um exemplo de planificação descrito pela linguagem PDDL, sendo encontrado resultados idênticos quando comparado às ferramentas *CUnf* e *MOLE*, porém divergentes aos resultados encontrados com a ferramenta *PUnf*.

Capítulo 6

Conclusão

A técnica de desdobramento é uma importante contribuição para problemas de alcançabilidade, uma vez que reduz consideravelmente o custo computacional na enumeração do espaço de estados acessíveis de uma rede de Petri marcada. Entretanto, as ferramentas computacionais existentes, para a geração da rede desdobrada, são complexas e de difícil manutenção, o que dificulta o estudo e a simulação de uma rede de Petri em diferentes cenários. Este trabalho apresentou uma abordagem matricial para a geração do desdobramento de uma rede de Petri segura, utilizando para tanto a ferramenta *MATLAB*, tendo por base o algoritmo ERV *Unfolding*. Nesta pesquisa buscou-se a criação e implementação de uma nova abordagem matricial, diferente das abordagens baseadas em ponteiros e listas, utilizadas pelas ferramentas de desdobramento mais conhecidas, como *CUnf, MOLE* e *PUnf*.

Para alcançar os objetivos desta pesquisa, foram estudados os conceitos básicos sobre redes de Petri e redes de ocorrência. A técnica de desdobramento foi estudada com profundidade, assim como foram analisadas as ferramentas computacionais implementadas. Buscou-se também, na literatura, outras implementações equivalentes a esta proposta, não tendo sido encontrada nenhuma referência. O estudo do algoritmo de ERV *Unfolding* foi necessário, uma vez que serviu de base para a implementação do algoritmo de desdobramento utilizando a ferramenta *MATLAB*. As implementações encontradas na bibliografia como *CUnf, MOLE* e *PUnf*, são soluções que utilizam por base esse mesmo algoritmo, mas com abordagens diferentes. As três ferramentas apresentadas utilizam a linguagem C, com ponteiros e listas. São implementações que tem bom desempenho, mas, em virtude da complexidade e a consequente dificuldade de manutenção do algoritmo, dificultam a alteração do código para estudos de cenários distintos. Com relação ao cenário de testes realizado, para ambientes de pequeno porte, os resultados foram idênticos, porém, quando simulado em um cenário de grande porte, os resultados obtidos foram iguais para as ferramentas *CUnf* e *MOLE*, mas diferente em relação a ferramenta *PUnf*. A divergência nos resultados é decorrente da forma como o algoritmo foi implementado no *PUnf* para gerar o desdobramento, utilizando o conceito de guarda de transição.

A ferramenta *MATLAB* e outras similares, oferece um vasto ferramental dedicado a operações com matrizes, cabendo ao desenvolvedor utilizá-las de acordo com a sua necessidade,

para auxiliá-lo no processo de desdobramento. Além da vantagem de permitir criar estruturas de acordo com a necessidade, a ferramenta permite a realização de operações utilizando matrizes e vetores, para gerar o desdobramento da rede. A dificuldade encontrada foi a implementação do algoritmo ERV *Unfolding*, de acordo com uma abordagem matricial, uma vez que não existem pesquisas que abordem essa maneira de implementar o algoritmo. Visando facilitar o trabalho, o código foi desenvolvido recebendo como entrada as matrizes Pre, Pos e M_0, sendo que durante o processo de desdobramento estruturas de laços e comparações são utilizadas entre as matrizes e vetores visando atingir esse fim, com base no algoritmo ERV *Unfolding*. O resultado gerado são informações referentes ao desdobramento da rede como matrizes pre e pos do desdobramento, as marcações alcançadas e as transições disparadas, e por fim, os eventos de corte.

Como resultado do presente estudo foram submetidos dois artigos junto ao XXIII Simpósio de Engenharia de Produção - SIMPEP, tendo como área temática modelagem, análise e simulação de pesquisa operacional, com os títulos “Aplicação da ferramenta computacional PIPE para simular uma rede de Petri para resolução do problema do menor caminho para uma rede de emergência hospitalar” e “Aplicação de técnica de desdobramento de redes de Petri para resolução do problema do menor caminho para uma rede de emergência na segurança pública”. Ambos os artigos foram aprovados junto ao Simpósio na modalidade de artigos.

Como trabalho futuro, destaca-se a possibilidade de implementar uma abordagem matricial utilizando paralelismo, tornando mais rápido o processo de desdobramento de uma rede de Petri segura. Ainda como trabalho futuro, pode-se sugerir o estudo das propriedades comportamentais e estruturais de acordo com a abordagem matricial implementada. Por fim, sugere-se o estudo com profundidade das ferramentas tradicionais de desdobramento *CUnf, MOLE* e *PUnf*, visando propor uma metodologia de análise, visando identificar o motivo dos resultados para cenários de grande porte, estarem gerando resultados diferentes entre as ferramentas *CUnf/MOLE* e *PUnf*.

Referências Bibliográficas

[Badouel et al., 2015] Badouel, E., Bernardinello, L., and Darondeau, P. (2015). Petri net synthesis. *Journal Theoretical Computer Science*, 13:339.

[Benaccio, 2008] Benaccio, J. H. Q. (2008). Planejamento em inteligência artificial utilizando redes de petri cíclicas. Master's thesis, Pós-Graduação em Informática - Universidade Federal do Paraná, Curitiba - Paraná.

[Benito, 2010] Benito, F. C. V. (2010). Desdobramento para redes de petri k-limitadas. Master's thesis, Pós-Graduação em Informática - Universidade Federal do Paraná, Curitiba - Paraná.

[Benito, 2015] Benito, F. C. V. (2015). *Desdobramento Relaxado para Redes de Petri Temporais*. PhD thesis, Pós-Graduação em Informática - Universidade Federal do Paraná, Curitiba - Paraná.

[Best and Stehno, 2016] Best, E. and Stehno, C. (2016). Parallel systems group. `https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/pep.html`. Acessado em 9/02/2016.

[Bonet et al., 2014] Bonet, B., Haslum, P., Khomenko, V., Thiébaux, S., and Vogler, W. (2014). Recent advances in unfolding technique. *Journal Theoretical Computer Science*, 551:84–101.

[Cabasino et al., 2010] Cabasino, M. P., Giua, A., Pocci, M., and Seatzu, C. (2010). Discrete event diagnosis using labeled nets. an application to manufacturing systems. *Journal Control Engineering Practice*, 19:989–1001.

[Cardoso and Valette, 1997] Cardoso, J. and Valette, R. (1997). *Redes de Petri*. Editora da Universidade Federal de Santa Catarina.

[Cassandras, 1999] Cassandras, C. G. (1999). *Introduction to Discrete Event Systems*. Editora Springer.

[Chapman, 2010] Chapman, S. J. (2010). *Programação em MATLAB para Engenheiros*. Editora Cengage Learning, São Paulo, São Paulo.

[Dias, 2007] Dias, G. L. (2007). Ferramentas para a integração de redes de petri e vhdl na síntese de sistemas digitais. Master's thesis, Pós-Graduação em Engenharia Elétrica - Universidade Estadual Paulista, Ilha Solteira - São Paulo.

[Dijkstra, 1971] Dijkstra, E. W. (1971). Hierarchical ordering of sequential processes. *Journal Formal Methods in System Design*, 6:115–138.

[Dubrau, 2012] Dubrau, A. (2012). Taming matlab. Master's thesis, Pós-Graduação em Ciências da Computação - McGill University, Montréal - Canadá.

[Elsevier, 2016] Elsevier (2016). Scopus - ferramenta de pesquisa. `https://www.scopus.com`. Acessado em 18/04/2016.

[Esparza and Heljanko, 2008] Esparza, J. and Heljanko, K. (2008). *Unfolding - A Partial Orders Approach to Model Checking*. Editora Springer, Berlin, Alemanha.

[Esparza et al., 2002] Esparza, J., Römer, S., and Vogler, W. (2002). An improvement of mcmillan's unfolding algorithm. *Journal Formal Methods in System Design*, 20(3):285–310.

[Esparza and Schröter, 2001] Esparza, J. and Schröter, C. (2001). Unfolding based algorithms for reachability problem. *Journal Fundamenta Informaticae*, 47:231–245.

[Han and Kamber, 2011] Han, J. and Kamber, M. (2011). *Data Mining: Concepts and Techniques*. Editora Morgan Kaufmann, San Francisco, Estados Unidos.

[Irturk, 2009] Irturk, A. U. (2009). *GUSTO: General architecture design Utility and Synthesis Tool for Optimization*. PhD thesis, Pós-Graduação em Ciência da Computação - Universidade de California, San Diego - Estados Unidos.

[Karp and Miller, 1969] Karp, R. M. and Miller, R. E. (1969). Parallel program schemata. *Journal Computer System Science*, 3:174–195.

[Khomenko, 2016] Khomenko, V. (2016). Unfolding tools. `http://homepages.cs.ncl.ac.uk/victor.khomenko/home.formal/tools/UnfoldingTools/current/`. Acessado em 9/02/2016.

[MathWorks, 2016] MathWorks (2016). The language of technical computing. `http://www.mathworks.com/products/matlab/?requestedDomain=www.mathworks.com/`. Acessado em 10/02/2016.

[McMillan, 1995] McMillan, K. L. (1995). A technique of state space search based on unfolding. *Journal Formal Methods in System Design*, 6(1):45–65.

[McMillan and Probst, 1995] McMillan, K. L. and Probst, D. K. (1995). A technique of state space search based on unfolding. *Journal Acta Informatica*, 1:45–65.

[Murata, 1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Journal of the Proceedings of the IEEE*, 77(4):541–580.

[PEP, 2016] PEP (2016). Format used in the pep tool. `http://www2.informatik.uni-stuttgart.de/fmi/szs/tools/mckit/pep.shtml`. Acessado em 9/02/2016.

[Petri, 1962] Petri, C. A. (1962). *Kommunication mit Automaten*. PhD thesis, Institutfür Instrumentelle Mathematik, Bonn - Alemanha.

[Prasad, 2012] Prasad, A. (2012). Automatic compilation of matlab programs for synergistic execution on heterogeneous processors. Master's thesis, Pós-Graduação em Ciências - Faculdade de Engenharia do Instituto Indiano de Ciências, Bangalore - India.

[Rodríguez and Schwoon, 2013] Rodríguez, C. and Schwoon, S. (2013). Cunf: A tool for unfolding and verifying petri nets with read arcs. *Journal Computer Science*, 8172:492–495.

[Santos et al., 2014] Santos, B. F., Villanueva, J. M. M., and Costa, M. M. (2014). Modelagem de uma turbina eólica utilizando redes de petri para controle de parques eólicos. *XX Congresso Brasileiro de Automática*, pages 1592–1599.

[Scappini, 2010] Scappini, R. J. R. (2010). Estudio del tráfico autosimilar orientado a la simulación mediante la utilización de wavelets. Master's thesis, Pós-Graduação em Redes de Dados - Universidad Nacional de la Plata, Argentina.

[Schwoon and Romer, 2016] Schwoon, S. and Romer, S. (2016). Mole - a petri net unfolder. `http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/`. Acessado em 9/02/2016.

[Taubin et al., 1998] Taubin, A., Kondratyev, A., and Kishinevsky, M. (1998). Deadlock prevention using petri nets and their unfoldings. *Journal Advanced Manufacturing Tachnolog*, 14:750–759.