**Universidade Federal de Pernambuco**

Graduação em Ciência da Computação

Centro de Informática

2015.1

Avaliação de algoritmos para conversão de Redes de Petri Estocásticas para Cadeias de Markov de Tempo Contínuo

**Proposta de Trabalho de Graduação**

**Aluno:** Marcelo de Lima Rodrigues (mlr@cin.ufpe.br)

**Orientadora:** Eduardo Antônio Guiamarães Tavares (eagt@cin.ufpe.br)

Sumário

[Contexto 3](#_Toc401774141)

[Objetivo 4](#_Toc401774142)

[Cronograma 5](#_Toc401774143)

[Possíveis Avaliadores 6](#_Toc401774144)

[Referências 7](#_Toc401774145)

[Assinaturas 8](#_Toc401774146)

# Contexto

Redes de Petri é uma família de modelos formais~~,~~ a qual possui uma representação matemática e possui mecanismos que permitem a análise e verificação de propriedades. Usando-se redes de Petri, pode-se modelar diversos tipos de sistemas, inclusive sistemas de grande porte e *datacenters*. Uma rede de Petri possui dois componentes básicos: um ativo denominado transição e um passivo denominado lugar. Os lugares correspondem às variáveis de estado, onde cada lugar tem uma determinada marcação(*tokens)* associada. As transições correspondem às ações realizadas pelo sistema. A realização de uma ação está associada a algumas condições(condição de alguma variável de estado), e de forma semelhante, após a realização de uma ação, algumas variáveis de estado(lugares) terão suas informações alteradas. Fora esse modelo básico, as redes de Petri possuem diversas extensões. Quando associamos tempo a cada transição e assume-se uma distribuição exponencial, e nós temos uma rede de Petri estocástica (SPN).

As redes de Petri estocásticas, entre outros elementos, possuem arcos-inibidores e transições imediatas, nessa ultima, o tempo associado é zero, e por isso, elas possuem precedência de disparo em relação as transições que associam tempo. Nesse caso, a rede passa a se chamar de rede de Petri estocástica generalizada (GSPN).

Em razão da Rede de Petri ser um modelo baseado em estados, é particularmente útil para representar sistemas, pois podem facilmente considerar dependências, por isso a representação de mecanismos mais complexos redundantes, bem como políticas de manutenção sofisticadas, é facilitada. Entretanto, essas técnicas de avaliação podem ser penalizadas com o grande aumento no tamanho do espaço de estados para representação de alguns sistemas mais complexos. A partir de sistemas bem representados, pode-se realizar análises de performabilidade (desempenho e dependabilidade) neles.

Para realizara análise de performabilidade das SPNs, deve-se gerar o espaço de estados da rede de Petri estocástica. O espaço de estados de uma SPN é isomorfo a cadeia de Markov de tempo contínuo (CTMC), onde a partir da CTMC, representada através de uma matriz de números reais, pode-se realizar análises estáticas e transientes utilizando métodos específicos de cálculo numérico (e.g. Gauss-Seidel).

Quando consideramos GSPN, a geração do espaço de estados torna-se diferente da SPN. O espaço de estados de uma GSPN possui dois tipos de estados: estados *vanish* e estados *tangible.* Os estados *tangible* são espaços que possuem exclusivamente transições exponenciais aptas a disparar, enquanto que os estados *vanish* são estados onde apenas transições imediatas estão aptas a dispararem. A particularidade do espaço de estados de uma GSPN consiste que os estados *vanish* não devem ser incluídos no espaço de estados, e por isso, devem-se realizar reduções nele a fim de eliminar esses estados. Ao fim do processo, o espaço de estados reduzido dessa GSPN torna-se isomorfo a CTMC associado à SPN, e o processo torna-se o mesmo da SPN. [4]

Para realizar a redução do espaço de estados de uma GSPN (eliminar os estados *vanish),* temos duas abordagens: realizar a eliminação “on-the-fly”, ou seja, a medida que o algoritmo de geração do espaço de estados encontrar um estado do tipo *vanish*, o mesmo será eliminado. O resultado final será o espaço de estados, correspondente a GSPN de entrada, devidamente reduzido. A outra abordagem seria um algoritmo “post-reduction”, em que o espaço de estados é gerado completo, incluindo os estados *vanish*. Após a geração, será realizada a remoção (*pruning*) dos estados *vanish*. É importante notar que o resultado dos dois algoritmos deve ser exatamente igual, apenas os meios em que ele atingirá o resultado final serão diferentes.

# Objetivo

O objetivo desse trabalho é implementar dois algoritmos, os quais irão utilizar as duas metodologias previamente descritas para geração do espaço de estados de uma rede de Petri estocástica generalizada (GSPN). Um dos algoritmos irá utilizar a metodologia “on-the-fly”, e o outro irá utilizar a metodologia “post-reduction” para gerar o espaço de estados. Os algoritmos serão integrados no ambiente Mercury, um ambiente de software voltado para avaliação de performabilidade de sistemas, que utiliza modelos formais de redes de Petri estocásticas e diagramas de blocos de confiabilidade (RBD) para realizar essa avaliação. Assim, quando o usuário for utilizar o ambiente Mercury para realizar uma avaliação de performabilidade utilizando GSPN, ele poderá escolher qual algoritmo será utilizado para gerar o espaço de estados.

O primeiro momento seria dedicado a implementar o algoritmo “on-the-fly”. Para isso, será necessário realizar estudos para definir quais estruturas de dados serão utilizadas pelo algoritmo, bem como definir alguns aspectos internos ao algoritmo, como por exemplo, qual o algoritmo de busca será utilizado. Após a implementação, é fundamental e imprescindível que sejam realizados exaustivos testes no algoritmo, utilizando GSPNs diversificadas e que abranjam o maior número de casos diferentes.

 Após concluído o processo descrito acima para o algoritmo “on-the-fly”, será iniciado o desenvolvimento do algoritmo utilizando a metodologia “post-reduction”. Será utilizada a mesma sistemática descrita acima: inicialmente serão definidas estruturas de dados e algoritmos que serão utilizados internamente no algoritmo “post-reduction”. Assim como no algoritmo “on-the-fly”, devem ser realizados diversos testes para verificar corretude do algoritmo.

Os espaços de estados produzidos por esses algoritmos devem ser exatamente iguais, apenas diferindo pela metodologia utilizada.

 Posteriormente ao desenvolvimento, será realizada uma análise e comparação desses dois algoritmos, inclusive em relação a métricas, tais como: consumo de memória e processamento.

# Cronograma

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Atividades** | **Março** | **Abril** | **Maio** | **Junho** | **Julho** |
| Pesquisa e levantamento bibliográfico |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| implementação do algoritmo “on-the-fly” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testes do algoritmo “on-the-fly” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementação do algoritmo “post-reduction” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testes do algoritmo “post-reduction” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Análise e comparação dos algoritmos  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Elaboração do projeto e apresentação |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Possíveis Avaliadores

Eduardo Antônio Guimarães Tavares

# Bibliografia

[1] Meyer, J. F. On *Evaluating the Performability of Degradable Computing Systems*. *IEEE Transactions on Computers*. v. 29, n. 8, 1980.

 [2] Misra, K. *Handbook of Performability Engineering.* Springer, Ed. 1a, 2008.

 [3] Marwah, M.; e outros. *Quantifying Sustainability Impact of Data Center Availability.* *GreenMetrics*, Seattle, 2009.

 [4]MARSAN, M. Ajmone; BALBO, Gianfranco; CONTE, Gianni. *Modelling with generalised stochastic petri nets.*Torino: [s.n.], 1980.

[5]CHAUDRON, M.r.v.. et al. *Reliability block diagrams analysis and tactics.*Eindhoven: [s.n.], 2000.

#

# Assinaturas

Recife, Pernambuco. Brasil

13 de Abril de 2015

 Eduardo Antônio Guimarães Tavares

(orientador)

 Marcelo de Lima Rodrigues

(proponente)