PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO PARA CONVERSÃO AUTOMÁTICA DE MODELOS SAN GTA PARA MODELOS SAN CTA

PAULO GUILHERME GIL

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Paulo Fernandes

Porto Alegre
2013

FICHA CATALOGRÁFICA EMITIDA PELA BIBLIOTECA
SUBSTITUA

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO
EMITIDA E ASSINADA PELA FACULDADE

Dedico as pessoas que me ajudaram no decorrer deste trabalho.

ALGORITMO PARA CONVERSÃO AUTOMÁTICA DE MODELOS SAN GTA PARA MODELOS SAN CTA

RESUMO

Este trabalho apresenta o formalismo para modelagem de sistemas chamado Redes de Autômatos Estocásticos (SAN). O formalismo SAN tem o objetivo de aumentar o nível de abstração e oferece uma alternativa de modelagem mais sofisticada do que a proporcionada pelas tradicionais Cadeias de Markov (MC). Este formalismo utiliza a álgebra tensorial clássica (CTA) e geralizada (GTA) para simplificar a matriz das transições entre os estados do modelo. Embora todos os modelos SAN descritos utilizando GTA possuam pelo menos um modelo equivalente descrito utilizando CTA, e que a solução de certos modelos utilizando CTA possa ser mais rápido que o modelo equivalente que utiliza GTA, este trabalho propõe um algoritmo para traduzir um modelo descrito em GTA para o modelo equivalente descrito em CTA. Espera-se com isto permitir que um modelo descrito utilizando funções (usando GTA) possa ser resolvido mais rapidamente ou ocupando menos memória através da solução de seu modelo convertido para CTA.

Palavras-chave: Modelagem computacional; Formalismos estruturados; Soluções numéricas; Cadeias de Markov; Redes de Autômatos Estocásticos; Descritores Markovianos; Ferramentas computacionais; Álgebra Tensorial.

AUTOMATIC CONVERSION OF SAN GTA MODELS TO SAN CTA MODELS

ABSTRACT

This work presents a formalism for modeling systems called Stochastic Automata Networks (SAN), SAN formalism aims to increase the abstraction's level and provides a sophisticated alternative model to the tadicional formalism of Markov Chains (MC). SAN uses both Classical (CTA) and Generalized Tensor Algebra (GTA) to simplify the matrix of transitions between states of the model. Despite all models described with GTA having at least one equivalent model described using CTA, and that the solution of certain models based on CTA could be faster than the equivalent GTA based model, this dissertation proposes an algorithm for translating a model described in GTA into the equivalent model described in CTA. It is expected that some models described using functions (using GTA) could be solved more quickly or taking less memory through the solution of its CTA-converted model.

Keywords: Computational modeling; Structured formalisms; Numerical solutions; Markov Chains; Stochastic Automata Networks; Markovian Descriptors; Computational tools; Tensor Algebra.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CTA	*Classical Tensor Algebra*
GTA	*Generalized Tensor Algebra*
MC	*Markov Chains*
PEPA	*Performance Evaluation Process Algebra*
PEPS	*Performance Evaluation of Parallel Systems*
SAN	*Stochastic Automata Networks*
SPN	*Stochastic Petri Nets*

SUMÁRIO

1. INTRODUÇÃO

Em 1865, o matemático alemão Leopold Kronecker propôs uma nova operação baseada em tensores, permitindo a representação de matrizes com mais de duas dimensões. Esta operação tornou-se uma extensão da álgebra linear ficando conhecido como produto Kronecker ou produto tensorial [11].

Desde então muitos trabalhos foram feitos utilizando o produto tensorial para representar operações em estruturas multidimensionais.

De acordo com Brenner, Fernandes e Sales [4] , somente no final da década de 1970, foram feitos estudos na área da Ciência da Computação utilizando as extensões de Kronecker para a álgebra linear, entre eles [1, 2, 6, 8]. Estes trabalhos levaram a criação de outra operação, a soma tensorial, que juntamente com o produto tensorial levou a criação de uma nova álgebra, chamada de Álgebra Tensorial Clássica (Classical Tensor Algebra - CTA).

Plateau [12] propôs em 1984 o primeiro formalismo para modelagem de sistemas representado através de Álgebra Tensorial Clássica: as Redes de Autômatos Estocásticos (Stochastic Automata Networks - SAN). Outros dois formalismos para modelagem de sistemas foram também representados utilizando CTA: as Redes de Petri Estocásticas (Stochastic Petri Nets - SPN) e Álgebra de Processos para Avaliação de Desempenho (Performance Evaluation Process Algebra - PEPA).

Como representações estruturadas de Cadeias de Markov, estes formalismos simplificam a imensa matriz com as transições entre os estados esparsas utilizando pequenas matrizes operadas por álgebra tensorial, possibilitando assim, uma modelagem mais compacta e otimizando o espaço de armazenamento do modelo.

Como um formalismo de modelagem de desempenho de sistemas, as Redes de Autômatos Estocásticos baseiam-se na descrição de um sistema complexo dividindo-o em vários subsistemas com comportamentos independentes e interdependências ocasionais através de taxas funcionais e eventos sincronizantes. Cada subsistema é representado por um autômato estocástico composto por estados e transições entre estes estados.

A representação algébrica de modelos estruturados utilizando SAN é denominada de descritor Markoviano, que pode utilizar os dois tipos de álgebra tensorial, a clássica (CTA) e a generalizada (GTA). Ou seja, o descritor markoviano é uma formulação matemática que utiliza álgebra tensorial.

1.1 Motivação

Os algoritmos que são responsáveis por resolver os descritores Makrovianos e gerar as probabilidades de permanência nos estados da SAN podem ocupar menos memória ou ser mais rápidos dependendo da abordagem utilizada na geração do descritor.

Sabe-se que toda SAN descrita com primitivas funcionais possui pelo menos uma SAN descrita somente com taxas constantes [5]. A motivação do trabalho é definir formalmente um tradutor que

converta qualquer modelo SAN que foi descrito utilizando primitivas GTA para um modelo SAN que somente utilize CTA na sua definição. Já que pode ser melhor um modelo convertido é interessante que se possa automaticamente gerar um modelo melhor antes de resolvê-lo.

1.2 Objetivo

O principal objetivo do trabalho é propor um método de conversão automática de modelos SAN que são descritos utilizando primitivas funcionais na sua definição por modelos equivalentes, porém que somente utilizem eventos locais e sincronizantes com taxas constantes. Entende-se por equivalência a geração da mesma cadeia de Markov subjacente.

Ao alcançar o objetivo, espera-se que este trabalho possa ajudar nas pesquisas de análise de desempenho no que diz respeito a procurar alternativas para minimizar o custo computacional da solução de modelos complexos.

1.3 Metodologia

Este trabalho traz um detalhamento do uso da álgebra tensorial nos formalismos para modelagem de sistemas, é apresentado o formalismo de Redes de Automatos Estocásticos e como suas matrizes são escritas no formato tensorial.

O algoritmo é definido formalmente e validado através de um conjunto de modelos que buscam representar todas as combinações de características que possuem relação com o conversor. Por fim o algoritmo será implementado na ferramenta PEPS para validar o seu funcionamento.

1.4 Organização

A dissertação está organizada da seguinte forma. No Capítulo 2 serão apresentadas a Álgebra Tensorial Clássica e a Álgebra Tensorial Generalizada. No Capítulo 3 será apresentado o formalismo de Redes de Autômatos Estocásticos. No Capítulo 4 será definido o algoritmo para a conversão de modelos SAN descritos utilizando primitivas GTA para modelos que somente utilizam primitivas CTA. Este algoritmo será validado com a maior combinação possível de modelos com características diferentes.

2. ÁLGEBRA TENSORIAL

Neste capítulo serão apresentados os conceitos de Álgebra Tensorial Clássica [1] e Generalizada [2]. Esses conceitos são utilizados pelos formalismos para modelagem de sistemas baseados em estruturas tensoriais, como as Redes de Autômatos Estocásticos [2].

2.1 Álgebra Tensorial Clássica

A seguir são detalhadas as operações de produto e soma tensorial, que são as operações da Álgebra Tensorial Clássica, também se fala de um caso particular de produto tensorial, os fatores normais.

2.1.1 Produto Tensorial

Dadas as matrizes:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

O produto tensorial das matrizes $C = A \otimes B$ é igual a:

$$C = \begin{pmatrix} a_{1,1}B & a_{1,2}B \\ a_{2,1}B & a_{2,2}B \end{pmatrix}$$

Ou:

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,1}b_{1,3} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} & a_{1,2}b_{1,3} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,1}b_{2,3} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} & a_{1,2}b_{2,3} \\ a_{1,1}b_{3,1} & a_{1,1}b_{3,2} & a_{1,1}b_{3,3} & a_{1,2}b_{3,1} & a_{1,2}b_{3,2} & a_{1,2}b_{3,3} \\ \hline a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,1}b_{1,3} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} & a_{2,2}b_{1,3} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,1}b_{2,3} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} & a_{2,2}b_{2,3} \\ a_{2,1}b_{3,1} & a_{2,1}b_{3,2} & a_{2,1}b_{3,3} & a_{2,2}b_{3,1} & a_{2,2}b_{3,2} & a_{2,2}b_{3,3} \end{array}\right)$$

Analisando as dimensões do produto tensorial definido acima, observa-se que a matriz C é o produto cartesiano dos elementos da matriz A com os elementos da matriz B. Ou seja, a dimensão da matriz do produto tensorial da matriz A de dimensão (x, y) com a matriz B de dimensão (p, q) será (xp, yq).

Também pode-se dizer que a matriz do produto tensorial é composta por blocos de dimensões (x, y) preenchidos internamente com elementos de dimensão (p, q). Resumindo este conceito, um elemento da matriz $C_{[xp][yq]} = A_{x,y}B_{p,q}$.

2.1.2 Fator Normal

O produto tensorial de uma matriz quadrada por uma matriz identidade se denomina como Fator Normal. Será utilizada a nomenclatura I_n como sendo uma matriz identidade de tamanho $n * n$.

Dadas as matrizes:

$$A_2 = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Com as matrizes acima pode-se obter dois fatores normais, sejam:

$$A \otimes I_3 = \left(\begin{array}{ccc|ccc} a_{1,1} & 0 & 0 & a_{1,2} & 0 & 0 \\ 0 & a_{1,1} & 0 & 0 & a_{1,2} & 0 \\ 0 & 0 & a_{1,1} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2} & 0 & 0 \\ 0 & a_{2,1} & 0 & 0 & a_{2,2} & 0 \\ 0 & 0 & a_{2,1} & 0 & 0 & a_{2,2} \end{array}\right)$$

$$I_3 \otimes A = \left(\begin{array}{cc|cc|cc} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & 0 & a_{2,1} & a_{2,2} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a_{1,1} & a_{1,2} \\ 0 & 0 & 0 & 0 & a_{2,1} & a_{2,2} \end{array}\right)$$

O produto tensorial de duas matrizes identidade gera uma matriz cuja dimensão é o produto das dimensões das duas matrizes:

$$I_n \otimes I_m = I_m \otimes I_n = I_{nm}$$

2.1.3 Soma Tensorial

Para calcular a soma tensorial, é necessário utilizar o conceito de produto tensorial. A soma tensorial de duas matrizes A e B é dada pelo produto tensorial da primeira matriz com a Identidade

do tamanho da segunda matriz somado ao produto tensorial da Identidade do tamanho da primeira matriz com a segunda. A fórmula abaixo representa a soma tensorial:

$$A \oplus B = (A \otimes I_{n_B}) + (I_{n_A} \otimes B)$$

Onde n_x é o tamanho de uma matriz quadrada x.
Como exemplo da soma tensorial, podemos utilizar duas matrizes:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

A soma tensorial dessas matrizes é:

$$A \oplus B = \left(\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) + \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \right)$$

Calculando os fatores normais, obtem-se:

$$A \oplus B = \left(\begin{array}{ccc|ccc} a_{1,1} & 0 & 0 & a_{1,2} & 0 & 0 \\ 0 & a_{1,1} & 0 & 0 & a_{1,2} & 0 \\ 0 & 0 & a_{1,1} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2} & 0 & 0 \\ 0 & a_{2,1} & 0 & 0 & a_{2,2} & 0 \\ 0 & 0 & a_{2,1} & 0 & 0 & a_{2,2} \end{array} \right) + \left(\begin{array}{ccc|ccc} b_{1,1} & b_{1,2} & b_{1,3} & 0 & 0 & 0 \\ b_{2,1} & b_{2,2} & b_{2,3} & 0 & 0 & 0 \\ b_{3,1} & b_{3,2} & b_{3,3} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & b_{1,1} & b_{1,2} & b_{1,3} \\ 0 & 0 & 0 & b_{2,1} & b_{2,2} & b_{2,3} \\ 0 & 0 & 0 & b_{3,1} & b_{3,2} & b_{3,3} \end{array} \right)$$

Resolvendo a soma das matrizes:

$$A \oplus B = \left(\begin{array}{ccc|ccc} a_{1,1}+b_{1,1} & b_{1,2} & b_{1,3} & a_{1,2} & 0 & 0 \\ b_{2,1} & a_{1,1}+b_{2,2} & b_{2,3} & 0 & a_{1,2} & 0 \\ b_{3,1} & b_{3,2} & a_{1,1}+b_{3,3} & 0 & 0 & a_{1,2} \\ \hline a_{2,1} & 0 & 0 & a_{2,2}+b_{1,1} & b_{1,2} & b_{1,3} \\ 0 & a_{2,1} & 0 & b_{2,1} & a_{2,2}+b_{2,2} & b_{2,3} \\ 0 & 0 & a_{2,1} & b_{3,1} & b_{3,2} & a_{2,2}+b_{3,3} \end{array} \right)$$

2.1.4 Propriedades

Abaixo são mostradas as principais propriedades da Álgebra Tensorial Clássica:

- Associativa

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{2.1}$$

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C \tag{2.2}$$

- Distributiva

$$(A+B) \otimes (C+D) = (A \otimes C) + (B \otimes C) + (A \otimes D) + (B \otimes D) \tag{2.3}$$

- Compatibilidade com a multiplicação

$$(A \times B) \otimes (C \times D) = (A \otimes C) \times (B \otimes D) \tag{2.4}$$

- Compatibilidade com a transposição de matrizes

$$(A \otimes B)^T = A^T \otimes B^T \tag{2.5}$$

- Compatibilidade com a inversão de matrizes

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{2.6}$$

- Decomposição de fatores normais

$$(A \otimes B) = (A \otimes I_{n_B}) x (I_{n_a} \otimes B) \tag{2.7}$$

- Distributividade com relação à multiplicação pela matriz identidade

$$(A \times B) \otimes I_n = (A \otimes I_n) \times (B \otimes I_n) \tag{2.8}$$

$$I_n \otimes (A \times B) = (I_n \otimes A) \times (I_n \otimes B) \tag{2.9}$$

- Comutatividade dos fatores normais

$$(A \otimes I_{n_B}) \times (I_{n_A} \otimes B) = (I_{n_A} \otimes B) \times (A \otimes I_{n_B}) \tag{2.10}$$

- Pseudo Comutatividade

$$\bigotimes_{k=1}^{N} A^{(k)} = P_\sigma \times \left[\bigotimes_{k=1}^{N} A^{(\sigma_k)} \right] \times {P_\sigma}^T \tag{2.11}$$

2.2 Álgebra Tensorial Generalizada

A Álgebra Tensorial Generalizada é uma amplicação da Álgebra Tensorial Clássica. Na Álgebra Tensorial Generalizada, os elementos das matrizes podem ser, além de elementos constantes (pertencentes a $\mathbb{R}$), elementos funcionais. Um elemento funcional é composto pelos índices das linhas de uma ou mais matrizes [7].

Caso algum índice de uma linha da matriz A pertencer a uma função f, diz-se que a função f é dependente da matriz A. A Matriz que possuir a função f (que é dependente da Matriz A) também é denominada dependente da matriz A.

O produto tensorial generalizado é representado pelo operador $\otimes_g$ e a soma tensorial generalizada é representada pelo operador $\oplus_g$.

2.2.1 Produto Tensorial Generalizado

Dada a matriz $A(\mathcal{B})$:

$$A(\mathcal{B}) = \begin{pmatrix} a_{1,1}[\mathcal{B}] & a_{1,2}[\mathcal{B}] \\ a_{2,1}[\mathcal{B}] & a_{2,2}[\mathcal{B}] \end{pmatrix}$$

Dada a matriz $B(\mathcal{A})$:

$$B(\mathcal{A}) = \begin{pmatrix} b_{1,1}[\mathcal{A}] & b_{1,2}[\mathcal{A}] & b_{1,3}[\mathcal{A}] \\ b_{2,1}[\mathcal{A}] & b_{2,2}[\mathcal{A}] & b_{2,3}[\mathcal{A}] \\ b_{3,1}[\mathcal{A}] & b_{3,2}[\mathcal{A}] & b_{3,3}[\mathcal{A}] \end{pmatrix}$$

O produto tensorial das matrizes $C = A(\mathcal{B}) \otimes_g B(\mathcal{A})$ é igual a:

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}(b_1)b_{1,1}(a_1) & a_{1,1}(b_1)b_{1,2}(a_1) & a_{1,1}(b_1)b_{1,3}(a_1) & a_{1,2}(b_1)b_{1,1}(a_1) & a_{1,2}(b_1)b_{1,2}(a_1) & a_{1,2}(b_1)b_{1,3}(a_1) \\ a_{1,1}(b_2)b_{2,1}(a_1) & a_{1,1}(b_2)b_{2,2}(a_1) & a_{1,1}(b_2)b_{2,3}(a_1) & a_{1,2}(b_2)b_{2,1}(a_1) & a_{1,2}(b_2)b_{2,2}(a_1) & a_{1,2}(b_2)b_{2,3}(a_1) \\ a_{1,1}(b_3)b_{3,1}(a_1) & a_{1,1}(b_3)b_{3,2}(a_1) & a_{1,1}(b_3)b_{3,3}(a_1) & a_{1,2}(b_3)b_{3,1}(a_1) & a_{1,2}(b_3)b_{3,2}(a_1) & a_{1,2}(b_3)b_{3,3}(a_1) \\ \hline a_{2,1}(b_1)b_{1,1}(a_2) & a_{2,1}(b_1)b_{1,2}(a_2) & a_{2,1}(b_1)b_{1,3}(a_2) & a_{2,2}(b_1)b_{1,1}(a_2) & a_{2,2}(b_1)b_{1,2}(a_2) & a_{2,2}(b_1)b_{1,3}(a_2) \\ a_{2,1}(b_2)b_{2,1}(a_2) & a_{2,1}(b_2)b_{2,2}(a_2) & a_{2,1}(b_2)b_{2,3}(a_2) & a_{2,2}(b_2)b_{2,1}(a_2) & a_{2,2}(b_2)b_{2,2}(a_2) & a_{2,2}(b_2)b_{2,3}(a_2) \\ a_{2,1}(b_3)b_{3,1}(a_2) & a_{2,1}(b_3)b_{3,2}(a_2) & a_{2,1}(b_3)b_{3,3}(a_2) & a_{2,2}(b_3)b_{3,1}(a_2) & a_{2,2}(b_3)b_{3,2}(a_2) & a_{2,2}(b_3)b_{3,3}(a_2) \end{array}\right)$$

Os elementos da matriz A variam conforme os elementos da matriz B, ou seja, o elemento $a_{1,1}(b_1)$ é o elemento $a_{1,1}$ da matriz A avaliado para a linha b_1 da matriz B. O mesmo ocorre com os elementos da matriz B que variam conforme os elementos da matriz A.

A definição algebrica do produto tensorial generalizado $C = A(\mathcal{B}) \otimes_g B(\mathcal{A})$ é feita pela atribuição do valor $a_{i,j}(b_k)b_{k,l}(a_i)$ ao elemento $c_{[i,k][j,l]}$.

$$c_{[i,k][j,l]} = a_{i,j}(b_k)b_{k,l}(a_i) \tag{2.12}$$

Onde $i, j \in [1..n_A]$ e $k, l \in [1..n_B]$.

2.2.2 Soma Tensorial Generalizada

A definição da soma tensorial generalizada é feita utilizando-se o produto tensorial generalizado da equacao 2.13

$$A \bigoplus_g B = (A \bigotimes_g I_{n_B}) + (I_{n_A} \bigotimes_g B) \tag{2.13}$$

Utilizando as matrizes $A(\mathcal{B})$ e $B(\mathcal{A})$ apresentadas anteriormente para definir o produto tensorial generalizado. A soma tensorial generalizada definida por $C = A(\mathcal{B}) \bigoplus_g B(\mathcal{A})$ é definida por:

$$C = \left(\begin{array}{ccc|ccc} a_{1,1}(b_1)+b_{1,1}(a_1) & b_{1,2}(a_1) & b_{1,3}(a_1) & a_{1,2}(b_1) & 0 & 0 \\ b_{2,1}(a_1) & a_{1,1}(b_2)+b_{2,2}(a_1) & b_{2,3}(a_1) & 0 & a_{1,2}(b_2) & 0 \\ b_{3,1}(a_1) & b_{3,2}(a_1) & a_{1,1}(b_3)+b_{3,3}(a_1) & 0 & 0 & a_{1,2}(b_3) \\ \hline a_{2,1}(b_1) & 0 & 0 & a_{2,2}(b_1)+b_{1,1}(a_2) & b_{1,2}(a_2) & b_{1,3}(a_2) \\ 0 & a_{2,1}(b_2) & 0 & b_{2,1}(a_2) & a_{2,2}(b_2)+b_{2,2}(a_2) & b_{2,3}(a_2) \\ 0 & 0 & a_{2,1}(b_3) & b_{3,1}(a_2) & b_{3,2}(a_2) & a_{2,2}(b_3)+b_{3,3}(a_2) \end{array}\right)$$

Seja $\delta_{i,j}$ o elemento da linha i e da coluna j de uma matriz identidade. A definição algebrica da soma tensorial generalizada $C = A(\mathcal{B}) \bigoplus_g B(\mathcal{A})$ é definida pela atribuição do valor $a_{i,j}(b_k)\delta_{k,l} + b_{k,l}(a_i)\delta_{i,j}$ ao elemento $c_{[i,k],[j,l]}$, :

$$c_{[i,k],[j,l]} = a_{i,j}(b_k)\delta_{k,l} \ + \ b_{k,l}(a_i)\delta_{i,j} \tag{2.14}$$

Onde $i,j \in [1..n_A]$ e $k,l \in [1..n_B]$.

2.2.3 Propriedades

Abaixo são mostradas as propriedades fundamentais da Álgebra Tensorial Generalizada [10]:

- Distributividade do produto tensorial generalizado em relação à soma comum de matrizes:

  $[A(\mathcal{C},\mathcal{D}) + B(\mathcal{C},\mathcal{D})] \bigotimes_g [C(\mathcal{A},\mathcal{B}) + D(\mathcal{A},\mathcal{B})] =$

$$A(\mathcal{C},\mathcal{D}) \bigotimes_g C(\mathcal{A},\mathcal{B}) \ + \ A(\mathcal{C},\mathcal{D}) \bigotimes_g D(\mathcal{A},\mathcal{B}) \ + B(\mathcal{C},\mathcal{D}) \bigotimes_g C(\mathcal{A},\mathcal{B}) \ + \ B(\mathcal{C},\mathcal{D}) \bigotimes_g D(\mathcal{A},\mathcal{B}) \tag{2.15}$$

- Associatividade do produto tensorial generalizado e da soma tensorial generalizada:

$$[\ A(\mathcal{B},\mathcal{C}) \bigotimes_g B(\mathcal{A},\mathcal{C})\ ] \bigotimes_g C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \bigotimes_g [\ B(\mathcal{A},\mathcal{C}) \bigotimes_g C(\mathcal{A},\mathcal{B})\ ] \tag{2.16}$$

$$[\ A(\mathcal{B},\mathcal{C}) \bigoplus_g B(\mathcal{A},\mathcal{C})\ ] \bigoplus_g C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \bigoplus_g [\ B(\mathcal{A},\mathcal{C}) \bigoplus_g C(\mathcal{A},\mathcal{B})\ ] \tag{2.17}$$

- Distributividade com relação à multiplicação pela matriz identidade:

$$[\ A(\mathcal{C}) \times B(\mathcal{C})\ ] \bigotimes_g I_{n_C} = A(\mathcal{C}) \bigotimes_g I_{n_C} \ \times \ B(\mathcal{C}) \bigotimes_g I_{n_C} \tag{2.18}$$

$$[\, I_{n_C} \bigotimes_g A(\mathcal{C})\,] \times B(\mathcal{C}) = I_{n_C} \bigotimes_g A(\mathcal{C}) \times I_{n_C} \bigotimes_g B(\mathcal{C}) \tag{2.19}$$

- Decomposição em fatores normais I:

$$A \bigotimes_g B(\mathcal{A}) = I_{n_A} \bigotimes_g B(\mathcal{A}) \times A \bigotimes_g I_{n_B} \tag{2.20}$$

- Decomposição em fatores normais II:

$$A(\mathcal{B}) \bigotimes_g B = A(\mathcal{B}) \bigotimes_g I_{n_B} \times I_{n_A} \bigotimes_g B \tag{2.21}$$

- Pseudo-comutatividade:

$$A(\mathcal{B}) \bigotimes_g B(\mathcal{A}) = P_\sigma \times B(\mathcal{A}) \bigotimes_g A(\mathcal{B}) \times {P_\sigma}^T \tag{2.22}$$

- Decomposição em produto tensorial clássico:

$$A \bigotimes_g B(\mathcal{A}) = \sum_{k=1}^{n_A} \ell_k(A) \otimes B(a_k) \tag{2.23}$$

3. REDES DE AUTÔMATOS ESTOCÁSTICOS

Neste capítulo será definido o formalismo denominado Redes de Autômatos Estocásticos (SAN - *Stochastic Automata Networks*) através da explicação de suas características e qualidades.

Redes de Autômatos Estocásticos é um formalismo proposto por Plauteau [13] para modelar um sistema em vários subsistemas pseudo-independentes. Um modelo SAN é um sistema complexo composto por uma coleção de subsistemas com comportamentos independentes através de eventos locais, e comportamentos interdependentes através de taxas funcionais e eventos sincronizantes. Cada subsistema do formalismo SAN é composto por estados e por transições entre esses estados.

Estas estruturas genéricas, definidas pelo formalismo SAN possibilitam ganhos em armazenamento e processamento para solucionar eficientemente sistemas complexos evitando assim, os prejuízos da explosão de espaços de estados como ocorriam em formalismos como as Cadeias de Markov. Destaca-se ainda que SAN e Cadeias de Markov possuem equivalência de representação [14].

Apresentamos a seguir, alguns conceitos que serão utilizados nos próximos capítulos deste trabalho. O Anexo A apresenta uma descrição sucinta de todas as notações que serão definidas neste capítulo.

Sejam

$\mathcal{A}$ o conjunto de autômatos que compreende N autômatos nomeados $\mathcal{A}^{(i)}$, onde $i \in [1..N]$ e $N = |\mathcal{A}|$;

$\mathcal{S}^{(i)}$ o conjunto de estados locais do autômato $\mathcal{A}^{(i)}$;

$x^{(i)}$ é um estado local do autômato $\mathcal{A}^{(i)}$, onde $x^{(i)} \in \mathcal{S}^{(i)}$;

$\hat{\mathcal{S}}$ é o espaço de estados produto do modelo, onde $\hat{\mathcal{S}} = \mathcal{S}^{(1)} \times \mathcal{S}^{(2)} \times ... \times \mathcal{S}^{(N)}$;

$\tilde{x}$ é o estado global do modelo o qual é uma composição dos estados locais do autômato, isto é, $\tilde{x} = (x^{(1)}, .., x^{(N)})$, onde $\tilde{x} \in \hat{\mathcal{S}}$;

$\mathcal{E}$ o conjunto de eventos que compreende E eventos, onde $\mathcal{E} = \{e_1, e_2, ..., e_E\}$ e $E = |\mathcal{E}|$;

O conceito de cada componente das redes de autômatos estocásticos serão detalhados a seguir.

3.1 Autômatos Estocásticos

Um autômato $\mathcal{A}^{(i)}$ é a representação de um sistema através de um modelo matemático com entradas e saídas discretas, onde o sistema pode estar em qualquer um dos estados possíveis [13]. O estado de cada autômato do modelo é denominado estado local, o estado local do autômato $\mathcal{A}^{(i)}$ é $x^{(i)}$. A combinação dos estados locais de toda a rede representam o estado global do sistema,

representado por $\tilde{x}$. A mudança do estado local de qualquer autômato gera a mudança no estado global do sistema.

Cada mudança de um estado para outro num autômato dá-se através de uma transição. A transição necessita ter ao menos um evento associado a ela para que ela possa ocorrer.

3.2 Eventos

A ocorrência de uma transição de um estado a outro dá-se através de eventos e, onde $e \in \mathcal{E}$. Quando ocorre um evento mudam-se os estados de todos os autômatos cujas transições possuem aquele evento. Conforme já foi dito anteriormente, a mudança no estado local de qualquer autômato no modelo muda também o estado global do modelo.

Os eventos são classificados em dois tipos, **local** ou **sincronizante**. Os eventos locais possuem a característica de somente alterar o estado de um único autômato sem que isso cause a mudança de estado de qualquer outro autômato da rede. Utiliza-se eventos locais para representar a independência entre os autômatos do modelo.

Um evento sincronizante pode alterar o estado local de dois ou mais autômatos simultaneamente, possibilitando a interação entre vários autômatos. É através do sincronismo entre o disparo do evento nas transições dos autômatos que é possível fazer iterações entre os autômatos do modelo.

Os eventos são definidos por seu tipo e um identificador. Os eventos possuem uma taxa de ocorrência e uma probabilidade, ambos podem ser associados diretamente a valores reais ou a funções. As taxas e probabilidades funcionais caracterizam-se pela dependência de seu valor com estados locais de um ou mais autômatos do modelo.

As probabilidades são associadas a um evento e a uma transição e, assim como as taxas de ocorrência também podem ser expressas por funções. Podemos definir formalmente os conceitos apresentados de acordo com as definições abaixo.

w é um conjunto de índices de autômatos (w é um subconjunto $[1..N]$);

$\tilde{x}^{(w)}$ composição dos estados locais $x^{(i)}$, onde $i \in w$;

$\hat{\mathcal{S}}^{(w)}$ é o espaço de estados produto do conjunto de estados locais do autômato $\mathcal{A}^{(i)}$ onde $i \in w$ e $\hat{\mathcal{S}}^{(w)} \subseteq \hat{\mathcal{S}}$;

e um evento, onde $e \in \mathcal{E}$;

ζ_e é o conjunto de índices dos autômatos onde o evento e aparece;

(e, τ_e) uma tupla de evento onde τ_e é um elemento funcional de $\hat{\mathcal{S}} \rightarrow \mathbb{R}^+$, o qual define a taxa de ocorrência do evento e;

(e, π_e) uma tupla de transição onde π_e é uma probabilidade funcional de $\hat{\mathcal{S}} \rightarrow \mathbb{R}^+$;

$\tau_e(\tilde{x})$ taxa de ocorrência do evento e avaliada para o estado global $\tilde{x}$;

$\mathcal{T}$ é um conjunto que contém todas as tuplas de transição (e, π_e), onde π_e é um elemento funcional definido de $\hat{\mathcal{S}}$, o qual representa a probabilidade de uma transição quando a ocorrência do evento e;

$\mathcal{T}^* = \mathcal{T} \bigcup \{\emptyset\}$;

$\mathcal{Q}^{(i)}$ é uma função de transição de $\mathcal{S}^{(i)} \times \mathcal{S}^{(i)} \rightarrow \mathcal{T}^*$ que contém as tuplas de transisão do autômato $\mathcal{A}^{(i)}$;

$\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ rótulo de transição do estado local $x^{(i)}$ para o estado $y^{(i)}$, contendo uma lista de tuplas de transição (e, π) em $\mathcal{T}$;

$\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$ rótulo de transição do estado global $\tilde{x}$ para $\tilde{y}$, contendo uma lista de tuplas de transição (e, π) em $\mathcal{T}$;

$\pi_e(x^{(i)}, y^{(i)})$ probabilidade da tupla de transição (e, π) associada ao rótulo de transição $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$;

$\pi_e(\tilde{x}, \tilde{y})$ probabilidade da transição global de $\tilde{x}$ para $\tilde{y}$ calculada pelo produtório de todas as probabilidades de rotação locais, ou seja, $\prod_{i \in \zeta_e, \exists (e,\pi) \in Q^{(i)}(x^{(i)}, y^{(i)})} \pi_e(x^{(i)}, y^{(i)})$;

$succ_e(\tilde{x})$ conjunto dos estados globais sucessores $\tilde{y}$ tais que o rótulo $\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$ possua uma tupla de transição com o identificador e e $\tau_e(\tilde{x}) \neq 0$ e $\pi_e(\tilde{x}, \tilde{y}) \neq 0$. O conjunto de estados sucessores de e em $\tilde{x}$ pode ser vazio, caso a transição não possa ser disparada em $\tilde{x}$ pelo evento e.

3.3 Funções

Como dito anteriormente, as taxas e probabilidades funcionais são utilizadas para representar uma probabilidade ou uma taxa de ocorrência levando em conta o estado global do sistema, aumentando assim a flexibilidade da modelagem através de SAN.

Seja

$\mathcal{F}$ o conjunto de funções do modelo;

$f(\hat{\mathcal{S}}^{(w)})$ uma função de $\hat{\mathcal{S}}^{(w)} \rightarrow \mathbb{R}^+$, onde $f \in \mathcal{F}$ e $w \subseteq [1..N]$;

f_e é uma função que corresponde ao elemento funciona τ_e associada a tupla do evento e, isto é, a tupla de evento (e, τ_e);

w_{f_e} é o conjunto dos índices dos autômatos os quais seu espaço de estados são o domínio da função $f_e \in \mathcal{F}$ do evento e;

w_{π_e} é o conjunto dos índices dos autômatos os quais seu espaço de estados são o domínio da probabilidade π_e do evento e;

$f_e(\tilde{x})$ é o resultado da avaliação da função f_e para o estado global $\tilde{x}$;

$\pi_e(\tilde{x})$ é o resultado da avaliação da probabilidade π_e para o estado global $\tilde{x}$;

Define-se uma **função como constante** quando para todo o seu domínio ela possui sempre o mesmo valor, isto é, sempre a mesma imagem.

Define-se também que uma **função é do tipo funcional** cuja ao menos um valor do seu domínio é diferente dos demais, ou seja, quando a avaliação da função der um valor diferente para um domínio e outro diferente em outro domínio.

Analogamente o mesmo princípio é aplicado para as probabilidades funcionais. Logo define-se uma **probabilidade como constante** quando para todo o seu domínio ela possui sempre o mesmo valor, isto é, sempre a mesma imagem.

Define-se também uma **probabilidade do tipo funcional**, cuja ao menos o valor para um domínio é diferente dos demais, ou seja, quando a avaliação da função der um valor direrente para um domínio e outro diferente em outro domínio.

Tanto uma função como uma probabilidade são consideradas nulas quando sua avaliação for sempre zero para todo o seu domínio.

3.4 Função de atingibilidade

A função de atingibilidade de um modelo SAN representa todos os estados globais atingíveis do modelo [14]. Na Figura 3.2 é apresentada a cadeia de Markov correspondente ao modelo SAN da Figura 3.1, pode-se ver que todos os estados são atingíveis, portanto a função de atingibilidade é 1.

3.5 Exemplo de um modelo SAN

Para demonstrar os conceitos apresentados no decorrer deste capítulo será apresentado agora um exemplo de um modelo SAN.

Para o nosso exemplo, utilizaremos a rede SAN representada na na Figura 3.1, o modelo possui 2 autômatos, $\mathcal{A}^{(1)}$ e $\mathcal{A}^{(2)}$. O autômato $\mathcal{A}^{(1)}$ possui três estados, que são os estados $0^{(1)}$, $1^{(1)}$ e $2^{(1)}$. O autômato $\mathcal{A}^{(2)}$ possui dois estados, o estado ($0^{(2)}$ e o estado $1^{(2)}$).

A rede SAN do nosso exemplo possui 5 eventos, 4 destes eventos ocorrem no autômato $\mathcal{A}^{(1)}$ ($e1$, $e2$, $e3$ e $e4$) com taxas constantes e 2 ocorrem no autômato $\mathcal{A}^{(2)}$ ($e4$, $e5$), onde o evento $e5$ possui taxa funcional. Pode-se notar também que o evento $e4$ é um evento sincronizante e ocorre nos 2 autômatos.

Para demonstrar a equivalência de representação entre SAN e Cadeias de Markov, na Figura 3.2 é apresentada a Cadeia de Markov equivalente ao rede SAN do nosso exemplo.

Note que na Figura 3.1, o evento $e4$ é um evento sincronizante e o evento $e5$ possui a função f associada a sua taxa de ocorrência conforme descrito abaixo:

$f = \lambda 1$ se o autômato $\mathcal{A}^{(1)}$ está no estado $0^{(1)}$

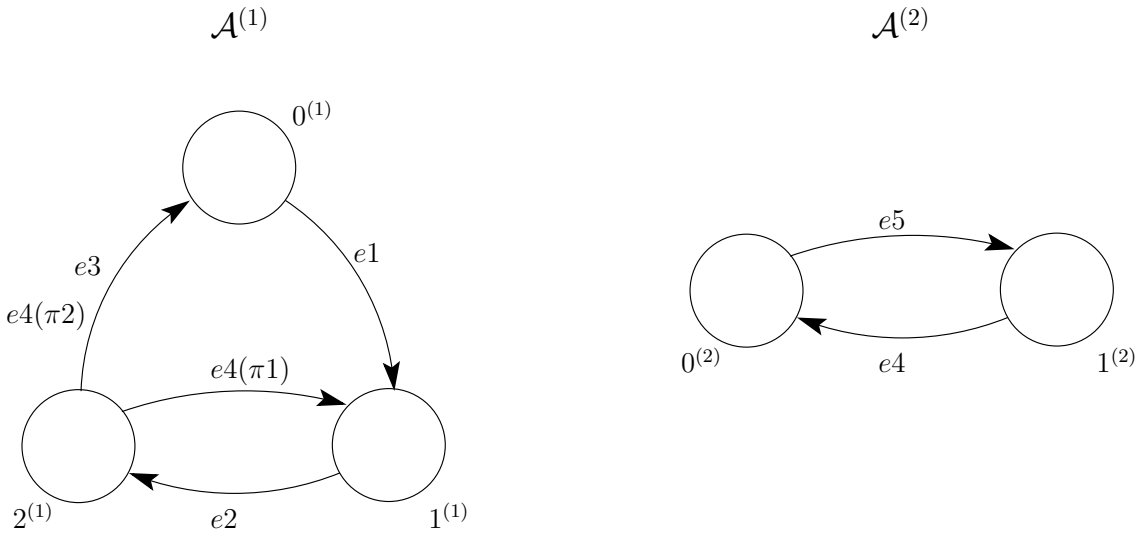

Figura 3.1: Exemplo de um modelo SAN

Tipo	Evento	Taxa
local	$e1$	$t1$
local	$e2$	$t2$
local	$e3$	$t3$
sincronizante	$e4$	$t4$
local	$e5$	f

Tabela 3.1: Taxa dos eventos da Figura 3.1

$f = 0$ se o autômato $\mathcal{A}^{(1)}$ está no estado $1^{(1)}$
$f = \lambda 2$ se o autômato $\mathcal{A}^{(1)}$ está no estado $2^{(1)}$

3.6 Descrição textual de um modelo SAN

Para avaliar um modelo SAN na ferramenta PEPS [3] utiliza-se uma descrição textual e estruturada.

A descrição correspondente do modelo SAN apresentada na Figura 3.1 é mostrada a seguir, lembrando que: os valores das taxas para os eventos $e1$, $e2$, $e3$, e $e4$ viraram valores reais 1, 2, 3 e 4 respectivamente; os valores de $\lambda 1$ e $\lambda 2$ viraram respectivamente 10 e 20 e os valores $\pi 1$ e $\pi 2$ viraram 0.7 e 0.3.

```
identifiers
txe1=1;
txe2=2;
txe3=3;
txe4=4;
txe5= ((st A1 == est0)*10)+((st A1 == est2)*20);
events
```

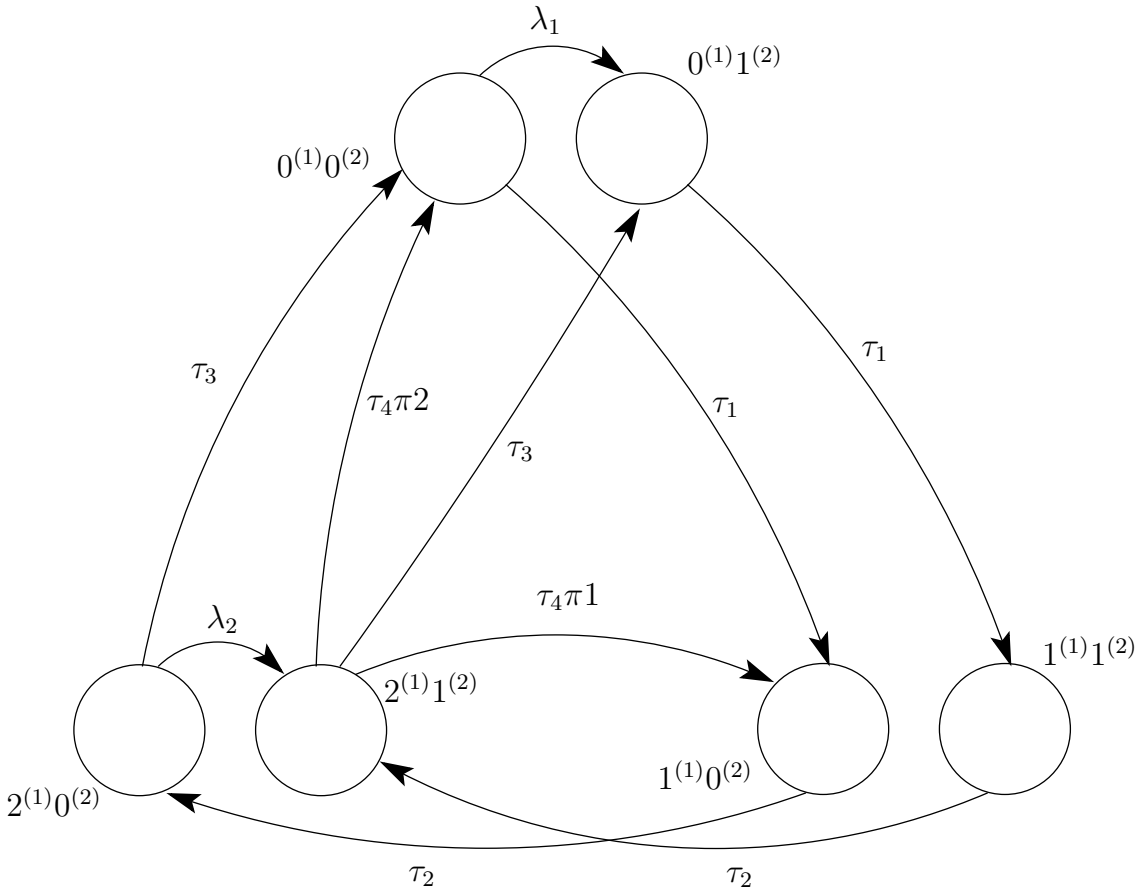

Figura 3.2: Cadeia de Markov equivalente ao modelo SAN da Figura 3.1

```
loc e1(txe1);
loc e2(txe2);
loc e3(txe3);
syn e4(txe4);
loc e5(txe5);
reachability = 1;
network exemplo(continuous)
aut A1 stt est0 to(est1) e1
       stt est1 to(est2) e2
       stt est2 to(est1) e4(0.7)
                to(est0) e4(0.3) e3
aut A2 stt est0 to(est1) e5
       stt est1 to(est0) e4
results
a2_estar_em_0 = st A2 == est0;
```

3.7 Descritores Markovianos

O formalismo de redes de autômatos estocásticos utiliza álgebra tensorial clássica e generalizada para simplificar a representação da matriz de transição (gerador infinitesimal) da cadeia de markov equivalente. Esta simplificação gera uma matriz de transição compacta, chamada de Descritor Markoviano.

O Descritor Markoviano é a representação tensorial das transições entre os estados do autômato. Basicamente o Descritor Markoviano é composto pela soma tensorial dos eventos locais somado ao produto tensorial das matrizes que representam a ocorrência de eventos sincronizantes para cada autômato.

A equação 3.1 mostra a expressão das transições expressas em formato tensorial [10].

$$Q = \bigoplus_{i=1}^{N} Q_l^{(i)} + \sum_{e \in E} \left(\bigotimes_{i=1}^{N} Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N} Q_{e^-}^{(i)} \right) \tag{3.1}$$

Onde:

- N é o número total de autômatos;
- E é o total de eventos sincronizantes;
- Q_l são as matrizes que representam a ocorrência e o ajuste da diagonal de eventos locais;
- Q_{e^+} são as matrizes que representam a ocorrência de eventos sincronizantes;
- Q_{e^-} são as matrizes que representam o ajuste diagonal dos eventos sincronizantes;

Primeiramente, na equação 3.2 é mostrado o primeiro termo da equação 3.1 representa a ocorrência e o ajuste da diagonal dos eventos locias.

$$\bigoplus_{i=1}^{N} Q_l^{(i)} \tag{3.2}$$

O próximo passo para a geração do Descritor Markoviano são as matrizes que representam os a ocorrência de eventos sincronizantes e das matrizes que representam o ajuste diagonal desses eventos, neste ponto iremos detalhar o segundo termo da equação 3.1, representado na equação 3.3.

$$\sum_{e \in E} \left(\bigotimes_{i=1}^{N} Q_{e^+}^{(i)} + \bigotimes_{i=1}^{N} Q_{e^-}^{(i)} \right) \tag{3.3}$$

A seguir, serão apresentados 2 exemplos de modelos SAN e o seus respectivos descritores markovianos, no primeiro exemplo será utilizado um modelo SAN simples com tensores pequenos (de dimensão 2) para demonstrar claramente a representação do descritor. Este mesmo exemplo será

usado no decorrer deste trabalho para demonstrar a conversão de modelos SAN descritos utilizando taxas funcionais para modelos SAN que somente utilizam eventos com taxas e probabilidades constantes.

3.7.1 Exemplo elementar

O primeiro exemplo de descritor markoviano será feito a partir da rede SAN da Figura 3.3. A rede possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ com 2 estados ($0^{(1)}$ e $1^{(1)}$), o autômato $\mathcal{A}^{(2)}$ também com 2 estados ($0^{(2)}$ e $1^{(2)}$) e o autômato $\mathcal{A}^{(3)}$ com os 2 estados ($0^{(3)}$ e $1^{(3)}$). Na Tabela 3.2 pode-se observar os eventos vinculados as transições do SAN da Figura 3.3.

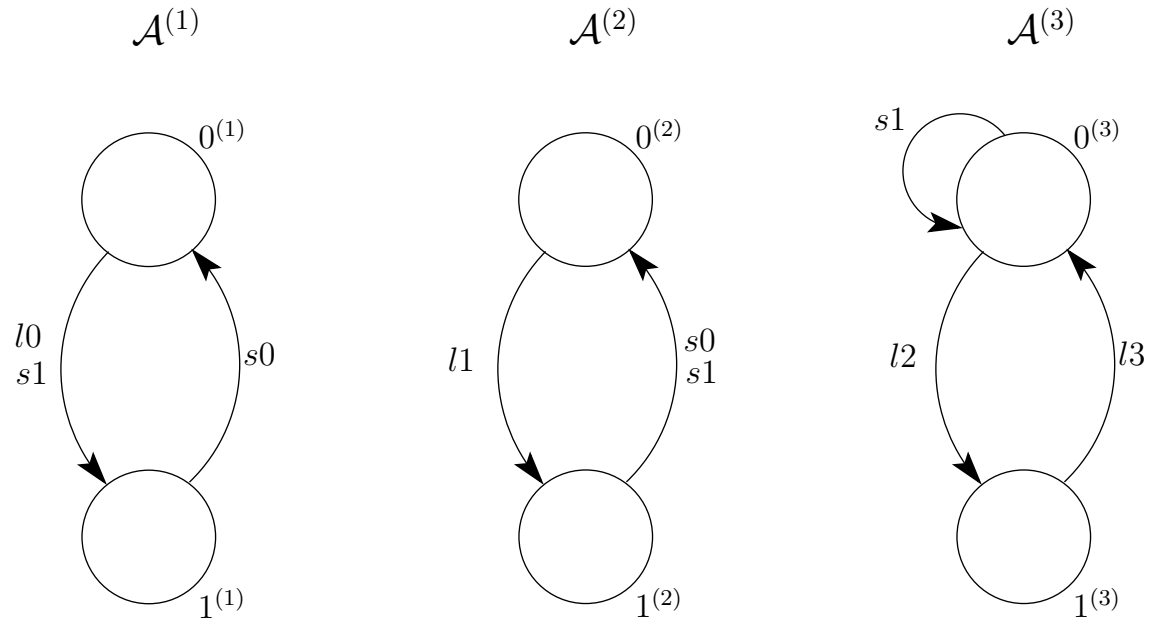

Figura 3.3: SAN utilizado para o exemplo.

Tipo	Evento	Taxa
loc	$l0$	$r0$
loc	$l1$	$r1$
loc	$l2$	$r2$
loc	$l3$	f
syn	$s0$	$r4$
syn	$s1$	$r5$

Tabela 3.2: Taxas dos eventos da SAN representada na Figura 3.3

O evento local l3 descreve a função f é representado por:

$$f = (stA^{(1)} == 0^{(1)}) \times r3 \tag{3.4}$$

Considerando a SAN acima descrita, para gerar o Descritor Markoviano será necessário um termo com as somas tensoriais (relativo a parte local) e quatro termos com os produtos tensorials (relativos

aos eventos sincronizantes). A seguir serão descritos estes termos:

Parte local dos autômatos:

$$Q_l = Q_l^{A^{(1)}} \oplus Q_l^{A^{(2)}} \oplus Q_l^{A^{(3)}} = \begin{pmatrix} -r0 & r0 \\ 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} -r1 & r1 \\ 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} -r2 & r2 \\ r3 & -r3 \end{pmatrix}$$

Parte sincronizante positiva relativa ao evento $s0$:

$$Q_{s0+} = Q_{s0+}^{A^{(1)}} \otimes Q_{s0+}^{A^{(2)}} \otimes Q_{s0+}^{A^{(3)}} = \begin{pmatrix} 0 & 0 \\ r4 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Parte sincronizante negativa relativa ao ajuste diagonal do evento $s0$:

$$Q_{s0-} = Q_{s0-}^{A^{(1)}} \otimes Q_{s0-}^{A^{(2)}} \otimes Q_{s0-}^{A^{(3)}} = \begin{pmatrix} 0 & 0 \\ 0 & -r4 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Parte sincronizante positiva relativa ao evento $s1$:

$$Q_{s1+} = Q_{s1+}^{A^{(1)}} \otimes Q_{s1+}^{A^{(2)}} \otimes Q_{s1+}^{A^{(3)}} = \begin{pmatrix} 0 & r5 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Parte sincronizante negativa relativa ao ajuste diagonal do evento $s1$:

$$Q_{s1-} = Q_{s1-}^{A^{(1)}} \otimes Q_{s1-}^{A^{(2)}} \otimes Q_{s1-}^{A^{(3)}} = \begin{pmatrix} -r5 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

O gerador infinitesimal da cadeia de markov equivalente a esta SAN é:

$$Q = Q_l + (Q_{s0+} + Q_{s0-}) + (Q_{s1+} + Q_{s1-})$$

3.7.2 Exemplo do modelo *Random Waypoint*

Neste exemplo, será demonstrado como gerar um Descritor Markoviano a partir da SAN definida na Figura 3.4. O modelo SAN em questão define o cálculo da mobilidade de nós móveis em redes *Wireless* (de forma *adhoc*) utilizando o algoritmo *Random Waypoint* [9].

Tipo	Evento	Taxa
loc	$moveW$	f_W
loc	$moveE$	f_E
loc	$moveN$	f_N
loc	$moveS$	f_S
syn	$chooseD(\pi)$	f_D

Tabela 3.3: Taxas dos eventos da SAN representada na Figura 3.4

Abaixo, podemos observar as funções f_W, f_E, f_N, f_S e f_D que são declaradas nas taxas da Tabela 3.3.

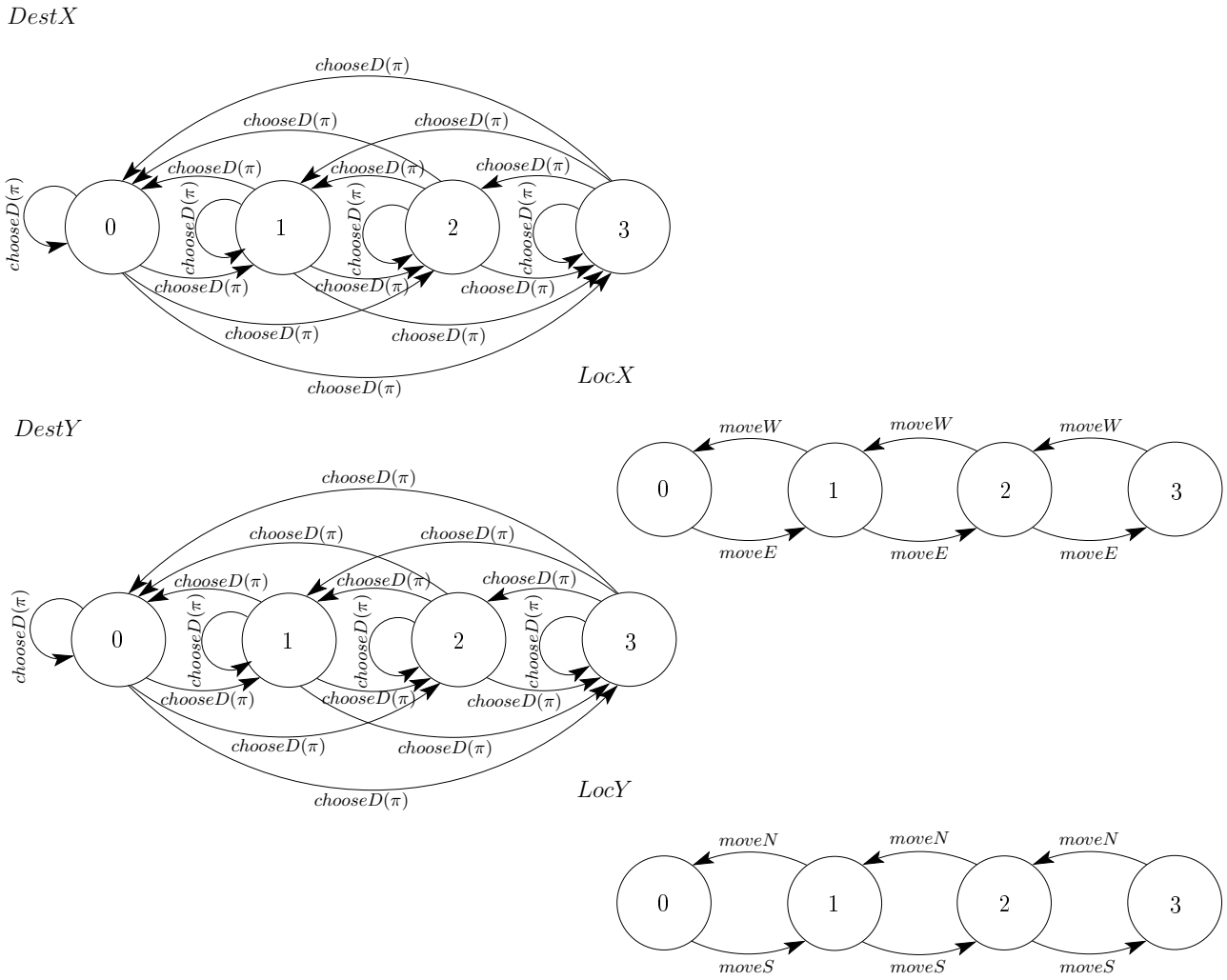

Figura 3.4: Modelo SAN para o padrão Random Waypoint 2D.

$$f_W = rtSpeed * ((stDestX) < (stLocX)) \tag{3.5}$$

$$f_E = rtSpeed * ((stDestX) > (stLocX)) \tag{3.6}$$

$$f_N = rtSpeed * ((stDestY) < (stLocY)) \tag{3.7}$$

$$f_S = rtSpeed * ((stDestY) < (stLocY)) \tag{3.8}$$

$$f_D = ((1/Pause) * (((stDestX) == (stLocX)) \&\& ((stDestY) == (stLocY)))) \tag{3.9}$$

Considerando o modelo SAN para Random Waypoint, para gerar o Descritor Markoviano será necessário um termo com as somas tensoriais (relativo à parte local) e um termo com os produtos tensoriais (relativos ao evento sincronizante $chooseD$). A seguir serão descritos estes termos:

O termo do Descritor Markoviano que representa a parte local dos autômatos dá-se como:

$$Q_l = Q_l^{DestX} \oplus_g Q_l^{LocX} \oplus_g Q_l^{DestY} \oplus_g Q_l^{LocY}$$

Onde:

$$Q_l^{DestX} = \begin{pmatrix} -(f_D+f_D+f_D+f_D) & f_D & f_D & f_D & f_D \\ f_D & -(f_D+f_D+f_D+f_D) & f_D & f_D & f_D \\ f_D & f_D & -(f_D+f_D+f_D+f_D) & f_D & f_D \\ f_D & f_D & f_D & -(f_D+f_D+f_D+f_D) & f_D \\ f_D & f_D & f_D & f_D & -(f_D+f_D+f_D+f_D) \end{pmatrix}$$

$$Q_l^{LocX} = \begin{pmatrix} -f_E & f_E & 0 & 0 & 0 \\ f_W & -(f_W+f_E) & f_E & 0 & 0 \\ 0 & f_W & -(f_W+f_E) & f_E & 0 \\ 0 & 0 & f_W & -(f_W+f_E) & f_E \\ 0 & 0 & 0 & f_W & -f_W \end{pmatrix}$$

$$Q_l^{DestY} = \begin{pmatrix} -(f_D+f_D+f_D+f_D) & f_D & f_D & f_D & f_D \\ f_D & -(f_D+f_D+f_D+f_D) & f_D & f_D & f_D \\ f_D & f_D & -(f_D+f_D+f_D+f_D) & f_D & f_D \\ f_D & f_D & f_D & -(f_D+f_D+f_D+f_D) & f_D \\ f_D & f_D & f_D & f_D & -(f_D+f_D+f_D+f_D) \end{pmatrix}$$

$$Q_l^{LocY} = \begin{pmatrix} -f_S & f_S & 0 & 0 & 0 \\ f_N & -(f_N+f_S) & f_S & 0 & 0 \\ 0 & f_N & -(f_N+f_S) & f_S & 0 \\ 0 & 0 & f_N & -(f_N+f_S) & f_S \\ 0 & 0 & 0 & f_N & -f_N \end{pmatrix}$$

Abaixo será demonstrado como gerar o termo com a parte sincronizante do evento $chooseD$:

$$Q_{chooseD+} = Q^{DestX}_{chooseD+} \otimes_g Q^{LocX}_{chooseD+} \otimes_g Q^{DestY}_{chooseD+} \otimes_g Q^{LocY}_{chooseD+}$$

Onde:

$$Q^{DestX}_{chooseD+} = \begin{pmatrix} 0 & f_D & f_D & f_D & f_D \\ f_D & 0 & f_D & f_D & f_D \\ f_D & f_D & 0 & f_D & f_D \\ f_D & f_D & f_D & 0 & f_D \\ f_D & f_D & f_D & f_D & 0 \end{pmatrix}$$

$$Q^{LocX}_{chooseD+} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Q^{DestY}_{chooseD+} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$Q^{LocY}_{chooseD+} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Parte sincronizante negativa relativa ao ajuste diagonal do evento $chooseD$:

$$Q_{chooseD-} = Q^{DestX}_{chooseD-} \otimes_g Q^{LocX}_{chooseD-} \otimes_g Q^{DestY}_{chooseD-} \otimes_g Q^{LocY}_{chooseD-}$$

Onde:

$$Q^{DestX}_{chooseD+} = \begin{pmatrix} -(f_D+f_D+f_D+f_D) & 0 & 0 & 0 & 0 \\ 0 & -(f_D+f_D+f_D+f_D) & 0 & 0 & 0 \\ 0 & 0 & -(f_D+f_D+f_D+f_D) & 0 & 0 \\ 0 & 0 & 0 & -(f_D+f_D+f_D+f_D) & 0 \\ 0 & 0 & 0 & 0 & -(f_D+f_D+f_D+f_D) \end{pmatrix}$$

$$Q^{LocX}_{chooseD+} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Q^{DestY}_{chooseD+} = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & 0 & -4 \end{pmatrix}$$

$$Q^{LocY}_{chooseD+} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4. DEFINIÇÃO DO ALGORITMO PARA O CONVERSOR

Neste capítulo, será demonstrado os passos necessários para se converter modelos SAN definidos utilizando primitivas funcionais (descritos em GTA) para modelos SAN definidos com eventos que utilizam somente taxas e probabilidades constantes (descritos em CTA).

De acordo com Brenner *et al.* em [5] qualquer modelo SAN que utiliza taxas representadas por funções (GTA) possui pelo menos um modelo equivalente utilizando somente operadores CTA, ou seja, qualquer modelo SAN com taxas ou probabilidades funcionais pode ser representado por um modelo SAN com taxas ou probabilidades constantes.

Embora modelos SAN descritos em GTA possuam uma representação compacta e mais eficiente em comparação a modelos descritos em CTA, dependendo do modelo de avaliação de função sua solução pode ser tornar custosa. Para certos modelos pode ser mais vantajosa a utilização de eventos sincronizantes em relação a utilização de funções [5].

A chave para conversão de um modelo SAN para utilizar ou não funções é a substituição dos eventos definidos através de funções por eventos sincronizantes com taxas constantes. Cada evento do modelo com taxa funcional, irá se transformar em um evento sincronizante cujos estados e autômatos utilizados na avaliação possuirão também o evento sincronizante. Isto nem sempre é trivial pois depende da composição da função que pode ter diferentes operadores lógicos, entre outros.

Para cada evento definido através de uma função, deve-se verificar todos os estados globais do subespaço de autômatos do domínio da função, para cada estado global deve-se avaliar a função. Caso a função não seja nula, é criado um novo evento sincronizante na transição do original e também através de novas transições em *loop* criadas para todos os estados dos autômatos que compõe o subespaço de domínio da função cuja avaliação não é nula. A taxa do evento sincronizante ficará com o valor avaliado da função.

Vale ressaltar que o algoritmo é conceitual, logo ele pode ser aplicado tanto sobre um modelo SAN quanto sobre seu descritor. Inicialmente, para demonstrar e validar o funcionamento do algoritmo será trabalhado no nível de modelo.

4.1 Algoritmo de conversão passo a passo

Conforme já descrito anteriormente, para converter um modelo SAN de GTA para CTA, deve-se alterar os eventos com taxas ou probabilidades funcionais de todas as transições entre os estados dos autômatos do modelo por eventos sincronizantes. No decorrer desta seção será descrito como converter um modelo SAN definido utilizando primitivas funcionais para um modelo SAN que não utiliza primitivas funcionais, ou seja, cujo o descritor possa ser representado por álgebra tensorial clássica.

Cabe lembrar que no Anexo A apresenta-se o resumo da notação definida no Capítulo 3. Esta

notação será necessária para a compreensão do Algoritmo 4.1.

Para exemplificar a conversão de uma SAN para substituir os eventos com taxas funcionais por eventos sincronizantes, utilizaremos o Algoritmo 4.1.

O algoritmo inicia percorrendo todos os eventos e, onde $e \in \mathcal{E}$, conforme linha 1. Na linha 2, para cada evento, é verificado se a função f correspondente a um elemento funcional associado à taxa de ocorrência do evento e **ou** se existe alguma probabilidade π definida para a transição associada ao evento e é do tipo funcional.

Garantindo que estamos trabalhando com um evento definido como sendo do tipo funcional, percorremos todos os estados globais $\tilde{x}$ do universo composto pela união de 3 conjuntos de autômatos $(\zeta_e \cup w_{f_e} \cup w_{\pi_e})$, sendo:

ζ_e o conjunto composto pelos autômatos onde o evento e aparece;

w_{f_e} o conjunto composto pelos autômatos que são domínimo da função f_e;

w_{π_e} o conjunto composto pelos autômatos que são domínio da função da probabilidade π_e.

Na linha 3, pode-se observar o laço através de $\tilde{x} \in \hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$.

Na linha 4, o conjunto $\mathcal{E}'$ é definido como vazio para que possa ser reutilizado no processo de reaproveitamento de eventos. Este processo será detalhado a seguir.

Na linha 5, percorre-se os estados globais $\tilde{y}$ sucessores de $\tilde{x}$. Note que serão considerados somente os estados $\tilde{y}$ cuja a avaliação da taxa do evento e das probabilidades são diferente de zero.

Para cada tupla de transição entre os estados dos autômatos que possuem o evento e associado a taxa que representa este evento e a probabilidade do evento e para a tupla em questão (de $\tilde{x}$ para $\tilde{y}$) devem ser avaliados através da sentença $f_e(\tilde{x}) \times \pi_e(\tilde{x}, \tilde{y})$.

O resultado da multiplicação entre a avaliação da função $f_e(\tilde{x})$ e da probabilidade $\pi_e(\tilde{x}, \tilde{y})$ será armazenado em $\tau_{e'}$, caso a avaliação resulte em um valor igual a outro já avaliado para a mesma tupla de transição entre $\tilde{x}$ e $\tilde{y}$, deve-se reutilizar o mesmo evento já criado. Podemos encontrar o trecho que representa este controle nas linhas 7, 16, 26 e 28 do algoritmo.

Cada estado dos autômatos que são domínios da função f_e e da probabilidade π_e devem receber uma tupla de transição para ele mesmo vinculada ao evento e', linhas 13 e 22.

Todas as tuplas de transição onde o evento e aparece devem ser substituidas para utilizar o novo evento sincronizante e', conforme as linhas 9, 10, 18 e 19.

Desta forma, todos os elementos do modelo SAN que são resolvidos utilizando funções serão substituídos por eventos sincronizantes e'.

Nas próximas seções, serão apresentados um modelos SAN que utilizam taxas funcionais e a conversão destes modelos para modelos equivalentes que somente utilizam taxas e probabilidades constantes através de eventos sincronizantes.

Algoritmo 4.1: Conversão de modelos SAN de GTA para CTA

Require: [Pré Condição] Uma rede de autômatos estocásticos bem definida utilizando eventos funcionais

Ensure: Uma rede de autômatos estocásticos bem definida equivalente a rede de entrada sem utilizar funções

1: **for all** $e \in \mathcal{E}$ **do**
2: **if** f_e é do tipo funcional **or** existe algum π_e do tipo funcional **then**
3: **for all** $\tilde{x} \in \hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$ **do**
4: $\mathcal{E}' \leftarrow \emptyset$
5: **for all** $\tilde{y} \in succ_e(\tilde{x})$ **do**
6: $\tau_{e'} = f_e(\tilde{x}) \times \pi_e(\tilde{x}, \tilde{y})$
7: **if** $\exists\ e'' \in \mathcal{E}'$ **such that** $\tau_{e''} = \tau_{e'}$ **then**
8: **for all** $i \in \zeta_e$ **do**
9: Remove $\pi_e(x^{(i)}, y^{(i)})$ de $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$
10: Adiciona em $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ a tupla $\pi_{e''}(x^{(i)}, y^{(i)})$
11: **end for**
12: **for all** $j \in (w_{f_e} \cup w_{\pi_e})$ **do**
13: Adiciona em $\mathcal{Q}^{(j)}(x^{(j)}, x^{(j)})$ a tupla $\pi_{e''}(x^{(j)}, x^{(j)})$
14: **end for**
15: **else**
16: Adiciona-se e' no conjunto $\mathcal{E}'$
17: **for all** $i \in \zeta_e$ **do**
18: Remove $\pi_e(x^{(i)}, y^{(i)})$ de $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$
19: Adiciona em $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ a tupla $\pi_{e'}(x^{(i)}, y^{(i)})$
20: **end for**
21: **for all** $j \in (w_{f_e} \cup w_{\pi_e})$ **do**
22: Adiciona em $\mathcal{Q}^{(j)}(x^{(j)}, x^{(j)})$ a tupla $\pi_{e'}(x^{(j)}, x^{(j)})$
23: **end for**
24: **end if**
25: **end for**
26: $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}'$
27: **end for**
28: Remove e do conjunto $\mathcal{E}$
29: **end if**
30: **end for**

4.2 Validação do algoritmo

Para validar o algoritmo que converte modelos SAN descrito utilizando álgebra tensorial generalizada (GTA) em modelos SAN descritos utilizando álgebra tensorial clássica (CTA), iremos utilizar uma gama de modelos que abrangem todas as combinações possíveis das características que possuem relação com a conversão e com o algoritmo. Abaixo segue a relação dos tipos de modelos diferente que serão utilizados para validar o algoritmo:

4.2.1 Rede SAN com evento local com taxa funcional sem especificação de probabilidades;

4.2.2 Rede SAN com evento local com taxa funcional e probabilidades constantes;

4.2.3 Rede SAN com evento local com taxa constante e probabilidades funcionais;

4.2.4 Rede SAN com evento local com taxa funcional e probabilidades funcionais;

4.2.5 Rede SAN com evento sincronizante com taxa funcional sem especificação de probabilidades;

4.2.6 Rede SAN com evento sincronizante com taxa funcional e probabilidades constantes;

4.2.7 Rede SAN com evento sincronizante com taxa constante e probabilidades funcionais;

4.2.8 Rede SAN com evento sincronizante com taxa funcional e probabilidades funcionais.

Através da combinação de característica dos modelos, como por exemplo eventos locais e sincronizantes e taxas e probabilidades funcionais ou constantes, pretende-se validar que qualquer modelo SAN definido utilizando primitivas GTA possa ser convertido utilizando o Algoritmo 4.1 **definido** neste trabalho.

4.2.1 Rede SAN com evento local com taxa funcional sem especificação de probabilidades

Nesta subseção será utilizado um modelo SAN simples para validar o funcionamento do algoritmo. O modelo possui somente eventos do tipo local, um deles com taxa funcional e sem definição explícita de probabilidades, ou seja, as probabilidades de todas as tansições é 1.

O modelo SAN é apresentado na Figura 4.1. Pode-se notar que a SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ possui 3 estados ($0^{(1)}$, $1^{(1)}$ e $2^{(1)}$), o autômato $\mathcal{A}^{(2)}$ possui 2 estados ($0^{(2)}$ e $1^{(2)}$) enquanto o autômato $\mathcal{A}^{(3)}$ possui 3 estados ($0^{(3)}$, $1^{(3)}$ e $2^{(3)}$).

Na Tabela 4.1 são apresentados os eventos do modelo da Figura 4.1. Nota-se que o autômato $\mathcal{A}^{(1)}$ possui um evento local $e1$ representado através de uma taxa funcional. A função que representa $e1$ avalia o comportamento tanto do autômato $\mathcal{A}^{(2)}$ quanto do autômato $\mathcal{A}^{(3)}$.

O evento local $e1$ utiliza a taxa funcional f que é representado por:

$$f = \Big[(stA^{(2)} == 0^{(2)}) \times 3\Big] + \Big[((stA^{(3)} == 0^{(3)}) \times 4) + ((stA^{(3)} == 2^{(3)}) \times 5)\Big] \qquad (4.1)$$

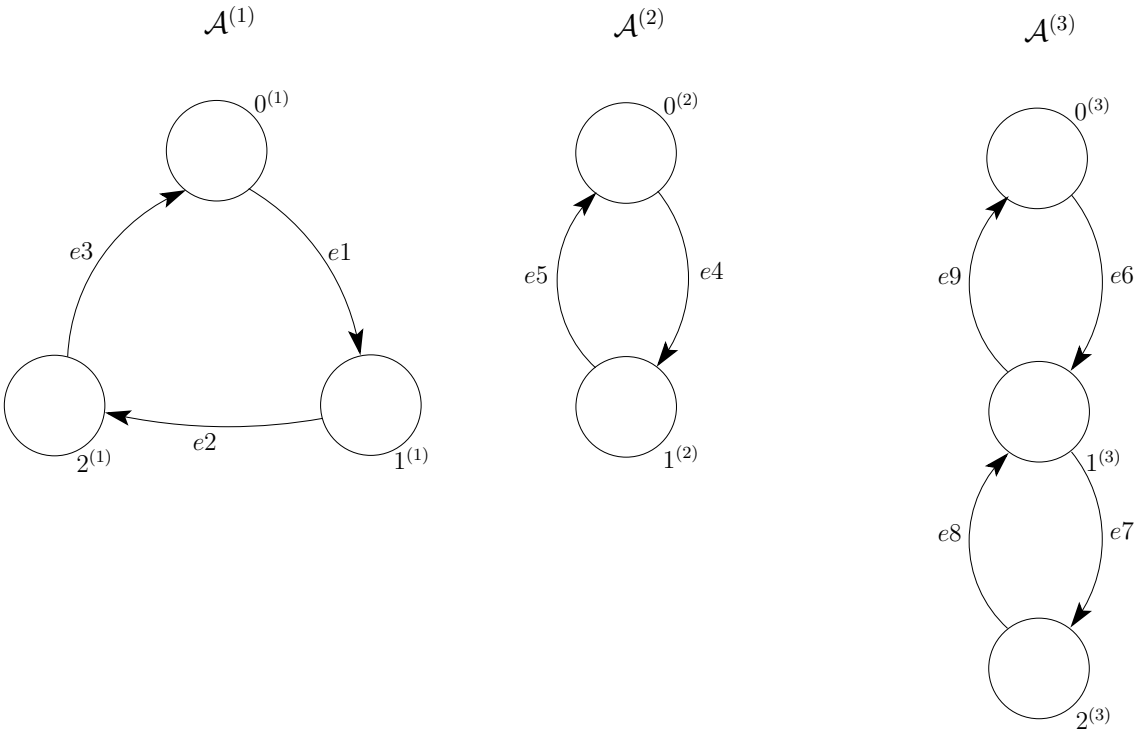

Figura 4.1: Exemplo de SAN com evento local com taxa funcional.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e1$	f	loc	$e6$	1
loc	$e2$	1	loc	$e7$	1
loc	$e3$	1	loc	$e8$	1
loc	$e4$	1	loc	$e9$	1
loc	$e5$	1			

Tabela 4.1: Taxas dos eventos da SAN representada na Figura 4.1

Para converter o modelo SAN da Figura 4.1 para não utilizar taxas funcionais, o único evento que necessita ser substituído por eventos sincronizantes é o evento $e1$, por possuir associado a ele uma taxa do tipo funcional.

Primeiramente, para gerar os eventos sincronizantes é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$. Conforme linha 3 do algoritmo.

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(2)}, \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \emptyset$

Neste ponto, percorre-se todo o espaço de estados dos autômatos definidos anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$, conforme a linha 5 do algoritmo.

A cada estado global percorrido a função é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

Por exemplo, avaliando a função para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(1^{(1)},0^{(2)},0^{(3)})$. O resultado da parte $\left[(stA^{(2)} == 0^{(2)}) \times 3\right]$ resulta 3 e $\left[((stA^{(3)} == 0^{(3)}) \times 4) + ((stA^{(3)} == 2^{(3)}) \times 5)\right]$ resulta 4. Finalmente, $3+4=7$. Neste ponto será criado um evento novo chamado $e10$ cuja taxa constante vale 7.

Na Tabela 4.2 podemos ver os resultados da função f do evento $e1$ avaliadas para as transições entre os estados globais $\tilde{x}$ e $\tilde{y}$.

$\tilde{x}$	$\tilde{y}$	Valor avaliado	Evento novo
$(0^{(1)},0^{(2)},0^{(3)})$	$(1^{(1)},0^{(2)},0^{(3)})$	7	$e10$
$(0^{(1)},0^{(2)},1^{(3)})$	$(1^{(1)},0^{(2)},1^{(3)})$	3	$e11$
$(0^{(1)},0^{(2)},2^{(3)})$	$(1^{(1)},0^{(2)},2^{(3)})$	8	$e12$
$(0^{(1)},1^{(2)},0^{(3)})$	$(1^{(1)},1^{(2)},0^{(3)})$	4	$e13$
$(0^{(1)},1^{(2)},1^{(3)})$	$(1^{(1)},1^{(2)},1^{(3)})$	0	
$(0^{(1)},1^{(2)},2^{(3)})$	$(1^{(1)},1^{(2)},2^{(3)})$	5	$e14$

Tabela 4.2: Resultado da avaliação da função f para o espaço de estados definido nela.

Cada estado do espaço de estados avaliado pela função deve receber uma transição para ele mesmo com o novo evento criado. Por exemplo para o espaço de estados ($0^{(2)}$ e $0^{(3)}$), o estado $0^{(2)}$ e o estado $0^{(3)}$ irão receber uma transição para eles mesmos com o evento $e10$. Assim ocorre para todo o espaço de estados. A transição do estado onde o evento $e1$ estava originalmente também recebe o evento $e10$.

Na Figura 4.2 é representado o modelo SAN da Figura 4.1 convertido para não utilizar taxas funcionais.

Tando o modelo SAN original (Figura 4.1) quanto o modelo convertido (Figura 4.2) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e2$	1	loc	$e9$	1
loc	$e3$	1	syn	$e10$	7
loc	$e4$	1	syn	$e11$	3
loc	$e5$	1	syn	$e12$	8
loc	$e6$	1	syn	$e13$	4
loc	$e7$	1	syn	$e14$	5
loc	$e8$	1			

Tabela 4.3: Taxas dos eventos da SAN representada na Figura 4.2

Na Tabela 4.4 pode-se observar que para todos os estados globais da rede SAN as probabilidades do modelo GTA e do modelo CTA são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

No Anexo B demonstra-se a definição dos modelos.

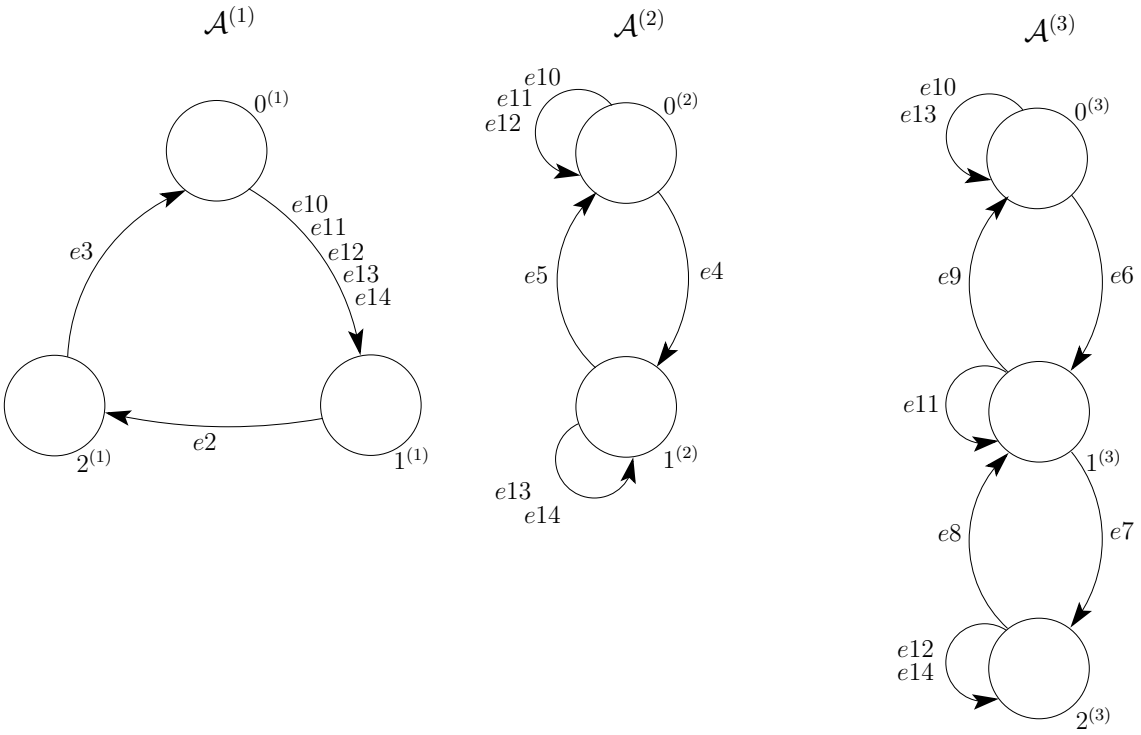

Figura 4.2: Exemplo de SAN com evento local com taxa funcional convertido para não mais usar funções.

Estado global	Modelo GTA (Figura 4.1)	Modelo CTA (Figura 4.2)	Estado global	Modelo GTA (Figura 4.1)	Modelo CTA (Figura 4.2)
$0^{(1)}0^{(2)}0^{(3)}$	0.0132532368	0.0132532368	$1^{(1)}1^{(2)}0^{(3)}$	0.0214757795	0.0214757795
$0^{(1)}0^{(2)}1^{(3)}$	0.0234590781	0.0234590781	$1^{(1)}1^{(2)}1^{(3)}$	0.0437318825	0.0437318825
$0^{(1)}0^{(2)}2^{(3)}$	0.0116752407	0.0116752407	$1^{(1)}1^{(2)}2^{(3)}$	0.0183177186	0.0183177186
$0^{(1)}1^{(2)}0^{(3)}$	0.0214757795	0.0214757795	$2^{(1)}0^{(2)}0^{(3)}$	0.0132532368	0.0132532368
$0^{(1)}1^{(2)}1^{(3)}$	0.0437318825	0.0437318825	$2^{(1)}0^{(2)}1^{(3)}$	0.0234590781	0.0234590781
$0^{(1)}1^{(2)}2^{(3)}$	0.0183177186	0.0183177186	$2^{(1)}0^{(2)}2^{(3)}$	0.0116752407	0.0116752407
$1^{(1)}0^{(2)}0^{(3)}$	0.0132532368	0.0132532368	$2^{(1)}1^{(2)}0^{(3)}$	0.0214757795	0.0214757795
$1^{(1)}0^{(2)}1^{(3)}$	0.0234590781	0.0234590781	$2^{(1)}1^{(2)}1^{(3)}$	0.0437318825	0.0437318825
$1^{(1)}0^{(2)}2^{(3)}$	0.0116752407	0.0116752407	$2^{(1)}1^{(2)}2^{(3)}$	0.0183177186	0.0183177186

Tabela 4.4: Resultados dos modelos SAN das Figuras 4.1 e 4.2

4.2.2 Rede SAN com evento local com taxa funcional e probabilidades constantes

Nesta subseção será utilizado um modelo SAN definido na Figura 4.3. O modelo possui somente eventos do tipo local, um deles com taxa funcional e que especifica probabilidades constantes. No caso deste exemplo, as probabilidades definem a transisão do estado $0^{(2)}$ em 0.5 para o estado $1^{(1)}$ e 0.5 para o estado $2^{(1)}$.

Observa-se que o modelo SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ com os 3 estados $0^{(1)}$, $1^{(1)}$ e $2^{(1)}$,o autômato $\mathcal{A}^{(2)}$ com os 3 estados $0^{(2)}$, $1^{(2)}$ e $2^{(2)}$. E o autômato $\mathcal{A}^{(3)}$ também com o 3 estados $0^{(3)}$, $1^{(3)}$ e $2^{(3)}$. Conforme pode-se observar na Figura 4.3.

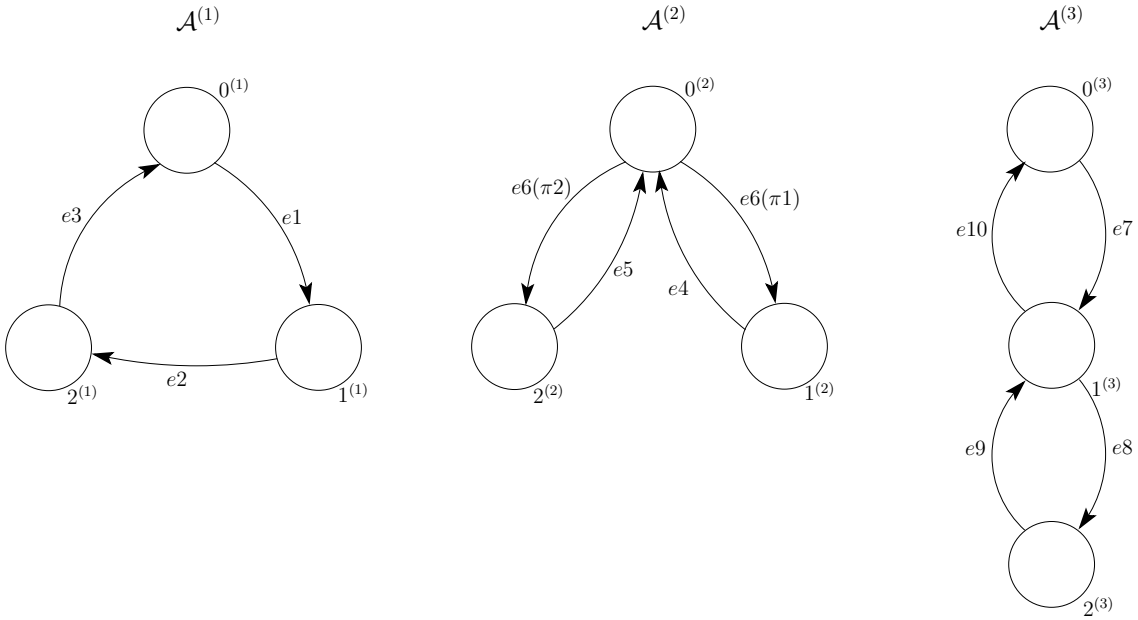

Figura 4.3: Exemplo de SAN com evento local com taxa funcional e probabilidade constante.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e1$	1	loc	$e6$	f
loc	$e2$	1	loc	$e7$	1
loc	$e3$	1	loc	$e8$	1
loc	$e4$	1	loc	$e9$	1
loc	$e5$	1	loc	$e10$	1

Tabela 4.5: Taxas dos eventos da SAN representada na Figura 4.3

Na Tabela 4.5 são apresentados os eventos do modelo da Figura 4.3. Nota-se que o autômato $\mathcal{A}^{(2)}$ possui um evento local $e6$ com uma taxa funcional f que é representada por:

$$f = \Big[((stA^{(3)} == 0^{(3)}) \times 2) + ((stA^{(3)} == 2^{(3)}) \times 5)\Big] \tag{4.2}$$

As probabilidades $\pi1$ e $\pi2$ que definem as saídas do estado $0^{(2)}$ são:

$$\pi1 = 0.5 \tag{4.3}$$

$$\pi 2 = 0.5 \tag{4.4}$$

Para converter o modelo SAN da Figura 4.3 para não utilizar taxas funcionais, o algoritmo irá percorrer todos os eventos procurando por algum evento ou probabilidade vinculada ao evento que seja do tipo funcional. Neste caso, somente o o evento $e6$ é do tipo funcional, portanto ele precisa ser substituído por eventos sincronizantes.

Para gerar os eventos sincronizantes, é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e, w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(2)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \emptyset$

Neste ponto, percorre-se todo o espaço de estados dos autômatos definidos anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e6$.

A cada estado global percorrido a função f é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

Por exemplo, avaliando a função para o estado global $\tilde{x}$ $(0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(1^{(2)},0^{(3)})$, obtém-se o resultado 1. Cria-se um novo evento chamado $e11$ cuja taxa constante vale 1.

O mesmo ocorre para os demais estados globais do subespaço de estados dos autômatos definidos acima. Na Tabela 4.6 pode-se verificar os estados globais e sua avaliação para a função f_e, são somente apresentados somente os valores que resultam diferente de zero.

$\tilde{x}$	$\tilde{y}$	Valor avaliado	Evento novo
$(0^{(2)},0^{(3)})$	$(1^{(2)},0^{(3)})$	1	$e11$
$(0^{(2)},0^{(3)})$	$(2^{(2)},0^{(3)})$	1	$e11$
$(0^{(2)},2^{(3)})$	$(1^{(2)},2^{(3)})$	2,5	$e12$
$(0^{(2)},2^{(3)})$	$(2^{(2)},2^{(3)})$	2,5	$e12$

Tabela 4.6: Resultado da avaliação da função f para o espaço de estados definido nela.

Podemos notar que o valor avaliado para o estado global $\tilde{x}$ $(0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(2^{(2)},0^{(3)})$ deu 1. Como já havia sido gerado um novo evento sincronizante $e11$ para o mesmo estado global $\tilde{x}$ e com o mesmo valor 1, o evento $e11$ é reaproveitado. Este controle pode ser verificado nas linhas 6 a 10 do algoritmo.

Cada estado do espaço de estados avaliado pela função deve receber uma transição para ele mesmo com o novo evento criado. Como a função utilizada neste modelo somente faz referencia ao autômato $\mathcal{A}^{(3)}$, o estado $0^{(3)}$ irá receber uma transição para ele mesmo com o evento $e11$ e o

estado $2^{(3)}$ irá receber uma transição para ele mesmo com o evento $e12$. A transição do estado onde o evento $e6$ estava originalmente, no autômato $\mathcal{A}^{(2)}$, também recebe os eventos $e11$ e $e12$.

Na Figura 4.4 é representado o modelo SAN da Figura 4.3 convertido para não utilizar taxas funcionais.

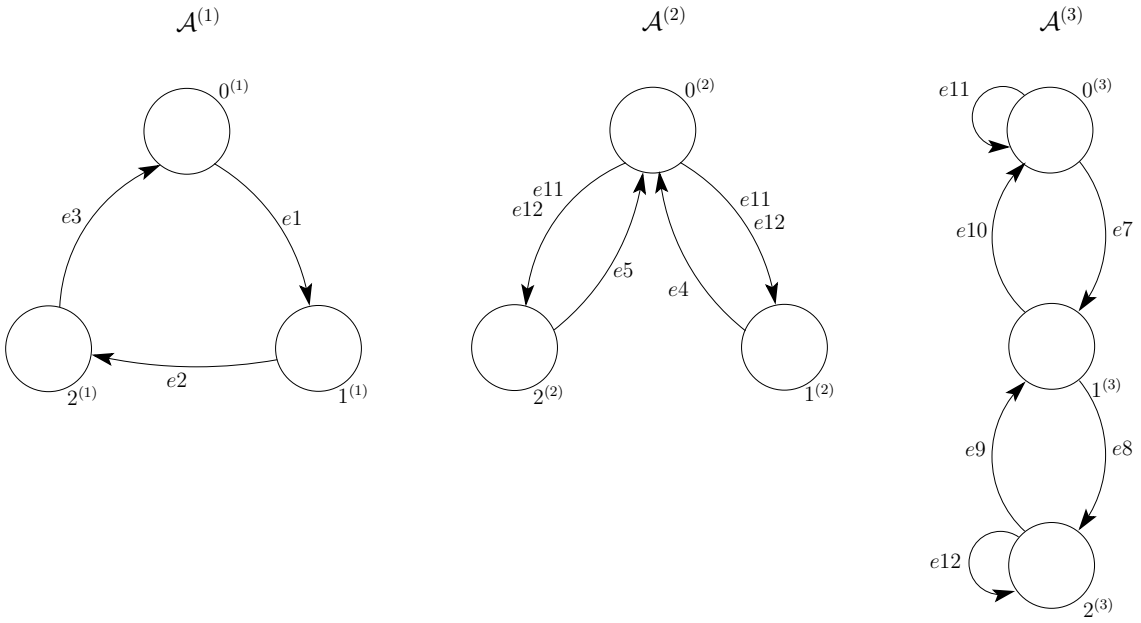

Figura 4.4: Exemplo de SAN com evento local com taxa funcional convertido para não mais usar funções.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e1$	1	loc	$e8$	1
loc	$e2$	1	loc	$e9$	1
loc	$e3$	1	loc	$e10$	1
loc	$e4$	1	syn	$e11$	2
loc	$e5$	1	syn	$e12$	5
loc	$e7$	1			

Tabela 4.7: Taxas dos eventos da SAN representada na Figura 4.4

Tando o modelo SAN original (Figura 4.3) quanto o modelo convertido (Figura 4.4) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.8 pode-se observar que para todos os estados globais da rede SAN as probabilidades são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

No Anexo B demonstra-se a definição dos modelos.

Estado global	Modelo GTA (Figura 4.3)	Modelo CTA (Figura 4.4)	Estado global	Modelo GTA (Figura 4.3)	Modelo CTA (Figura 4.4)
$0^{(1)}0^{(2)}0^{(3)}$	0.0426179603	0.0426179603	$1^{(1)}2^{(2)}0^{(3)}$	0.0342465753	0.0342465753
$0^{(1)}0^{(2)}1^{(3)}$	0.0593607304	0.0593607304	$1^{(1)}2^{(2)}1^{(3)}$	0.0258751903	0.0258751903
$0^{(1)}0^{(2)}2^{(3)}$	0.0243531202	0.0243531202	$1^{(1)}2^{(2)}2^{(3)}$	0.0433789954	0.0433789954
$0^{(1)}1^{(2)}0^{(3)}$	0.0342465753	0.0342465753	$2^{(1)}0^{(2)}0^{(3)}$	0.0426179603	0.0426179603
$0^{(1)}1^{(2)}1^{(3)}$	0.0258751903	0.0258751903	$2^{(1)}0^{(2)}1^{(3)}$	0.0593607304	0.0593607304
$0^{(1)}1^{(2)}2^{(3)}$	0.0433789954	0.0433789954	$2^{(1)}0^{(2)}2^{(3)}$	0.0243531202	0.0243531202
$0^{(1)}2^{(2)}0^{(3)}$	0.0342465753	0.0342465753	$2^{(1)}1^{(2)}0^{(3)}$	0.0342465753	0.0342465753
$0^{(1)}2^{(2)}1^{(3)}$	0.0258751903	0.0258751903	$2^{(1)}1^{(2)}1^{(3)}$	0.0258751903	0.0258751903
$0^{(1)}2^{(2)}2^{(3)}$	0.0433789954	0.0433789954	$2^{(1)}1^{(2)}2^{(3)}$	0.0433789954	0.0433789954
$1^{(1)}0^{(2)}0^{(3)}$	0.0426179603	0.0426179603	$2^{(1)}2^{(2)}0^{(3)}$	0.0342465753	0.0342465753
$1^{(1)}0^{(2)}1^{(3)}$	0.0593607304	0.0593607304	$2^{(1)}2^{(2)}1^{(3)}$	0.0258751903	0.0258751903
$1^{(1)}0^{(2)}2^{(3)}$	0.0243531202	0.0243531202	$2^{(1)}2^{(2)}2^{(3)}$	0.0433789954	0.0433789954
$1^{(1)}1^{(2)}0^{(3)}$	0.0342465753	0.0342465753			
$1^{(1)}1^{(2)}1^{(3)}$	0.0258751903	0.0258751903			
$1^{(1)}1^{(2)}2^{(3)}$	0.0433789954	0.0433789954			

Tabela 4.8: Resultados dos modelos SAN das Figuras 4.3 e 4.4

4.2.3 Rede SAN com evento local com taxa constante e probabilidades funcionais

Nesta subseção será utilizado um modelo SAN definido na Figura 4.5. O modelo possui somente eventos do tipo local e com taxas constantes. Além disto, o modelo SAN especifica probabilidades funcionais para sair do estado $0^{(1)}$ do autômato $\mathcal{A}^{(1)}$.

Observa-se que o modelo SAN possui 2 autômatos, o autômato $\mathcal{A}^{(1)}$ com 4 estados ($0^{(1)}$, $1^{(1)}$, $2^{(1)}$ e $3^{(1)}$) e o autômato $\mathcal{A}^{(2)}$ com 3 estados ($0^{(2)}$, $1^{(2)}$ e $2^{(2)}$). Conforme pode-se observar na Figura 4.5.

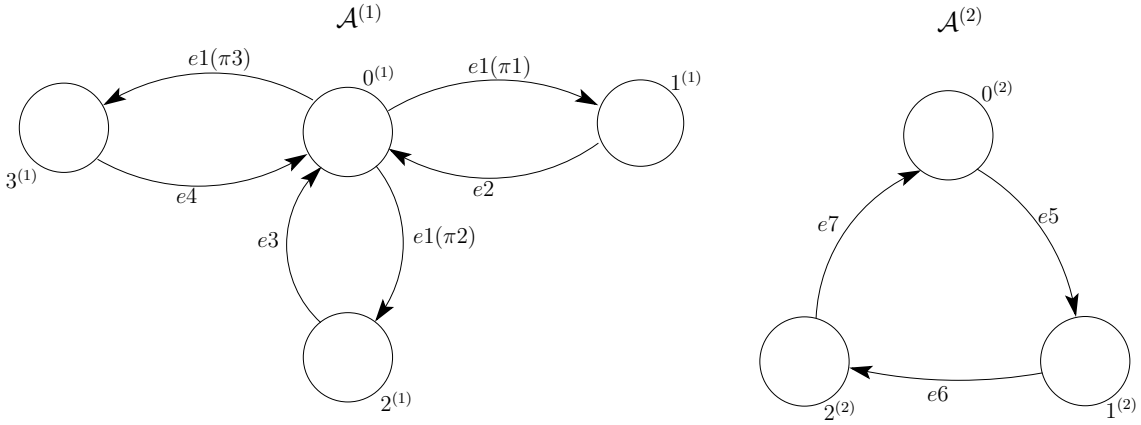

Figura 4.5: Exemplo de SAN com evento local com taxa constante e probabilidades funcionais.

Na Tabela 4.9 são apresentados os eventos do modelo da Figura 4.5. Podemos notar que todos os eventos são locais e com valores constantes como taxa.

Tipo	Evento	Taxa
loc	$e1$	9
loc	$e2$	1
loc	$e3$	1
loc	$e4$	1
loc	$e5$	1
loc	$e6$	1
loc	$e7$	1

Tabela 4.9: Taxas dos eventos da SAN representada na Figura 4.5

Na Figura 4.5, podemos notar que as saídas do estado $0^{(1)}$ são definidas pelas probabilidades $\pi1$, $\pi2$ e $\pi3$. As funções de cada probabilidade são:

$$\pi1 = ((stA^{(2)} == 0) * 0,6) + ((stA^{(2)} == 1) * 0,2) + ((stA^{(2)} == 2) * 0,2) \quad (4.5)$$

$$\pi2 = ((stA^{(2)} == 0) * 0,2) + ((stA^{(2)} == 1) * 0,6) + ((stA^{(2)} == 2) * 0,2) \quad (4.6)$$

$$\pi3 = ((stA^{(2)} == 0) * 0,2) + ((stA^{(2)} == 1) * 0,2) + ((stA^{(2)} == 2) * 0,6) \quad (4.7)$$

Para converter o modelo SAN da Figura 4.5 para não utilizar taxas funcionais, o evento $e1$ e as probabilidades $\pi1$, $\pi2$ e $\pi3$ associadas a ele, necessitam ser substituídos por eventos sincronizantes, de acordo com a avaliação da função das probabilidades e da taxa do evento $e1$.

Primeiramente, para gerar os eventos sincronizantes é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)} \}$

$w_{f_e} \quad = \emptyset$

$w_{\pi_e} \quad = \{ \mathcal{A}^{(2)} \}$

Neste ponto, percorre-se todo o espaço de estados definido anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$, conforme linha 5 do algoritmo.

A cada estado global percorrido a função da probabilidade é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

No nosso exemplo, para o estado global $\tilde{x}$ ($0^{(1)}$,$0^{(2)}$) e o sucessor $\tilde{y}$ ($1^{(1)}$,$0^{(2)}$), o resultado da avaliação de $e1(\pi1)$ é 5,4. Será criado um evento sincronizante $e8$ com a taxa constante 5,4. Este novo evento será associado a transição entre os eventos $0^{(1)}$ e $1^{(1)}$ e uma transição do estado $0^{(2)}$ para ele mesmo.

Na Tabela 4.10 podemos ver os resultados das probabilidades funcionais do evento $e1$ avaliadas para as transições entre os estados globais $\tilde{x}$ e $\tilde{y}$.

$\tilde{x}$	$\tilde{y}$	Valor Avaliado	Evento Novo
$(0^{(1)},0^{(2)})$	$(1^{(1)},0^{(2)})$	5,4	$e8$
$(0^{(1)},0^{(2)})$	$(2^{(1)},0^{(2)})$	1,8	$e9$
$(0^{(1)},0^{(2)})$	$(3^{(1)},0^{(2)})$	1,8	$e9$
$(0^{(1)},1^{(2)})$	$(1^{(1)},1^{(2)})$	1,8	$e10$
$(0^{(1)},1^{(2)})$	$(2^{(1)},1^{(2)})$	5,4	$e11$
$(0^{(1)},1^{(2)})$	$(3^{(1)},1^{(2)})$	1,8	$e10$
$(0^{(1)},2^{(2)})$	$(1^{(1)},2^{(2)})$	1,8	$e12$
$(0^{(1)},2^{(2)})$	$(2^{(1)},2^{(2)})$	1,8	$e12$
$(0^{(1)},2^{(2)})$	$(3^{(1)},2^{(2)})$	5,4	$e13$

Tabela 4.10: Resultado da avaliação de $e1(\pi)$ para os estados globais $\tilde{x}$ e $\tilde{y}$.

Todas as transições em que o evento $e1$ aparece serão substituídas para utilizar o novo evento sincronizante com a taxa definida na Tabela 4.10. Este evento sincronizante também deve ocorrer para os estados que são domínio das probabilidades funcionais de e.

Na Figura 4.6 é representado o modelo SAN da Figura 4.5 convertido para não utilizar taxas funcionais.

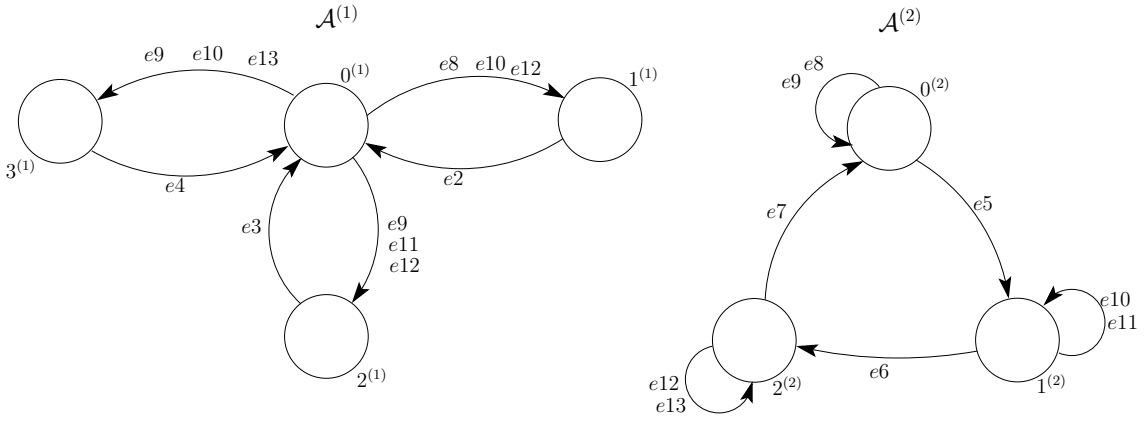

Figura 4.6: Exemplo de SAN da Figura 4.5 convertido para não mais usar funções.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e2$	1	syn	$e8$	5,4
loc	$e3$	1	syn	$e9$	1,8
loc	$e4$	1	syn	$e10$	1,8
loc	$e5$	1	syn	$e11$	5,4
loc	$e6$	1	syn	$e12$	1,8
loc	$e7$	1	syn	$e12$	5,4

Tabela 4.11: Taxas dos eventos da SAN representada na Figura 4.6

Tando o modelo SAN original (Figura 4.5) quanto o modelo convertido (Figura 4.6) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.12 pode-se observar que para todos os estados globais da rede SAN as probabilidades do modelo GTA e do modelo CTA são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

Estado global	Modelo GTA (Figura 4.5)	Modelo CTA (Figura 4.6)	Estado global	Modelo GTA (Figura 4.5)	Modelo CTA (Figura 4.6)
$0^{(1)}0^{(2)}$	0.0333333333	0.0333333333	$2^{(1)}0^{(2)}$	0.0771428573	0.0771428573
$0^{(1)}1^{(2)}$	0.0333333333	0.0333333333	$2^{(1)}1^{(2)}$	0.1285714286	0.1285714286
$0^{(1)}2^{(2)}$	0.0333333333	0.0333333333	$2^{(1)}2^{(2)}$	0.0942857140	0.0942857140
$1^{(1)}0^{(2)}$	0.1285714286	0.1285714286	$3^{(1)}0^{(2)}$	0.0942857140	0.0942857140
$1^{(1)}1^{(2)}$	0.0942857140	0.0942857140	$3^{(1)}1^{(2)}$	0.0771428573	0.0771428573
$1^{(1)}2^{(2)}$	0.0771428573	0.0771428573	$3^{(1)}2^{(2)}$	0.1285714286	0.1285714286

Tabela 4.12: Resultados dos modelos SAN das Figuras 4.5 e 4.6

No Anexo B demonstra-se a definição dos modelos.

4.2.4 Rede SAN com evento local com taxa funcional e probabilidades funcionais

Nesta subseção será utilizado um modelo SAN definido na Figura 4.7. O modelo possui somente eventos do tipo local, um deles com taxa do tipo funcional. . Além disto, o modelo especifica probabilidades funcionais para sair do estado $0^{(1)}$ do autômato $\mathcal{A}^{(1)}$.

Observa-se que o modelo SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ com 3 estados ($0^{(1)}$, $1^{(1)}$ e $2^{(1)}$), o autômato $\mathcal{A}^{(2)}$ com 3 estados ($0^{(2)}$, $1^{(2)}$ e $2^{(2)}$) e o autômato $\mathcal{A}^{(3)}$ com 2 estados ($0^{(2)}$, $1^{(2)}$). Conforme pode-se observar na Figura 4.7.

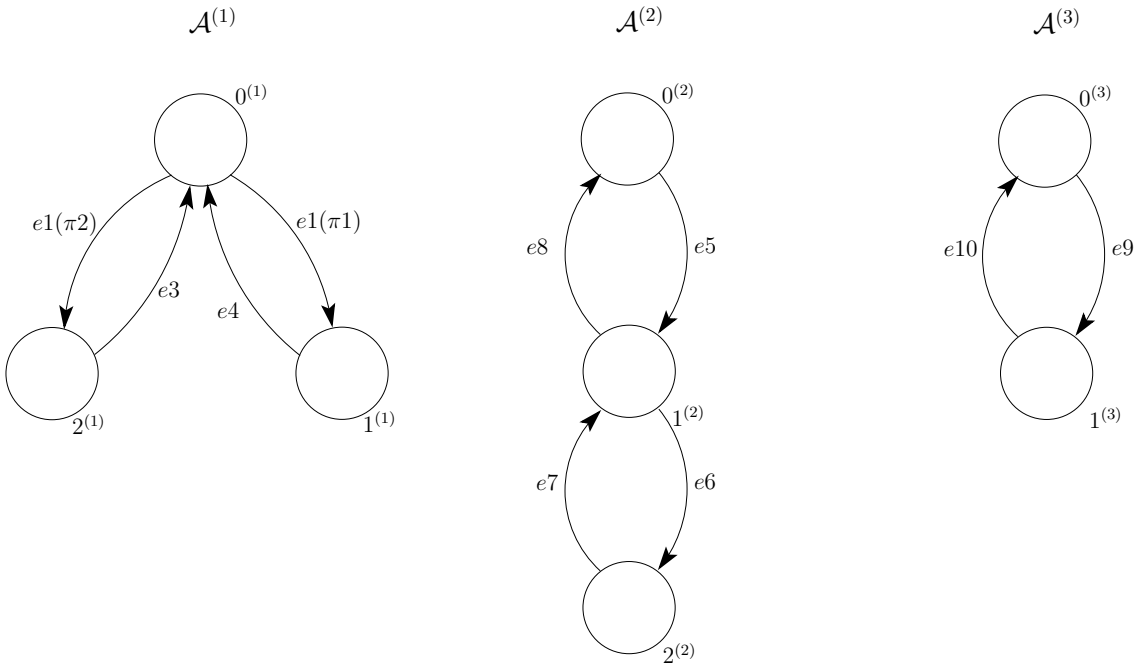

Figura 4.7: Exemplo de SAN com evento local com taxa funcional e probabilidade funcional.

Na Tabela 4.13 são apresentados os eventos do modelo da Figura 4.7. Podemos notar que todos os eventos são locais e com valores constantes como taxa, com exceção do evento $e1$ que é do tipo funcional.

Tipo	Evento	Taxa
loc	e1	f
loc	e3	1
loc	e4	1
loc	e5	1
loc	e6	1
loc	e7	1
loc	e8	1
loc	e9	1
loc	e10	1

Tabela 4.13: Taxas dos eventos da SAN representada na Figura 4.7

O evento local $e1$ utiliza a taxa funcional f que é representado por:

$$f = ((stA^{(3)} == 0^{(3)}) \times 6) + ((stA^{(3)} == 1^{(3)}) \times 4) \tag{4.8}$$

As probabilidades funcionais $\pi 1$ e $\pi 2$ que definem a saída do estado $0^{(1)}$ do autômato $\mathcal{A}^{(1)}$ são definidas por:

$$\pi 1 = ((stA^{(2)} == 0^{(2)}) \times 0,8) + ((stA^{(2)} == 2^{(2)}) \times 0.2) \tag{4.9}$$

$$\pi 2 = ((stA^{(2)} == 0^{(2)}) \times 0,2) + ((stA^{(2)} == 2^{(2)}) \times 0.8) \tag{4.10}$$

Para converter o modelo SAN da Figura 4.7 para não utilizar taxas e probabilidades funcionais, o evento $e1$ e as probabilidades $\pi 1$, $\pi 2$ e $\pi 3$ associadas a ele, necessitam ser substituídos por eventos sincronizantes, de acordo com a avaliação da função das probabilidades e da taxa do evento $e1$.

Primeiramente, para gerar os eventos sincronizantes é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \{ \mathcal{A}^{(2)} \}$

Neste ponto, percorre-se todo o espaço de estados definido anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$, conforme linha 5 do algoritmo.

A cada estado global percorrido a função do evento $e1$ e das probabilidades são avaliadas e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

No nosso exemplo, para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(1^{(1)},0^{(2)},0^{(3)})$, o resultado da avaliação de $e1(\pi 1)$ é 4,8. Será criado um evento sincronizante $e11$ com a taxa constante 4,8. Este novo evento será associado a transição entre os eventos $0^{(1)}$ e $1^{(1)}$ e uma transição do estado $0^{(2)}$ e $0^{(3)}$ para eles mesmos.

Na Tabela 4.14 podemos ver os resultados das probabilidades funcionais do evento $e1$ avaliadas para as transições entre os estados globais $\tilde{x}$ e $\tilde{y}$.

$\tilde{x}$	$\tilde{y}$	Valor Avaliado	Evento Novo
$(0^{(1)},0^{(2)},0^{(3)})$	$(1^{(1)},0^{(2)},0^{(3)})$	4,8	$e11$
$(0^{(1)},0^{(2)},0^{(3)})$	$(2^{(1)},0^{(2)},0^{(3)})$	1,2	$e12$
$(0^{(1)},0^{(2)},1^{(3)})$	$(1^{(1)},0^{(2)},1^{(3)})$	3,2	$e13$
$(0^{(1)},0^{(2)},1^{(3)})$	$(2^{(1)},0^{(2)},1^{(3)})$	0,8	$e14$
$(0^{(1)},1^{(2)},0^{(3)})$	$(1^{(1)},1^{(2)},0^{(3)})$	0	
$(0^{(1)},1^{(2)},0^{(3)})$	$(2^{(1)},1^{(2)},0^{(3)})$	0	
$(0^{(1)},1^{(2)},1^{(3)})$	$(1^{(1)},1^{(2)},1^{(3)})$	0	
$(0^{(1)},1^{(2)},1^{(3)})$	$(2^{(1)},1^{(2)},1^{(3)})$	0	
$(0^{(1)},2^{(2)},0^{(3)})$	$(1^{(1)},2^{(2)},0^{(3)})$	1,2	$e15$
$(0^{(1)},2^{(2)},0^{(3)})$	$(2^{(1)},2^{(2)},0^{(3)})$	4,8	$e16$
$(0^{(1)},2^{(2)},1^{(3)})$	$(1^{(1)},2^{(2)},1^{(3)})$	0,8	$e17$
$(0^{(1)},2^{(2)},1^{(3)})$	$(1^{(1)},2^{(2)},1^{(3)})$	3,2	$e18$

Tabela 4.14: Resultado da avaliação da função f para o espaço de estados definido nela.

Todas as transições em que o evento $e1$ aparece serão substituídas para utilizar o novo evento sincronizante com a taxa definida na Tabela 4.14. Este evento sincronizante também deve ocorrer para os estados que são domínio das probabilidades funcionais de e.

Na Figura 4.8 é representado o modelo SAN da Figura 4.7 convertido para não utilizar taxas funcionais.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	e3	1	syn	e11	4.8
loc	e4	1	syn	e12	1.2
loc	e5	1	syn	e13	3.2
loc	e6	1	syn	e14	0.8
loc	e7	1	syn	e15	1.2
loc	e8	1	syn	e16	4.8
loc	e9	1	syn	e17	0.8
loc	e10	1	syn	e18	3.2

Tabela 4.15: Taxas dos eventos da SAN representada na Figura 4.8

Tando o modelo SAN original (Figura 4.7) quanto o modelo convertido (Figura 4.8) foram

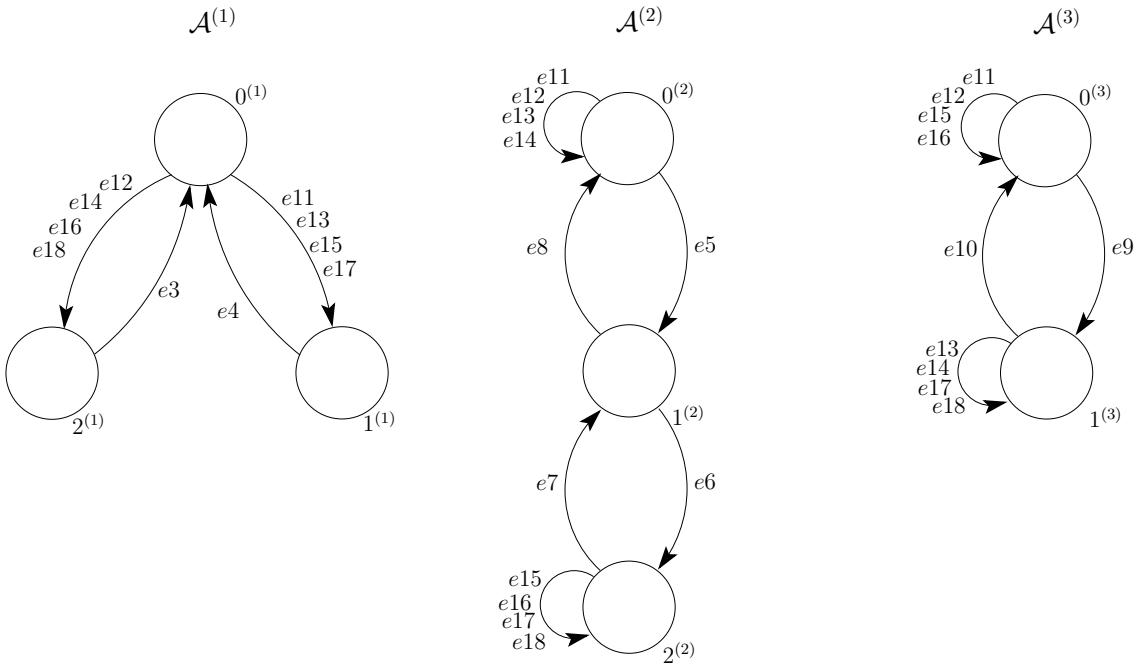

Figura 4.8: Exemplo do SAN da Figura 4.7 convertido para não mais usar funções.

resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.16 pode-se observar que para todos os estados globais da rede SAN as probabilidades são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

Estado global	Modelo GTA (Figura 4.7)	Modelo CTA (Figura 4.8)	Estado global	Modelo GTA (Figura 4.7)	Modelo CTA (Figura 4.8)
$0^{(1)}0^{(2)}0^{(3)}$	0.0315876974	0.0315876974	$1^{(1)}1^{(2)}1^{(3)}$	0.0428096425	0.0428096425
$0^{(1)}0^{(2)}1^{(3)}$	0.0399002493	0.0399002493	$1^{(1)}2^{(2)}0^{(3)}$	0.0402327517	0.0402327517
$0^{(1)}1^{(2)}0^{(3)}$	0.0777223607	0.0777223607	$1^{(1)}2^{(2)}1^{(3)}$	0.0383208647	0.0383208647
$0^{(1)}1^{(2)}1^{(3)}$	0.0810473815	0.0810473815	$2^{(1)}0^{(2)}0^{(3)}$	0.0402327517	0.0402327517
$0^{(1)}2^{(2)}0^{(3)}$	0.0315876974	0.0315876974	$2^{(1)}0^{(2)}1^{(3)}$	0.0383208647	0.0383208647
$0^{(1)}2^{(2)}1^{(3)}$	0.0399002493	0.0399002493	$2^{(1)}1^{(2)}0^{(3)}$	0.0444721529	0.0444721529
$1^{(1)}0^{(2)}0^{(3)}$	0.0948462175	0.0948462175	$2^{(1)}1^{(2)}1^{(3)}$	0.0428096425	0.0428096425
$1^{(1)}0^{(2)}1^{(3)}$	0.0884455524	0.0884455524	$2^{(1)}2^{(2)}0^{(3)}$	0.0948462175	0.0948462175
$1^{(1)}1^{(2)}0^{(3)}$	0.0444721529	0.0444721529	$2^{(1)}2^{(2)}1^{(3)}$	0.0884455524	0.0884455524

Tabela 4.16: Resultados dos modelos SAN das Figuras 4.7 e 4.8

No Anexo B demonstra-se a definição dos modelos.

4.2.5 Rede SAN com evento sincronizante com taxa funcional sem especificação de probabilidades

Nesta subseção será utilizado um modelo SAN que possui eventos do tipo local e sincronizantes, um deles com taxa funcional. Nenhum evento define explicitamente probabilidades, ou seja, as probabilidades de todas as tansições é 1.

O modelo SAN é apresentado na Figura 4.9. Pode-se notar que a SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ possui 3 estados ($0^{(1)}$, $1^{(1)}$ e $2^{(1)}$), o autômato $\mathcal{A}^{(2)}$ possui 2 estados ($0^{(2)}$ e $1^{(2)}$) enquanto o autômato $\mathcal{A}^{(3)}$ possui 3 estados ($0^{(3)}$, $1^{(3)}$ e $2^{(3)}$).

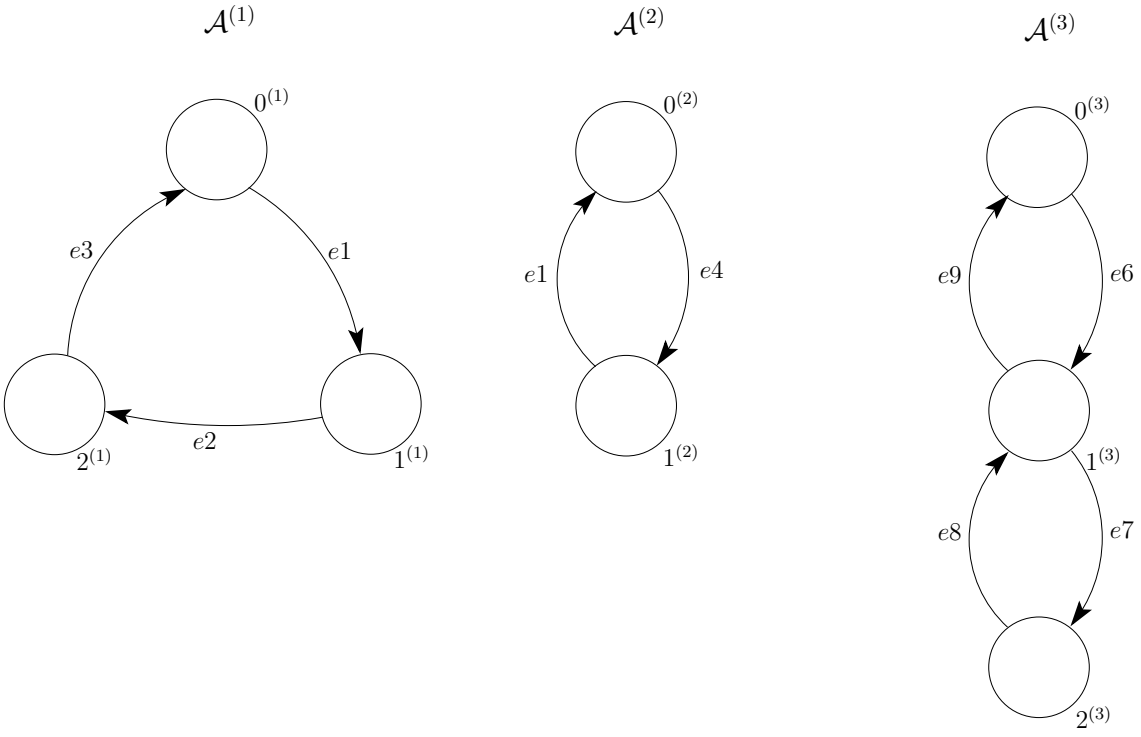

Figura 4.9: Exemplo de SAN com evento sincronizante com taxa funcional.

Na Tabela 4.17 são apresentados os eventos do modelo da Figura 4.9. Nota-se que os autômatos $\mathcal{A}^{(1)}$ e $\mathcal{A}^{(2)}$ possuem um evento sincronizante $e1$ representado através de uma taxa funcional. A função que representa $e1$ avalia o comportamento do autômato $\mathcal{A}^{(3)}$.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
syn	$e1$	f	loc	$e6$	1
loc	$e2$	1	loc	$e7$	1
loc	$e3$	1	loc	$e8$	1
loc	$e4$	1	loc	$e9$	1

Tabela 4.17: Taxas dos eventos da SAN representada na Figura 4.9

O evento local $e1$ utiliza a taxa funcional f que é representado por:

$$f = \Big[((stA^{(3)} == 0^{(3)}) \times 4) + ((stA^{(3)} == 2^{(3)}) \times 5)\Big] \tag{4.11}$$

Para converter o modelo SAN da Figura 4.9 para não utilizar taxas funcionais, o único evento que necessita ser substituído por eventos sincronizantes é o evento $e1$, por possuir associado a ele uma taxa do tipo funcional.

Primeiramente, para gerar os eventos sincronizantes é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$. Conforme linha 3 do algoritmo.

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)}, \mathcal{A}^{(2)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \emptyset$

Neste ponto, percorre-se todo o espaço de estados dos autômatos definidos anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$, conforme a linha 5 do algoritmo.

A cada estado global percorrido a função é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

Na Tabela 4.18 podemos ver os resultados da função f do evento $e1$ avaliadas para as transições entre os estados globais $\tilde{x}$ e $\tilde{y}$.

$\tilde{x}$	$\tilde{y}$	Valor Avaliado	Evento Novo
$(0^{(1)},1^{(2)},0^{(3)})$	$(1^{(1)},0^{(2)},0^{(3)})$	4	$e10$
$(0^{(1)},1^{(2)},1^{(3)})$	$(1^{(1)},0^{(2)},1^{(3)})$	0	
$(0^{(1)},1^{(2)},2^{(3)})$	$(1^{(1)},0^{(2)},2^{(3)})$	5	$e11$

Tabela 4.18: Resultado da avaliação da função f para o espaço de estados definido nela.

Cada estado do espaço de estados avaliado pela função deve receber uma transição para ele mesmo com o novo evento criado. Por exemplo para o espaço de estados ($0^{(1)}$, $1^{(2)}$ e $0^{(3)}$), o estado $0^{(3)}$ irão receber uma transição para ele mesmo com o evento $e10$. Assim ocorre para todo o espaço de estados. A transição do estado onde o evento $e1$ estava originalmente também recebe o evento $e10$.

Na Figura 4.10 é representado o modelo SAN da Figura 4.9 convertido para não utilizar taxas funcionais.

Tando o modelo SAN original (Figura 4.9) quanto o modelo convertido (Figura 4.10) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.20 pode-se observar que para todos os estados globais da rede SAN as probabilidades são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

No Anexo B demonstra-se a definição dos modelos.

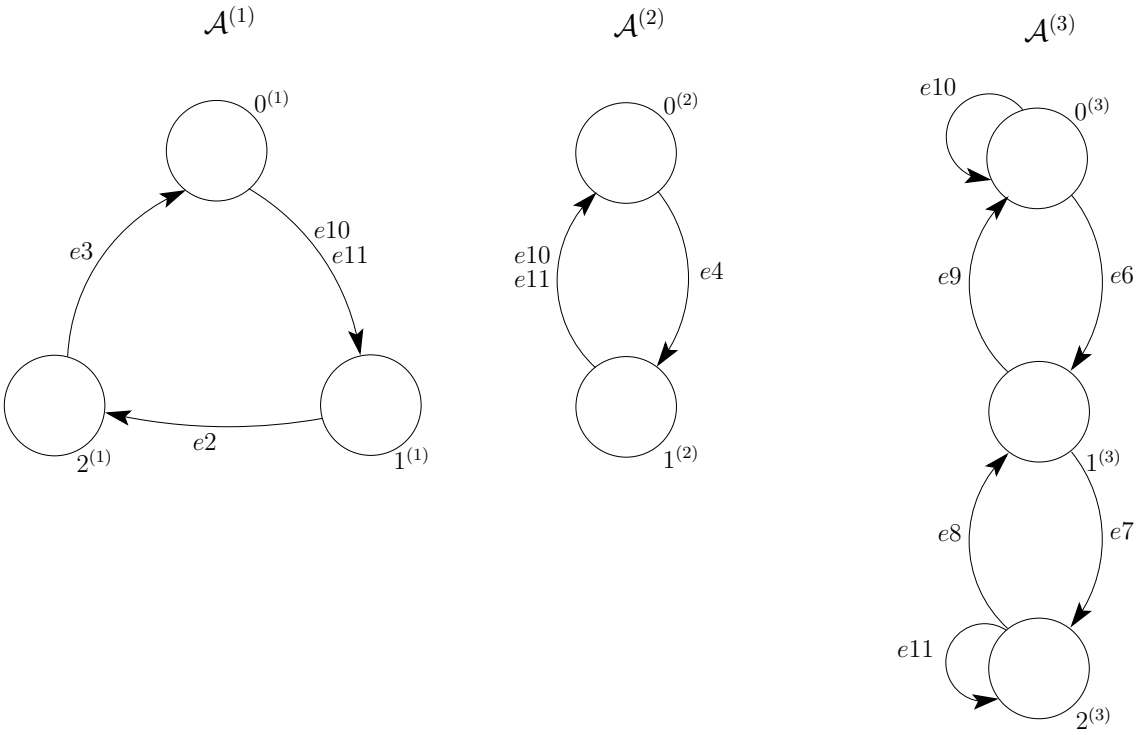

Figura 4.10: Exemplo de SAN com evento sincronizante com taxa funcional convertido para não mais usar funções.

Tipo	Evento	Taxa
loc	$e2$	1
loc	$e3$	1
loc	$e4$	1
loc	$e5$	1
loc	$e6$	1
loc	$e7$	1
loc	$e8$	1
loc	$e9$	1
syn	$e10$	4
syn	$e11$	5

Tabela 4.19: Taxas dos eventos da SAN representada na Figura 4.10

4.2.6 Rede SAN com evento sincronizante com taxa funcional e probabilidades constantes

Nesta subseção será utilizado um modelo SAN definido na Figura 4.11. O modelo possui eventos do tipo local e sincronizantes, o sincronizante $e1$ com taxa funcional e que especifica probabilidades constantes. No caso deste exemplo, as probabilidades definem a transisão do estado $0^{(2)}$ em 0.7 para o estado $1^{(1)}$ e 0.3 para o estado $2^{(1)}$.

Observa-se que o modelo SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ com os 3 estados $0^{(1)}$, $1^{(1)}$ e $2^{(1)}$,o autômato $\mathcal{A}^{(2)}$ com os 3 estados $0^{(2)}$, $1^{(2)}$ e $2^{(2)}$ e o autômato $\mathcal{A}^{(3)}$ também com o 3 estados $0^{(3)}$, $1^{(3)}$ e $2^{(3)}$. Conforme pode-se observar na Figura 4.11.

Na Tabela 4.21 são apresentados os eventos do modelo da Figura 4.11. Nota-se que o autômato

Estado global Estado global	Modelo GTA (Figura 4.9)	Modelo CTA (Figura 4.10)	Estado global Estado global	Modelo GTA (Figura 4.9)	Modelo CTA (Figura 4.10)
$0^{(1)}0^{(2)}0^{(3)}$	0.0306881542	0.0306881542	$1^{(1)}1^{(2)}0^{(3)}$	0.0443867335	0.0443867335
$0^{(1)}0^{(2)}1^{(3)}$	0.0291137739	0.0291137739	$1^{(1)}1^{(2)}1^{(3)}$	0.0995133399	0.0995133399
$0^{(1)}0^{(2)}2^{(3)}$	0.0311786153	0.0311786153	$1^{(1)}1^{(2)}2^{(3)}$	0.0372750469	0.0372750469
$0^{(1)}1^{(2)}0^{(3)}$	0.0443867335	0.0443867335	$2^{(1)}0^{(2)}0^{(3)}$	0.0306881542	0.0306881542
$0^{(1)}1^{(2)}1^{(3)}$	0.0995133399	0.0995133399	$2^{(1)}0^{(2)}1^{(3)}$	0.0291137739	0.0291137739
$0^{(1)}1^{(2)}2^{(3)}$	0.0372750469	0.0372750469	$2^{(1)}0^{(2)}2^{(3)}$	0.0311786153	0.0311786153
$1^{(1)}0^{(2)}0^{(3)}$	0.0306881542	0.0306881542	$2^{(1)}1^{(2)}0^{(3)}$	0.0443867335	0.0443867335
$1^{(1)}0^{(2)}1^{(3)}$	0.0291137739	0.0291137739	$2^{(1)}1^{(2)}1^{(3)}$	0.0995133399	0.0995133399
$1^{(1)}0^{(2)}2^{(3)}$	0.0311786153	0.0311786153	$2^{(1)}1^{(2)}2^{(3)}$	0.0372750469	0.0372750469

Tabela 4.20: Resultados dos modelos SAN das Figuras 4.9 e 4.10

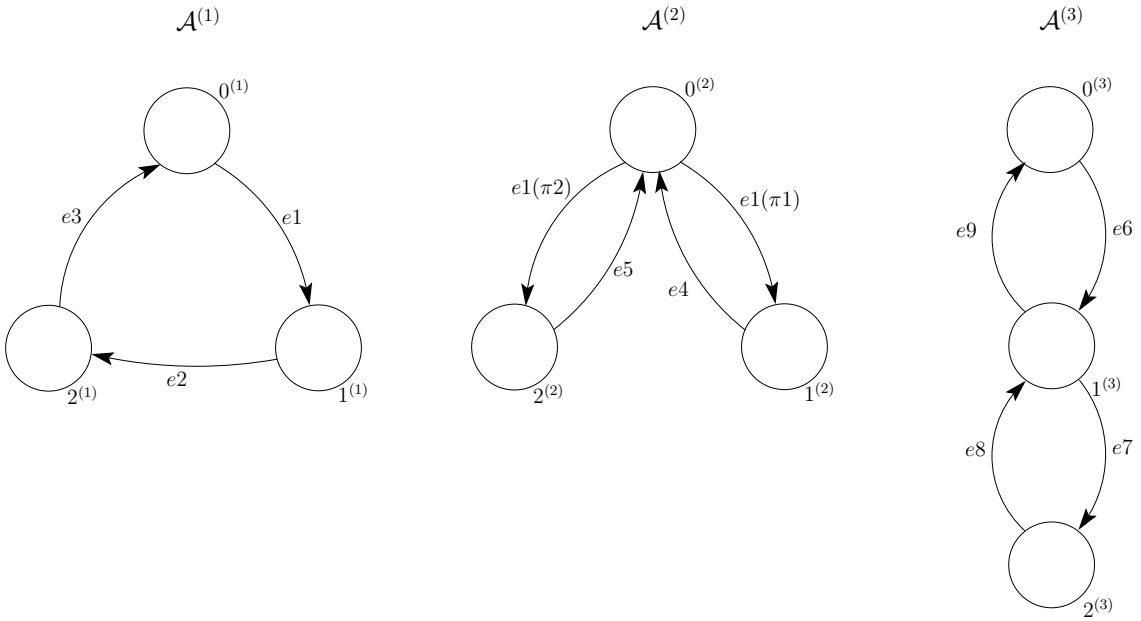

Figura 4.11: Exemplo de SAN com evento sincronizante com taxa funcional e probabilidade constante.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
syn	$e1$	f	loc	$e6$	1
loc	$e2$	1	loc	$e7$	1
loc	$e3$	1	loc	$e8$	1
loc	$e4$	1	loc	$e9$	1
loc	$e5$	1			

Tabela 4.21: Taxas dos eventos da SAN representada na Figura 4.11

$\mathcal{A}^{(2)}$ possui um evento sincronizante $e1$ com uma taxa funcional f que é representada por:

$$f = ((stA^{(3)} == 0^{(3)}) \times 5) + ((stA^{(3)} == 2^{(3)}) \times 3) \tag{4.12}$$

As probabilidades constantes $\pi 1$ e $\pi 2$ que definem a saída do estado $0^{(2)}$ do autômato $\mathcal{A}^{(2)}$ são:

$$\pi 1 = 0,7 \tag{4.13}$$

$$\pi 2 = 0,3 \tag{4.14}$$

Para converter o modelo SAN da Figura 4.11 para não utilizar taxas funcionais, o algoritmo irá percorrer todos os eventos procurando por algum evento ou probabilidade vinculada ao evento que seja do tipo funcional. Neste caso, somente o o evento $e1$ é do tipo funcional, portanto ele precisa ser substituído por novos eventos sincronizantes.

Para gerar os eventos sincronizantes, é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e, w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)}, \mathcal{A}^{(2)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \emptyset$

Neste ponto, percorre-se todo o espaço de estados dos autômatos definidos anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$.

A cada estado global percorrido a função f é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

Por exemplo, avaliando a função para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(1^{(1)},1^{(2)},0^{(3)})$, obtém-se então $e1(\pi 1) = 3,5$. Cria-se um novo evento chamado $e10$ cuja taxa constante vale $3,5$.

Será criado uma nova transição do estado $0^{(3)}$ para ele mesmo com o novo evento $e10$ criado. Também para as transições que continham o evento $e1$ passarão a conter o evento $e10$.

O mesmo ocorre para os demais estados globais do subespaço de estados dos autômatos definido anteriormente. Na Tabela 4.22 pode-se verificar os estados globais e sua avaliação para a função f_e, são somente apresentados somente os valores que resultam diferente de zero.

$\tilde{x}$	$\tilde{y}$	Valor avaliado	Evento novo
$(0^{(1)},0^{(2)},0^{(3)})$	$(1^{(1)},1^{(2)},0^{(3)})$	3,5	$e10$
$(0^{(1)},0^{(2)},0^{(3)})$	$(1^{(1)},2^{(2)},0^{(3)})$	1,5	$e11$
$(0^{(1)},0^{(2)},1^{(3)})$	$(1^{(1)},1^{(2)},1^{(3)})$	0	
$(0^{(1)},0^{(2)},1^{(3)})$	$(1^{(1)},2^{(2)},1^{(3)})$	0	
$(0^{(1)},0^{(2)},2^{(3)})$	$(1^{(1)},1^{(2)},2^{(3)})$	2,1	$e12$
$(0^{(1)},0^{(2)},2^{(3)})$	$(1^{(1)},2^{(2)},2^{(3)})$	0,9	$e13$

Tabela 4.22: Resultado da avaliação da função f para o espaço de estados definido nela.

Cada estado do espaço de estados avaliado pela função deve receber uma transição para ele mesmo com o novo evento criado. Como a função utilizada neste modelo somente faz referencia ao autômato $\mathcal{A}^{(3)}$, o estado $0^{(3)}$ irá receber uma transição para ele mesmo com os eventos $e10$ e $e11$ e o estado $2^{(3)}$ irá receber uma transição para ele mesmo com os eventos $e12$ e $e13$. A transição do estado onde o evento $e1$ estava originalmente, nos autômatos $\mathcal{A}^{(1)}$ e $\mathcal{A}^{(2)}$, também recebem os novos eventos criados.

Na Figura 4.12 é representado o modelo SAN da Figura 4.11 convertido para não utilizar taxas funcionais.

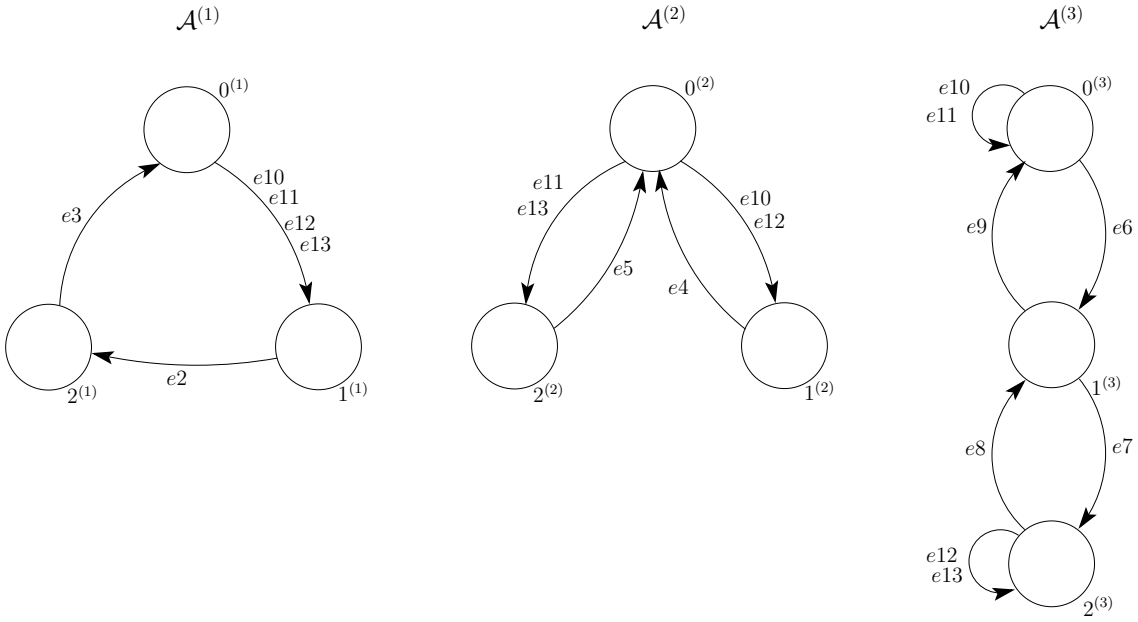

Figura 4.12: Exemplo do SAN da Figura 4.11 convertido para não mais usar funções.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e2$	1	loc	$e8$	1
loc	$e3$	1	loc	$e9$	1
loc	$e4$	1	syn	$e10$	3,5
loc	$e5$	1	syn	$e11$	1,5
loc	$e6$	1	syn	$e12$	2,1
loc	$e7$	1	syn	$e13$	0,9

Tabela 4.23: Taxas dos eventos da SAN representada na Figura 4.12

Tando o modelo SAN original (Figura 4.11) quanto o modelo convertido (Figura 4.12) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.24 pode-se observar que para todos os estados globais da rede SAN as probabilidades são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

No Anexo B demonstra-se a definição dos modelos.

Estado global	Modelo GTA (Figura 4.11)	Modelo CTA (Figura 4.12)	Estado global	Modelo GTA (Figura 4.11)	Modelo CTA (Figura 4.12)
$0^{(1)}0^{(2)}0^{(3)}$	0.0378668058	0.0378668058	$1^{(1)}2^{(2)}0^{(3)}$	0.0224984241	0.0224984241
$0^{(1)}0^{(2)}1^{(3)}$	0.1042807192	0.1042807192	$1^{(1)}2^{(2)}1^{(3)}$	0.0106950636	0.0106950636
$0^{(1)}0^{(2)}2^{(3)}$	0.0557226979	0.0557226979	$1^{(1)}2^{(2)}2^{(3)}$	0.0202818306	0.0202818306
$0^{(1)}1^{(2)}0^{(3)}$	0.0216428807	0.0216428807	$2^{(1)}0^{(2)}0^{(3)}$	0.0920017143	0.0920017143
$0^{(1)}1^{(2)}1^{(3)}$	0.0199641190	0.0199641190	$2^{(1)}0^{(2)}1^{(3)}$	0.0864517643	0.0864517643
$0^{(1)}1^{(2)}2^{(3)}$	0.0207808721	0.0207808721	$2^{(1)}0^{(2)}2^{(3)}$	0.0889231122	0.0889231122
$0^{(1)}2^{(2)}0^{(3)}$	0.0092755205	0.0092755205	$2^{(1)}1^{(2)}0^{(3)}$	0.0233216423	0.0233216423
$0^{(1)}2^{(2)}1^{(3)}$	0.0085560512	0.0085560512	$2^{(1)}1^{(2)}1^{(3)}$	0.0174686040	0.0174686040
$0^{(1)}2^{(2)}2^{(3)}$	0.0089060883	0.0089060883	$2^{(1)}1^{(2)}2^{(3)}$	0.0215976251	0.0215976251
$1^{(1)}0^{(2)}0^{(3)}$	0.0642350327	0.0642350327	$2^{(1)}2^{(2)}0^{(3)}$	0.0099949895	0.0099949895
$1^{(1)}0^{(2)}1^{(3)}$	0.0534753184	0.0534753184	$2^{(1)}2^{(2)}1^{(3)}$	0.0074865445	0.0074865445
$1^{(1)}0^{(2)}2^{(3)}$	0.0605407102	0.0605407102	$2^{(1)}2^{(2)}2^{(3)}$	0.0092561250	0.0092561250
$1^{(1)}1^{(2)}0^{(3)}$	0.0524963229	0.0524963229			
$1^{(1)}1^{(2)}1^{(3)}$	0.0249551486	0.0249551486			
$1^{(1)}1^{(2)}2^{(3)}$	0.0473242714	0.0473242714			

Tabela 4.24: Resultados dos modelos SAN das Figuras 4.11 e 4.12

4.2.7 Rede SAN com evento sincronizante com taxa constante e probabilidades funcionais

Nesta subseção será utilizado um modelo SAN definido na Figura 4.13. O modelo possui eventos do tipo local e sincronizantes, todos com taxas constantes. Além disto, o modelo SAN especifica probabilidades funcionais para sair do estado $0^{(1)}$ do autômato $\mathcal{A}^{(1)}$.

Observa-se que o modelo SAN possui 3 autômatos, o autômato $\mathcal{A}^{(1)}$ com 4 estados ($0^{(1)}$, $1^{(1)}$, $2^{(1)}$ e $3^{(1)}$), o autômato $\mathcal{A}^{(2)}$ com 3 estados ($0^{(2)}$, $1^{(2)}$ e $2^{(2)}$) e o autômato $\mathcal{A}^{(3)}$ com 2 estados ($0^{(3)}$ e $1^{(3)}$). Conforme pode-se observar na Figura 4.13.

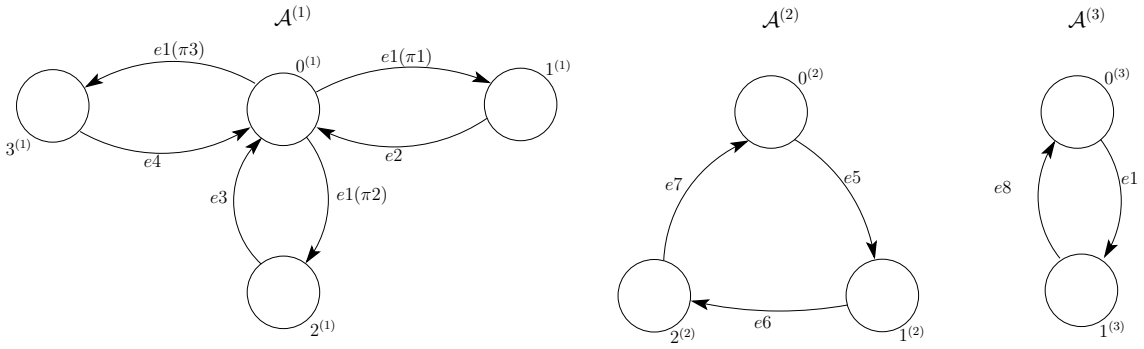

Figura 4.13: Exemplo de SAN com evento local com taxa constante e probabilidades funcionais.

Na Tabela 4.25 são apresentados os eventos do modelo da Figura 4.13. Podemos notar que todos os eventos possuem valores constantes como taxa.

Na Figura 4.13, podemos notar que as saídas do estado $0^{(1)}$ são definidas pelas probabilidades $\pi1$, $\pi2$ e $\pi3$. As funções de cada probabilidade são:

Tipo	Evento	Taxa
syn	$e1$	9
loc	$e2$	1
loc	$e3$	1
loc	$e4$	1
loc	$e5$	1
loc	$e6$	1
loc	$e7$	1
loc	$e8$	1

Tabela 4.25: Taxas dos eventos da SAN representada na Figura 4.13

$$\pi 1 = ((stA^{(2)} == 0) * 0,6) + ((stA^{(2)} == 1) * 0,2) + ((stA^{(2)} == 2) * 0,2) \quad (4.15)$$

$$\pi 2 = ((stA^{(2)} == 0) * 0,2) + ((stA^{(2)} == 1) * 0,6) + ((stA^{(2)} == 2) * 0,2) \quad (4.16)$$

$$\pi 3 = ((stA^{(2)} == 0) * 0,2) + ((stA^{(2)} == 1) * 0,2) + ((stA^{(2)} == 2) * 0,6) \quad (4.17)$$

Para converter o modelo SAN da Figura 4.13 para não utilizar taxas funcionais, o evento $e1$ e as probabilidades $\pi 1$, $\pi 2$ e $\pi 3$ associadas a ele, necessitam ser substituídos por eventos sincronizantes, de acordo com a avaliação da função das probabilidades e da taxa do evento $e1$.

Primeiramente, para gerar os eventos sincronizantes é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e \cup w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)}, \mathcal{A}^{(3)} \}$

$w_{f_e} \quad = \emptyset$

$w_{\pi_e} \quad = \{ \mathcal{A}^{(2)} \}$

Neste ponto, percorre-se todo o espaço de estados definido anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$, conforme linha 5 do algoritmo.

A cada estado global percorrido a função da probabilidade é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

No nosso exemplo, para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)})$ e o sucessor $\tilde{y}$ $(1^{(1)},0^{(2)},1^{(3)})$, o resultado da avaliação de $e1(\pi 1)$ é 5,4. Será criado um evento sincronizante $e9$ com a taxa constante 5,4. Este novo evento será associado à transição entre os eventos $0^{(1)}$ e $1^{(1)}$, entre os eventos $0^{(3)}$ e $1^{(3)}$ e uma transição do estado $0^{(2)}$ para ele mesmo.

Na Tabela 4.26 podemos ver os resultados das probabilidades funcionais do evento $e1$ avaliadas para as transições entre os estados globais $\tilde{x}$ e $\tilde{y}$.

$\tilde{x}$	$\tilde{y}$	Valor Avaliado	Evento Novo
$(0^{(1)},0^{(2)},0^{(3)})$	$(1^{(1)},0^{(2)},1^{(3)})$	5,4	$e9$
$(0^{(1)},0^{(2)},0^{(3)})$	$(2^{(1)},0^{(2)},1^{(3)})$	1,8	$e10$
$(0^{(1)},0^{(2)},0^{(3)})$	$(3^{(1)},0^{(2)},1^{(3)})$	1,8	$e10$
$(0^{(1)},1^{(2)},0^{(3)})$	$(1^{(1)},1^{(2)},1^{(3)})$	1,8	$e11$
$(0^{(1)},1^{(2)},0^{(3)})$	$(2^{(1)},1^{(2)},1^{(3)})$	5,4	$e12$
$(0^{(1)},1^{(2)},0^{(3)})$	$(3^{(1)},1^{(2)},1^{(3)})$	1,8	$e11$
$(0^{(1)},2^{(2)},0^{(3)})$	$(1^{(1)},2^{(2)},1^{(3)})$	1,8	$e13$
$(0^{(1)},2^{(2)},0^{(3)})$	$(2^{(1)},2^{(2)},1^{(3)})$	1,8	$e13$
$(0^{(1)},2^{(2)},0^{(3)})$	$(3^{(1)},2^{(2)},1^{(3)})$	5,4	$e14$

Tabela 4.26: Resultado da avaliação de $e1(\pi)$ para os estados globais $\tilde{x}$ e $\tilde{y}$.

Todas as transições em que o evento $e1$ aparece serão substituídas para utilizar o novo evento sincronizante com a taxa definida na Tabela 4.26. Este evento sincronizante também deve ocorrer para os estados que são domínio das probabilidades funcionais de e.

Na Figura 4.14 é representado o modelo SAN da Figura 4.13 convertido para não utilizar taxas funcionais.

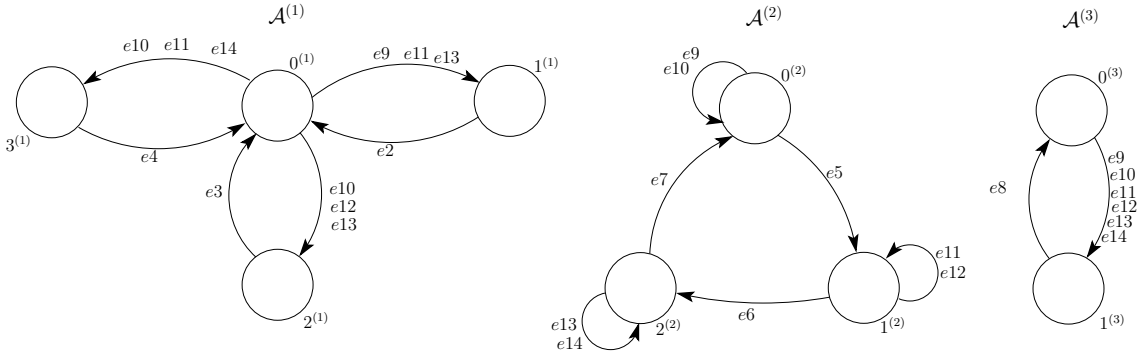

Figura 4.14: Exemplo de SAN da figura 4.13 convertido para não mais usar funções.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e2$	1	syn	$e9$	5,4
loc	$e3$	1	syn	$e10$	1,8
loc	$e4$	1	syn	$e11$	1,8
loc	$e5$	1	syn	$e12$	5,4
loc	$e6$	1	syn	$e13$	1,8
loc	$e7$	1	syn	$e14$	5,4
loc	$e8$	1	syn		

Tabela 4.27: Taxas dos eventos da SAN representada na Figura 4.6

Tando o modelo SAN original (Figura 4.13) quanto o modelo convertido (Figura 4.14) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Na Tabela 4.28 pode-se observar que para todos os estados globais da rede SAN as probabilidades

do modelo GTA e do modelo CTA são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

Estado global	Modelo GTA (Figura 4.13)	Modelo CTA (Figura 4.14)	Estado global	Modelo GTA (Figura 4.13)	Modelo CTA (Figura 4.14)
$0^{(1)}0^{(2)}0^{(3)}$	0.0229885057	0.0229885057	$1^{(1)}1^{(2)}1^{(3)}$	0.0302387267	0.0302387267
$0^{(1)}0^{(2)}1^{(3)}$	0.1034482750	0.1034482750	$1^{(1)}2^{(2)}0^{(3)}$	0.0293292916	0.0293292916
$0^{(1)}1^{(2)}0^{(3)}$	0.0229885057	0.0229885057	$1^{(1)}2^{(2)}1^{(3)}$	0.0238726790	0.0238726790
$0^{(1)}1^{(2)}1^{(3)}$	0.1034482750	0.1034482750	$2^{(1)}0^{(2)}0^{(3)}$	0.0293292916	0.0293292916
$0^{(1)}2^{(2)}0^{(3)}$	0.0229885057	0.0229885057	$2^{(1)}0^{(2)}1^{(3)}$	0.0238726790	0.0238726790
$0^{(1)}2^{(2)}1^{(3)}$	0.1034482750	0.1034482750	$2^{(1)}1^{(2)}0^{(3)}$	0.0393330809	0.0393330809
$1^{(1)}0^{(2)}0^{(3)}$	0.0393330809	0.0393330809	$2^{(1)}1^{(2)}1^{(3)}$	0.0493368700	0.0493368700
$1^{(1)}0^{(2)}1^{(3)}$	0.0493368700	0.0493368700	$2^{(1)}2^{(2)}0^{(3)}$	0.0347859040	0.0347859040
$1^{(1)}1^{(2)}0^{(3)}$	0.0347859040	0.0347859040	$2^{(1)}2^{(2)}1^{(3)}$	0.0302387267	0.0302387267

Tabela 4.28: Resultados dos modelos SAN das Figuras 4.13 e 4.14

No Anexo B demonstra-se a definição dos modelos.

4.2.8 Rede SAN com evento sincronizante com taxa funcional e probabilidades funcionais

Nesta subseção será utilizado um modelo SAN definido na Figura 4.15. O modelo possui eventos do tipo local e sincronizantes, o sincronizante $e1$ com taxa funcional e que especifica probabilidades funcionais.

Observa-se que o modelo SAN possui 4 autômatos, o autômato $\mathcal{A}^{(1)}$ com os 3 estados $0^{(1)}$, $1^{(1)}$ e $2^{(1)}$,o autômato $\mathcal{A}^{(2)}$ com os 3 estados $0^{(2)}$, $1^{(2)}$ e $2^{(2)}$, o autômato $\mathcal{A}^{(3)}$ também com o 3 estados $0^{(3)}$, $1^{(3)}$ e $2^{(3)}$ e o autômato $\mathcal{A}^{(4)}$ com os 2 estados $0^{(2)}$ e $1^{(2)}$. Conforme pode-se observar na Figura 4.15.

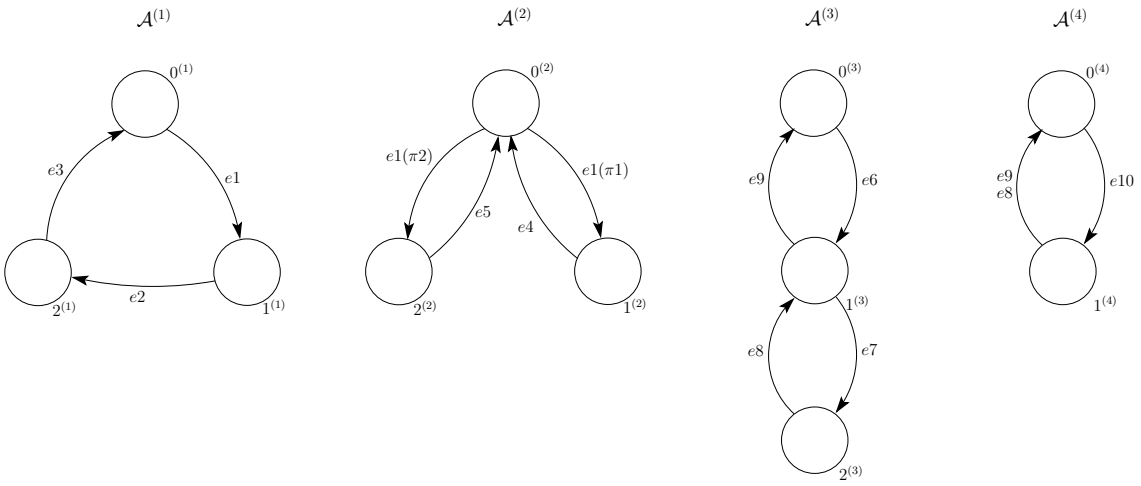

Figura 4.15: Exemplo de SAN com evento sincronizante com taxa funcional e probabilidades funcionais.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
syn	$e1$	f	loc	$e6$	1
loc	$e2$	1	loc	$e7$	1
loc	$e3$	1	syn	$e8$	1
loc	$e4$	1	syn	$e9$	1
loc	$e5$	1	loc	$e10$	1

Tabela 4.29: Taxas dos eventos da SAN representada na Figura 4.15

Na Tabela 4.29 são apresentados os eventos do modelo da Figura 4.15. Nota-se que os autômatos $\mathcal{A}^{(1)}$ e $\mathcal{A}^{(2)}$ possuem um evento sincronizante $e1$ com uma taxa funcional f que é representada por:

$$f = ((stA^{(3)} == 0^{(3)}) \times 5) + ((stA^{(3)} == 2^{(3)}) \times 3) \tag{4.18}$$

As probabilidades constantes $\pi 1$ e $\pi 2$ que definem a saída do estado $0^{(2)}$ do autômato $\mathcal{A}^{(2)}$ são:

$$\pi 1 = ((stA^{(4)} == 0^{(4)}) \times 0,7) + ((stA^{(4)} == 1^{(4)}) \times 0,3) \tag{4.19}$$

$$\pi 2 = ((stA^{(4)} == 0^{(4)}) \times 0,3) + ((stA^{(4)} == 1^{(4)}) \times 0,7) \tag{4.20}$$

Para converter o modelo SAN da Figura 4.15 para não utilizar taxas funcionais, o algoritmo irá percorrer todos os eventos procurando por algum evento ou probabilidade vinculada ao evento que seja do tipo funcional. Neste caso, somente o o evento $e1$ é do tipo funcional, portanto ele precisa ser substituído por novos eventos sincronizantes.

Para gerar os eventos sincronizantes, é necessário percorrer o espaço de estados definido por: $\hat{\mathcal{S}}^{(\zeta_e, w_{f_e} \cup w_{\pi_e})}$

Onde

$\zeta_e \quad = \{ \mathcal{A}^{(1)}, \mathcal{A}^{(2)} \}$

$w_{f_e} \quad = \{ \mathcal{A}^{(3)} \}$

$w_{\pi_e} \quad = \{ \mathcal{A}^{(4)} \}$

Neste ponto, percorre-se todo o espaço de estados dos autômatos definidos anteriormente. Para cada estado global $\tilde{x}$ deste espaço de estados, percorre-se os estados globais sucessores $\tilde{y}$ que possuem transição com o evento $e1$.

A cada estado global percorrido a função f é avaliada e o resultado, caso não seja zero, será a taxa constante de um novo evento sincronizante que será criado.

Por exemplo, avaliando a função para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)},0^{(4)})$ e o sucessor $\tilde{y}$ $(1^{(1)},1^{(2)},0^{(3)},0^{(4)})$, obtém-se então $e1(\pi 1) = 3,5$. Cria-se um novo evento chamado $e11$ cuja taxa constante vale 3,5.

Será criado uma nova transição dos estados $0^{(3)}$ e $0^{(4)}$ para eles mesmos com o novo evento $e11$ criado. Também para as transições que continham o evento $e1$ passarão a conter o evento $e11$.

O mesmo ocorre para os demais estados globais do subespaço de estados dos autômatos definido anteriormente. Na Tabela 4.30 pode-se verificar os estados globais e sua avaliação para a função f_e, são somente apresentados somente os valores que resultam diferente de zero.

$\tilde{x}$	$\tilde{y}$	Valor avaliado	Evento novo
$(0^{(1)},0^{(2)},0^{(3)},0^{(4)})$	$(1^{(1)},1^{(2)},0^{(3)},0^{(4)})$	3,5	$e11$
$(0^{(1)},0^{(2)},0^{(3)},0^{(4)})$	$(1^{(1)},2^{(2)},0^{(3)},0^{(4)})$	1,5	$e12$
$(0^{(1)},0^{(2)},0^{(3)},1^{(4)})$	$(1^{(1)},1^{(2)},0^{(3)},1^{(4)})$	1,5	$e13$
$(0^{(1)},0^{(2)},0^{(3)},1^{(4)})$	$(1^{(1)},2^{(2)},0^{(3)},1^{(4)})$	3,5	$e14$
$(0^{(1)},0^{(2)},2^{(3)},0^{(4)})$	$(1^{(1)},1^{(2)},2^{(3)},0^{(4)})$	2,1	$e15$
$(0^{(1)},0^{(2)},2^{(3)},0^{(4)})$	$(1^{(1)},2^{(2)},2^{(3)},0^{(4)})$	0,9	$e16$
$(0^{(1)},0^{(2)},2^{(3)},1^{(4)})$	$(1^{(1)},1^{(2)},2^{(3)},1^{(4)})$	0,9	$e17$
$(0^{(1)},0^{(2)},2^{(3)},1^{(4)})$	$(1^{(1)},2^{(2)},2^{(3)},1^{(4)})$	2,1	$e18$

Tabela 4.30: Resultado da avaliação da função f para o espaço de estados definido nela.

Cada estado do espaço de estados avaliado pela função deve receber uma transição para ele mesmo com o novo evento criado. Por exemplo, para o estado global $\tilde{x}$ $(0^{(1)},0^{(2)},0^{(3)},0^{(4)})$. Como a função utilizada neste modelo faz referência ao autômato $\mathcal{A}^{(3)}$, o estado $0^{(3)}$ irá receber uma transição para ele mesmo com os eventos $e11$ e $e12$. Como a probabilidade $\pi1$ faz referência ao autômato $\mathcal{A}^{(4)}$, o estado $0^{(4)}$ irá receber também uma transição para ele mesmo com os eventos $e11$ e $e12$.

A transição do estado onde o evento $e1$ estava originalmente, nos autômatos $\mathcal{A}^{(1)}$ e $\mathcal{A}^{(2)}$, também recebem os novos eventos criados.

Na Figura 4.16 é representado o modelo SAN da Figura 4.15 convertido para não utilizar taxas funcionais.

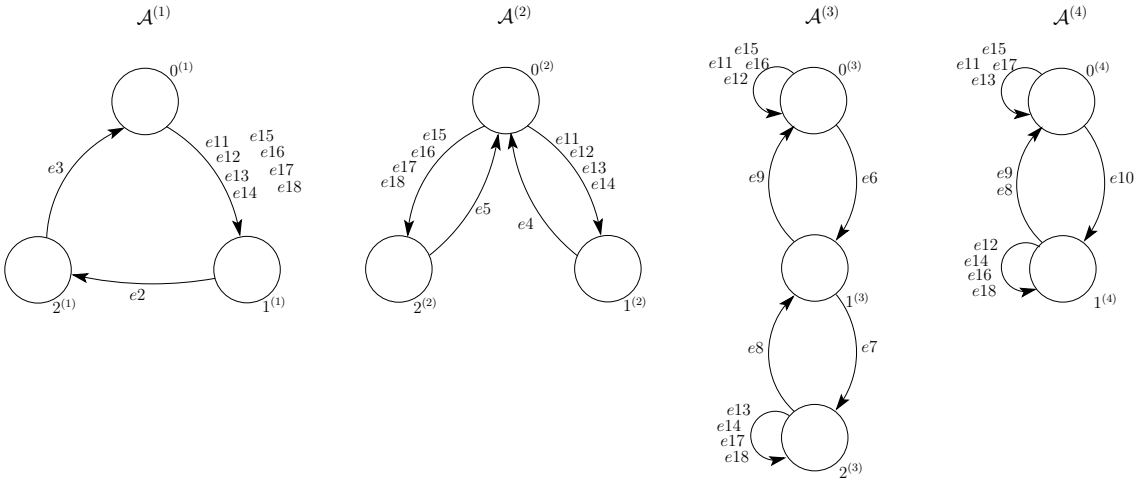

Figura 4.16: Exemplo do SAN da Figura 4.15 convertido para não mais usar funções.

Tando o modelo SAN original (Figura 4.15) quanto o modelo convertido (Figura 4.16) foram resolvidos na ferramenta PEPS e provou-se que eles são equivalentes.

Tipo	Evento	Taxa	Tipo	Evento	Taxa
loc	$e2$	1	loc	$e8$	1
loc	$e3$	1	loc	$e9$	1
loc	$e4$	1	syn	$e10$	3,5
loc	$e5$	1	syn	$e11$	1,5
loc	$e6$	1	syn	$e12$	2,1
loc	$e7$	1	syn	$e13$	0,9

Tabela 4.31: Taxas dos eventos da SAN representada na Figura 4.16

Na Tabela 4.32 pode-se observar que para todos os estados globais da rede SAN as probabilidades são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

Estado global	Modelo GTA (Figura 4.15)	Modelo CTA (Figura 4.16)	Estado global	Modelo GTA (Figura 4.15)	Modelo CTA (Figura 4.16)
$0^{(1)}0^{(2)}0^{(3)}$	0.0378668058	0.0378668058	$1^{(1)}2^{(2)}0^{(3)}$	0.0224984241	0.0224984241
$0^{(1)}0^{(2)}1^{(3)}$	0.1042807192	0.1042807192	$1^{(1)}2^{(2)}1^{(3)}$	0.0106950636	0.0106950636
$0^{(1)}0^{(2)}2^{(3)}$	0.0557226979	0.0557226979	$1^{(1)}2^{(2)}2^{(3)}$	0.0202818306	0.0202818306
$0^{(1)}1^{(2)}0^{(3)}$	0.0216428807	0.0216428807	$2^{(1)}0^{(2)}0^{(3)}$	0.0920017143	0.0920017143
$0^{(1)}1^{(2)}1^{(3)}$	0.0199641190	0.0199641190	$2^{(1)}0^{(2)}1^{(3)}$	0.0864517643	0.0864517643
$0^{(1)}1^{(2)}2^{(3)}$	0.0207808721	0.0207808721	$2^{(1)}0^{(2)}2^{(3)}$	0.0889231122	0.0889231122
$0^{(1)}2^{(2)}0^{(3)}$	0.0092755205	0.0092755205	$2^{(1)}1^{(2)}0^{(3)}$	0.0233216423	0.0233216423
$0^{(1)}2^{(2)}1^{(3)}$	0.0085560512	0.0085560512	$2^{(1)}1^{(2)}1^{(3)}$	0.0174686040	0.0174686040
$0^{(1)}2^{(2)}2^{(3)}$	0.0089060883	0.0089060883	$2^{(1)}1^{(2)}2^{(3)}$	0.0215976251	0.0215976251
$1^{(1)}0^{(2)}0^{(3)}$	0.0642350327	0.0642350327	$2^{(1)}2^{(2)}0^{(3)}$	0.0099949895	0.0099949895
$1^{(1)}0^{(2)}1^{(3)}$	0.0534753184	0.0534753184	$2^{(1)}2^{(2)}1^{(3)}$	0.0074865445	0.0074865445
$1^{(1)}0^{(2)}2^{(3)}$	0.0605407102	0.0605407102	$2^{(1)}2^{(2)}2^{(3)}$	0.0092561250	0.0092561250
$1^{(1)}1^{(2)}0^{(3)}$	0.0524963229	0.0524963229			
$1^{(1)}1^{(2)}1^{(3)}$	0.0249551486	0.0249551486			
$1^{(1)}1^{(2)}2^{(3)}$	0.0473242714	0.0473242714			

Tabela 4.32: Resultados dos modelos SAN das Figuras 4.15 e 4.16

No Anexo B demonstra-se a definição dos modelos.

4.3 Implementação do algoritmo

A conversão automática de modelos SAN proposto no algoritmo deste capítulo foi implementada na ferramenta PEPS. Ou seja, a partir de um modelo SAN, é feita uma análise dos eventos definidos por taxas ou probabilidades funcionais e, se necessário, estes eventos são transformados em eventos sincronizantes equivalente, gerando por fim um modelo que somente utiliza diretivas de álgebra tensorial clássica.

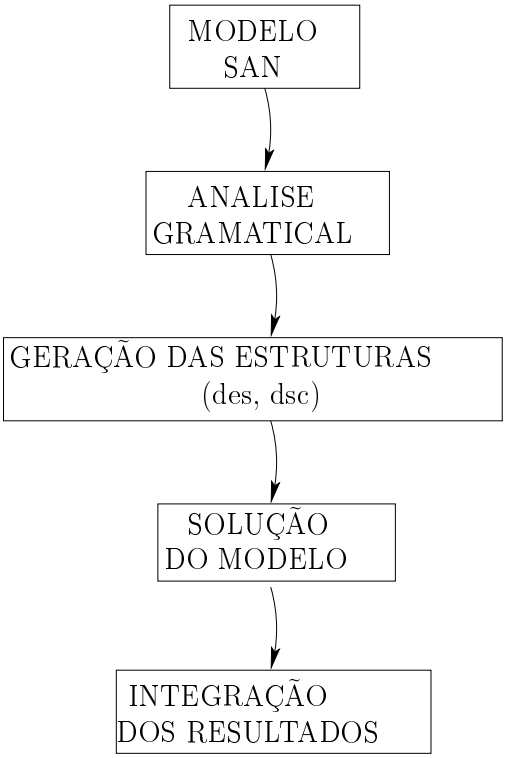

Figura 4.17: Funcionamento de execução da ferramenta PEPS.

O ciclo de execução da ferramenta PEPS pode ser observado na Figura 4.17. Basicamente consiste na leitura de um arquivo com a extensão *.san* que representa textualmente o modelo SAN. Após a leitura deste arquivo a ferramenta analisa gramaticalmente o modelo e armazena as informações em memória. Neste ponto são gerados os arquivos nos diretórios[1] *des* e *dsc*. A solução e os resultados são executados nos 2 últimos passos.

A implementação do algoritmo que converte o modelo SAN para não utilizar mais funções é feita após a análise gramatical do arquivo do modelo SAN, neste ponto o algoritmo modifica a estrutura do SAN e libera o PEPS para solucionar o modelo. A Figura 4.18 representa o ciclo de execução da ferramenta PEPS incluindo o passo do conversor.

[1]O leitor interessado na descrição dos arquivos e diretórios utilizados pela ferramenta PEPS podem encontrar informações em *http://www-id.imag.fr/Logiciels/peps/*.

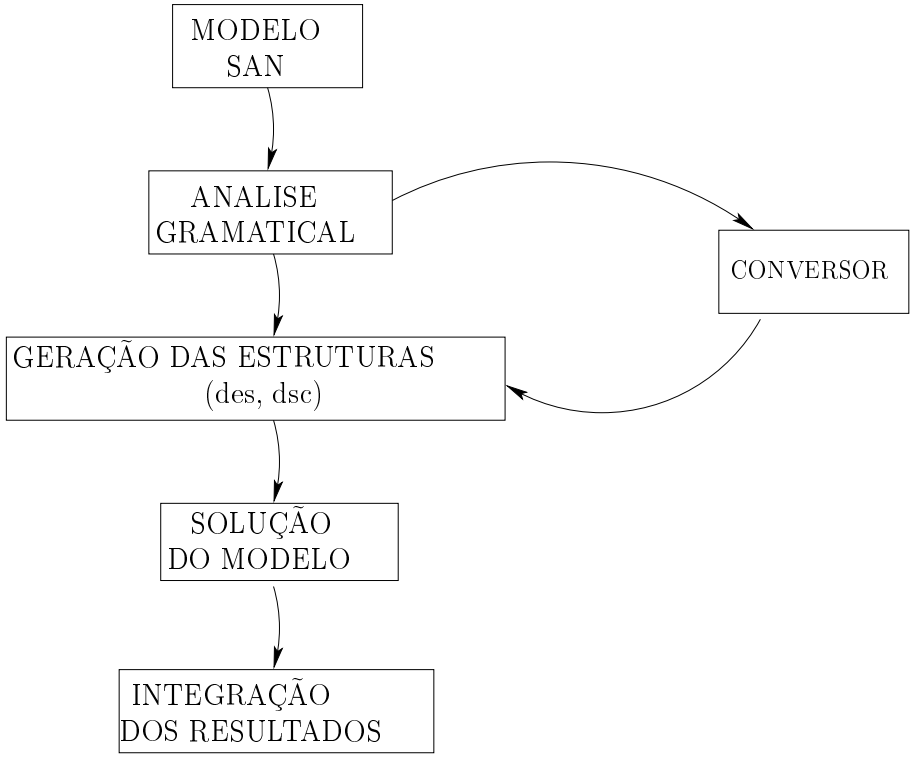

Figura 4.18: Funcionamento de execução da ferramenta PEPS incluindo o algoritmo do conversor.

4.3.1 Teste da implementação do algoritmo

Para validar o funcionamento do algoritmo, iremos utilizar o modelo SAN que descreve o padrão de mobilidade *Random Waypoint* com 1 dimensão. O modelo possui 2 autômatos, um responsável pelo deslocamento do nodo em uma área de 5 espaços. O outro autômato é o responsável por decidir para qual espaço o nodo deve se deslocar.

A Figura 4.19 representa o padrão de mobilidade *Random Waypoint*, na Tabela 4.33 são apresentados os eventos utilizados pelo modelo.

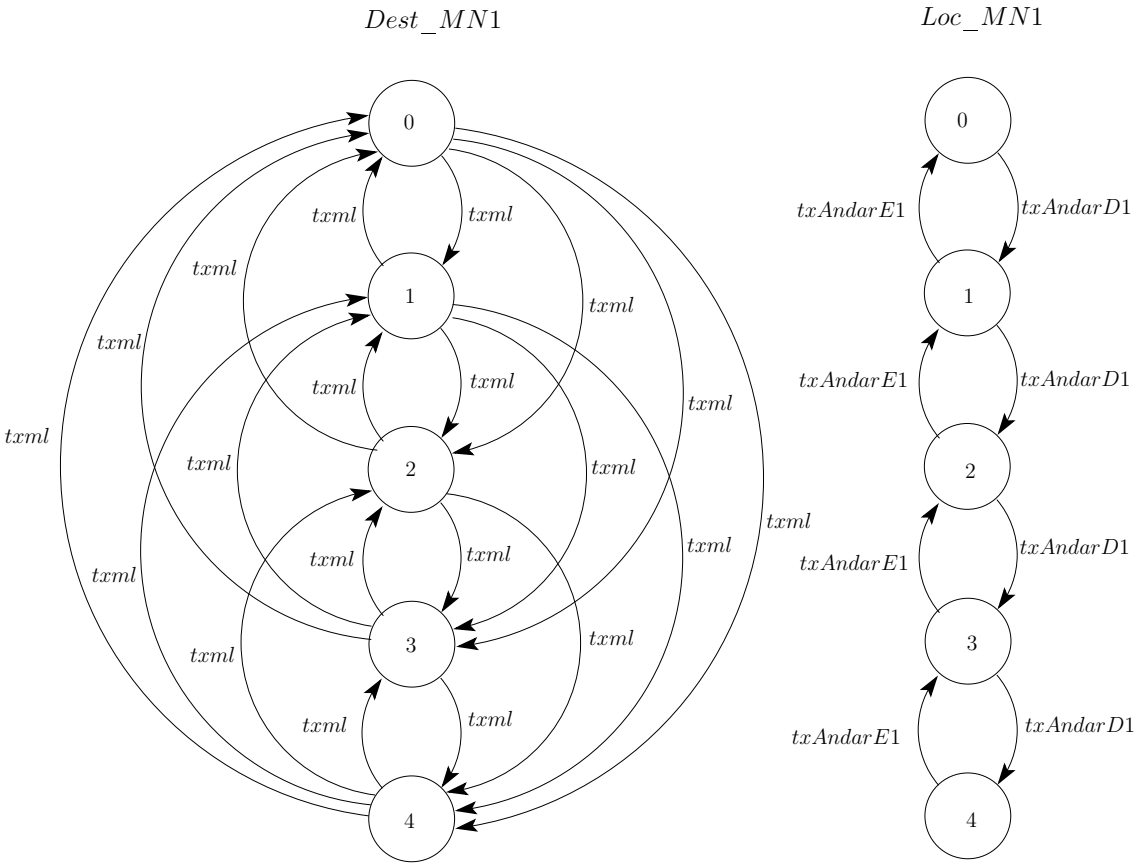

Figura 4.19: Padrão de mobilidade *Random Waypoint* com taxas funcionais.

Tipo	Evento	Taxa
loc	$txml$	$TaxaDecDest1$
loc	$txAndarD1$	$TaxaAndarDir1$
loc	$txAndarE1$	$TaxaAndarEsq1$

Tabela 4.33: Eventos definidos na figura 4.19

As funções citadas na Tabela 4.33 são descritas à seguir.

$$TaxaDecDest1 = ((stDest\_MN1) == (stLoc\_MN1)) * (1/(Parada1 * (NumEspacos - 1))); \quad (4.21)$$

Onde $Parada = 0,1$ e $NumEspacos = 5$.

$$TaxaAndarDir1 = TaxaVel1 * (stDest\_MN1 > stLoc\_MN1); \quad (4.22)$$

$$TaxaAndarEsq1 = TaxaVel1 * (stDest\_MN1 < stLoc\_MN1); \quad (4.23)$$

Onde $TaxaVel = 1$.

As caractersísticas deste modelo correspondem ao exemplo estudado na seção 4.2.1 (Rede SAN com evento local com taxa funcional sem especificação de probabilidades). O modelo SAN convertido utilizando o algoritmo descrito neste capítulo pode ser observado nas Figuras 4.20 e 4.21. Os eventos do modelo estão divididos em 2 figuras para evitar poluição visual do modelo e facilitar o entendimento do mesmo. Na Figura 4.20 somente ão apresentados as transições entre os estados dos autômatos, já na Figura 4.21 são apresentadas somente as transições que levam dos estados para eles mesmos.

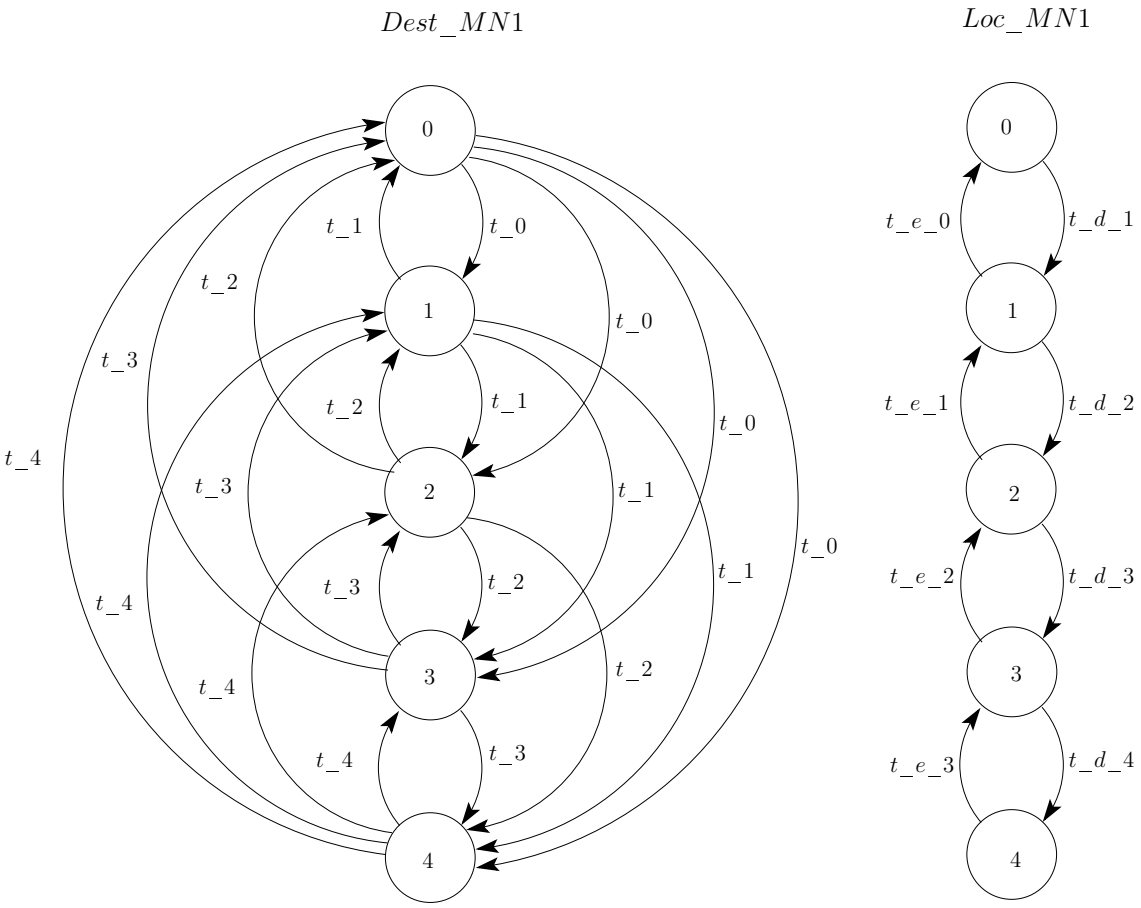

Figura 4.20: Padrão de mobilidade *Random Waypoint* convertido para CTA - parte 1.

Na Tabela 4.34, são demonstrados os eventos sincronizantes criados pelo conversor e as taxas de cada um.

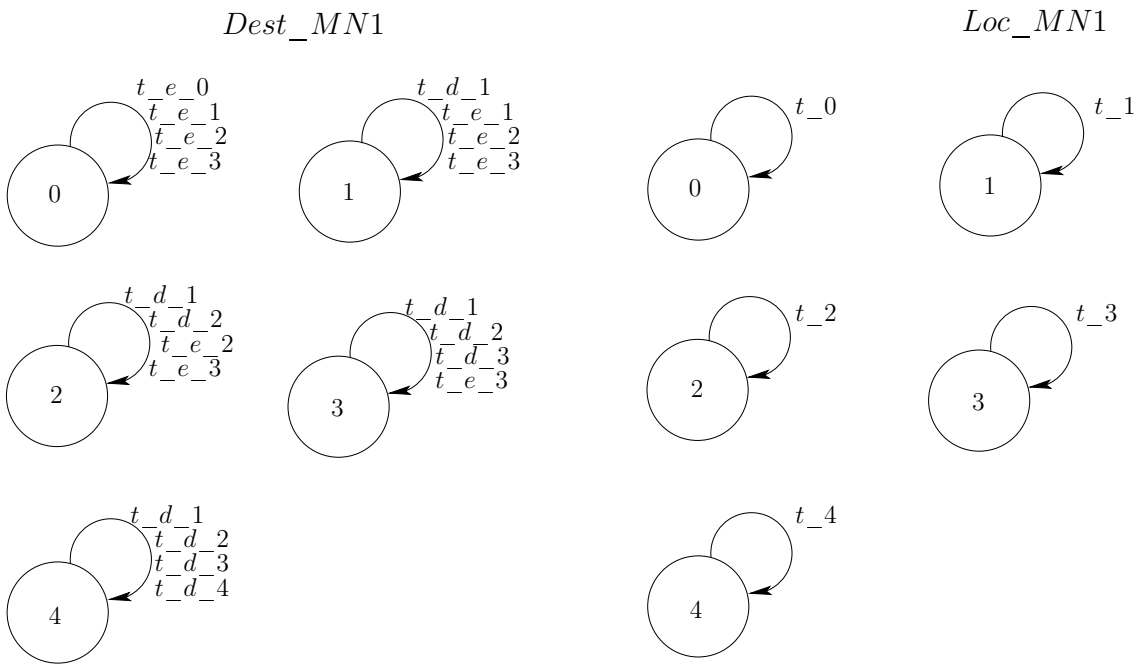

Figura 4.21: Padrão de mobilidade *Random Waypoint* convertido para CTA - parte 2.

Tipo	Evento	Taxa
syn	t_0	2,5
syn	t_1	2,5
syn	t_2	2,5
syn	t_3	2,5
syn	t_4	2,5
syn	t_d_1	1
syn	t_d_2	1
syn	t_d_3	1
syn	t_d_4	1
syn	t_e_0	1
syn	t_e_1	1
syn	t_e_2	1
syn	t_e_3	1

Tabela 4.34: Eventos definidos nas Figuras 4.20 e 4.21

Tanto o modelo SAN original quanto o modelo convertido foram resolvidos na ferramenta PEPS (o primeiro na ferramenta original e o segundo na ferramenta modificada com o conversor), obtendo-se o mesmo resultado. Na Tabela 4.35 pode-se observar que para todos os estados globais da rede SAN as probabilidades do modelo GTA e do modelo CTA são iguais. Desta forma, demonstra-se os resultados mostrando que os modelos são equivalentes dentro da precisão solicitada, 10^{-10}.

Estado global	Modelo GTA	Modelo CTA
$(stLoc_M N1 == 0)$	1.0012486103	1.0012486103
$(stLoc_M N1 == 1)$	2.4993757799	2.4993757799
$(stLoc_M N1 == 2)$	2.9987512195	2.9987512195
$(stLoc_M N1 == 3)$	2.4993757799	2.4993757799
$(stLoc_M N1 == 4)$	1.0012486103	1.0012486103

Tabela 4.35: Resultados gerados pelo modelo SAN original (GTA) e pelo modelo SAN convertido para CTA.

5. CONSIDERAÇÕES FINAIS E PERSPECTIVAS

Esta dissertação apresentou formalismo estruturado para modelagem de sistemas denominado SAN (Redes de Autômatos Estocásticos) e os conceitos de álgebra tensorial clássica e generalizada utilizados por este formalimo.

Um modelo SAN é descrito através de um descritor Markoviano e este, por sua vez, utiliza álgebra tensorial na sua construção. Embora modelos SAN descritos em GTA possuam uma representação compacta e mais eficiente em relação a um modelo equivalente descrito em CTA, dependendo do modelo a avaliação das funções em tempo de solução pode se tornar bastante custosa. Para estes casos, pode ser mais vantajosa a utilização de eventos sincronizantes em relação à utilização de funções.

O objetivo deste trabalho era propor um método de conversão automática de modelos SAN que são descritos utilizando primitivas funcionais na sua definição por modelos equivalentes, porém que somente utilizem eventos locais e sincronizantes com taxas constantes.

O método definido neste trabalho preocupou-se em minimizar o número de eventos sincronizantes criados através do reaproveitamento de eventos cuja avaliação tenha sido igual. Esta otimização faz-se nescessária para não gerar eventos demais durante a conversão dos modelos.

Foram mostrados os passos necessários para se converter um modelo SAN com primitivas funcionais (descritos em GTA) por um modelo SAN com eventos e taxas constantes (descritos em CTA) através de um algoritmo conceitual.

Para validar o algoritmo apresentado neste trabalho, foram utilizados modelos que abrangem todas as características possíveis. Qualquer exemplo de um sistema certamente possuirá características que se enquadram em uma das possibilidades testadas. Por fim o algoritmo foi implementado na ferramenta PEPS e validado utilizando um modelo real para garantir o funcionamento do algoritmo.

Como trabalhos futuros, pode-se detalhar as vantagens e desvantagens de transformar modelos que utilizam funções em modelos equivalentes que não utilizam através da eficiência do algoritmo e principalmente a qualidade e as virtudes do modelo traduzido, ou seja, analisar para diferentes famílias de casos práticos se o modelo sem função ocupa menos memória ou é mais rápido que o modelo com função.

O esforço computacional para executar o algoritmo depende de muitos fatores, pois um modelo pode ter diversos eventos com taxas e probabilidades funcionais. Além disto, cada função pode avaliar diversas situações para diversos autômatos. Apesar de termos todas estas variações de modelos diferentes o esforço computacional do algoritmo é um tema que pode ser estudado em trabalhos futuros.

Bibliografia

[1] V. Amoia, G. De Micheli, and M. Santomauro. Computer-oriented formulation of transition-rate matrices via kronecker algebra. *IEEE Transactions on Reliability*, 30(2):123–132, 1981.

[2] R. Bellman. Introduction to matrix analysis. *McGraw-Hill*, 1960.

[3] A. Benoit, L. Brenner, P. Fernandes, B. Plateau, and W. J. Stewart. The PEPS software tool. *In Computer Performance Evaluation / TOOLS 2003*, 2794:98–115, 2003.

[4] L. Brenner, P. Fernandes, and A. Sales. Why you should care about Generalized Tensor Algebra. (TR 037), November 2003. http://www3.pucrs.br/pucrs/files/uni/poa/facin/pos/relatoriostec/tr037.pdf.

[5] L. Brenner, P. Fernandes, and A. Sales. The Need for and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. *International Journal of Simulation: Systems, Science & Technology (IJSIM)*, 6(3-4):52–60, February 2005.

[6] J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, CAS-25(9):772–780, 1978.

[7] R. M. Czekster. *Solução numérica de descritores Markovianos a partir de re-estruturações de termos tensoriais*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Brasil, May 2010.

[8] M. Davio. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, C-30(2):116–125, 1981.

[9] F. L. Dotti, P. Fernandes, and Cristina M. Nunes. Structured markovian models for discrete spatial mobile node distribution. *J. Braz. Comp. Soc.*, 17(1):31–52, 2011.

[10] P. Fernandes. *Méthodes numériques pour la solution de systèmes Markoviens à grand espace d'États*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1998.

[11] L. Kronecker. Uber einige interpolationsformeln fur ganze funktionen mehrerer variabeln. *L. Kroneckers Werke*, I:133–141, 1865.

[12] B. Plateau. *De lÉvaluation du Parallélisme et de la Synchronisation*. PhD thesis, Paris-Sud, 1984.

[13] B. Plateau and K. Atif. Stochastic automata networks for modelling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.

[14] A. Sales. Formalismos Estruturados de Modelagem para Sistemas Markovianos Complexos. Master's thesis, PUCRS-FACIN-PPGCC, Porto Alegre, December 2003.

A. NOTAÇÃO DEFINIDA NO CAPÍTULO 3

Este apêndice mostrará o resumo da notação definida no Capítulo 3.

Sejam

$\mathcal{A}$ o conjunto de autômatos que compreende N autômatos nomeados $\mathcal{A}^{(i)}$, onde $i \in [1..N]$ e $N = |\mathcal{A}|$;

$\mathcal{S}^{(i)}$ o conjunto de estados locais do autômato $\mathcal{A}^{(i)}$;

$x^{(i)}$ é um estado local do autômato $\mathcal{A}^{(i)}$, onde $x^{(i)} \in \mathcal{S}^{(i)}$;

$\hat{\mathcal{S}}$ é o espaço de estados produto do modelo, onde $\hat{\mathcal{S}} = \mathcal{S}^{(1)} \times \mathcal{S}^{(2)} \times ... \times \mathcal{S}^{(N)}$;

$\tilde{x}$ é o estado global do modelo o qual é uma composição dos estados locais do autômato, isto é, $\tilde{x} = (x^{(1)}, .., x^{(N)})$, onde $\tilde{x} \in \hat{\mathcal{S}}$;

$\mathcal{E}$ o conjunto de eventos que compreende E eventos, onde $\mathcal{E} = \{e_1, e_2, ..., e_E\}$ e $E = |\mathcal{E}|$;

w é um conjunto de índices de autômatos (w é um subconjunto $[1..N]$);

$\tilde{x}^{(w)}$ composição dos estados locais $x^{(i)}$, onde $i \in w$;

$\hat{\mathcal{S}}^{(w)}$ é o espaço de estados produto do conjunto de estados locais do autômato $\mathcal{A}^{(i)}$ onde $i \in w$ e $\hat{\mathcal{S}}^{(w)} \subseteq \hat{\mathcal{S}}$;

e um evento, onde $e \in \mathcal{E}$;

ζ_e é o conjunto de índices dos autômatos onde o evento e aparece;

(e, τ_e) uma tupla de evento onde τ_e é um elemento funcional de $\hat{\mathcal{S}} \to \mathbb{R}^+$, o qual define a taxa de ocorrência do evento e;

(e, π_e) uma tupla de transição onde π_e é uma probabilidade funcional de $\hat{\mathcal{S}} \to \mathbb{R}^+$;

$\mathcal{T}$ é um conjunto que contém todas as tuplas de transição (e, π_e), onde π_e é um elemento funcional definido de $\hat{\mathcal{S}}$, o qual representa a probabilidade de uma transição quando a ocorrência do evento e;

$\mathcal{T}^* = \mathcal{T} \bigcup \{\emptyset\}$;

$\mathcal{Q}^{(i)}$ é uma função de transição de $\mathcal{S}^{(i)} \times \mathcal{S}^{(i)} \to \mathcal{T}^*$ que contém as tuplas de transisão do autômato $\mathcal{A}^{(i)}$;

$\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$ rótulo de transição do estado local $x^{(i)}$ para o estado $y^{(i)}$, contendo uma lista de tuplas de transição (e, π) em $\mathcal{T}$;

$\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$ rótulo de transição do estado global $\tilde{x}$ para $\tilde{y}$, contendo uma lista de tuplas de transição (e, π) em $\mathcal{T}$;

$\pi_e(x^{(i)}, y^{(i)})$ probabilidade da tupla de transição (e, π) associada ao rótulo de transição $\mathcal{Q}^{(i)}(x^{(i)}, y^{(i)})$;

$\pi_e(\tilde{x}, \tilde{y})$ probabilidade da transição global de $\tilde{x}$ para $\tilde{y}$ calculada pelo produtório de todas as probabilidades de rotação locais, ou seja, $\prod_{i \in \zeta_e, \exists (e,\pi) \in Q^{(i)}(x^{(i)}, y^{(i)})} \pi_e(x^{(i)}, y^{(i)})$;

$succ_e(\tilde{x})$ conjunto dos estados globais sucessores $\tilde{y}$ tais que o rótulo $\tilde{\mathcal{Q}}(\tilde{x}, \tilde{y})$ possua uma tupla de transição com o identificador e e $\tau_e(\tilde{x}) \neq 0$ e $\pi_e(\tilde{x}, \tilde{y}) \neq 0$. O conjunto de estados sucessores de e em $\tilde{x}$ pode ser vazio, caso a transição não possa ser disparada em $\tilde{x}$ pelo evento e.

$\mathcal{F}$ o conjunto de funções do modelo;

$f(\hat{\mathcal{S}}^{(w)})$ uma função de $\hat{\mathcal{S}}^{(w)} \rightarrow \mathbb{R}^+$, onde $w \subseteq [1..N]$;

f_e é uma função que corresponde ao elemento funcional τ_e associada a dupla do evento e, isto é, a tupla de evento (e, τ_e);

w_{f_e} é o conjunto dos índices dos autômatos os quais seu espaço de estados são o domínio da função $f_e \in \mathcal{F}$ do evento e;

w_{π_e} é o conjunto dos índices dos autômatos os quais seu espaço de estados são o domínio da probabilidade π_e do evento e;

$f_e(\tilde{x})$ é o resultado da avaliação da função f_e para o estado global $\tilde{x}$;

$\pi_e(\tilde{x})$ é o resultado da avaliação da probabilidade π_e para o estado global $\tilde{x}$;

B. MODELOS UTILIZADOS PARA A VALIDAÇÃO

Este apêndice mostrará os modelos utilizados para validação do algoritmo.

B.1 Rede SAN com evento local com taxa funcional sem especificação de probabilidades construído utilizando primitivas GTA

```
identifiers

l1= ( ( (st A2 == est0) * 3 ) + ( ((st A3 == est0)*4) + ((st A3 == est2)*5) ) );
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;

events

loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_loc_func_f (continuous)

  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
```

```
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e4
    stt est1 to (est0) e5

  aut A3
    stt est0 to (est1) e6
    stt est1 to (est2) e7 to (est0) e9
    stt est2 to (est1) e8

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
```

B.2 Rede SAN com evento local com taxa funcional sem especificação de probabilidades convertido utilizando somente primitivas CTA

```
identifiers

l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;

l10=7;
l11=3;
l12=8;
l13=4;
l14=5;

events

//loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
syn e10 (l10);
syn e11 (l11);
syn e12 (l12);
syn e13 (l13);
syn e14 (l14);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));
```

```
network ev_loc_func_c (continuous)

  aut A1
    stt est0 to (est1) e10 e11 e12 e13 e14
    stt est1 to (est2) e2
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est0) e10 e11 e12 to (est1) e4
    stt est1 to (est0) e5 to (est1) e13 e14

  aut A3
    stt est0 to (est0) e10 e13 to (est1) e6
    stt est1 to (est2) e7 to (est0) e9 to (est1) e11
    stt est2 to (est1) e8 to (est2) e12 e14

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
```

B.3 Rede SAN com evento local com taxa funcional e probabilidades constantes construído utilizando primitivas GTA

```
identifiers

l1=1;
l2=1;
l3=1;
l4=1;
l5=1;
l6=( ((st A3 == est0)*2) + ((st A3 == est2)*5) );
l7=1;
l8=1;
l9=1;
l10=1;

pi1=(1/2);
pi2=(1/2);

events

loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
loc e10(l10);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_loc_func_pro_cons_f (continuous)

  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
```

```
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e6(pi1) to (est2) e6(pi2)
    stt est1 to (est0) e4
    stt est2 to (est0) e5

  aut A3
    stt est0 to (est1) e7
    stt est1 to (est2) e8 to (est0) e10
    stt est2 to (est1) e9

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_0_a2_2_a3_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_a1_0_a2_2_a3_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_a1_0_a2_2_a3_2 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_2_a3_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_a1_1_a2_2_a3_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_a1_1_a2_2_a3_2 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_2_a3_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
```

```
estar_a1_2_a2_2_a3_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
estar_a1_2_a2_2_a3_2 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2);
```

B.4 Rede SAN com evento local com taxa funcional e probabilidades constantes convertido utilizando somente primitivas CTA

```
identifiers
l1=1;
l2=1;
l3=1;
l4=1;
l5=1;
l7=1;
l8=1;
l9=1;
l10=1;
l11=1;
l12=2.5;

events

loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
loc e10(l10);
syn e11(l11);
syn e12(l12);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_loc_func_pro_cons_c (continuous)

  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
    stt est2 to (est0) e3
```

```
  aut A2
    stt est0 to (est1) e11 e12 to (est2) e11 e12
    stt est1 to (est0) e4
    stt est2 to (est0) e5

  aut A3
    stt est0 to (est1) e7 to (est0) e11
    stt est1 to (est2) e8 to (est0) e10
    stt est2 to (est1) e9 to (est2) e12

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_0_a2_2_a3_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_a1_0_a2_2_a3_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_a1_0_a2_2_a3_2 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_2_a3_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_a1_1_a2_2_a3_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_a1_1_a2_2_a3_2 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_2_a3_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_a1_2_a2_2_a3_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
estar_a1_2_a2_2_a3_2 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2);
```

B.5 Rede SAN com evento local com taxa constante e probabilidades funcionais construído utilizando primitivas GTA

```
identifiers

l1=9;
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;

pi1=( ((st A2 == est0)*0.6) + ((st A2 == est1)*0.2) + ((st A2 == est2)*0.2) );
pi2=( ((st A2 == est0)*0.2) + ((st A2 == est1)*0.6) + ((st A2 == est2)*0.2) );
pi3=( ((st A2 == est0)*0.2) + ((st A2 == est1)*0.2) + ((st A2 == est2)*0.6) );

events

loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);

partial reachability = ((st A1 == est0) && (st A2 == est0));

network ev_loc_cons_pro_func_f (continuous)

  aut A1
    stt est0 to (est1) e1(pi1) to (est2) e1(pi2) to (est3) e1(pi3)
    stt est1 to (est0) e2
    stt est2 to (est0) e3
    stt est3 to (est0) e4

  aut A2
    stt est0 to (est1) e5
```

```
    stt est1 to (est2) e6
    stt est2 to (est0) e7

results

estar_a1_0_a2_0 = (st A1 == est0) && (st A2 == est0);
estar_a1_0_a2_1 = (st A1 == est0) && (st A2 == est1);
estar_a1_0_a2_2 = (st A1 == est0) && (st A2 == est2);
estar_a1_1_a2_0 = (st A1 == est1) && (st A2 == est0);
estar_a1_1_a2_1 = (st A1 == est1) && (st A2 == est1);
estar_a1_1_a2_2 = (st A1 == est1) && (st A2 == est2);
estar_a1_2_a2_0 = (st A1 == est2) && (st A2 == est0);
estar_a1_2_a2_1 = (st A1 == est2) && (st A2 == est1);
estar_a1_2_a2_2 = (st A1 == est2) && (st A2 == est2);
estar_a1_3_a2_0 = (st A1 == est3) && (st A2 == est0);
estar_a1_3_a2_1 = (st A1 == est3) && (st A2 == est1);
estar_a1_3_a2_2 = (st A1 == est3) && (st A2 == est2);
```

B.6 Rede SAN com evento local com taxa constante e probabilidades funcionais convertido utilizando somente primitivas CTA

```
identifiers

l1=9;
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=5.4;
l9=1.8;
l10=1.8;
l11=5.4;
l12=1.8;
l13=5.4;

events

loc e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
loc e10 (l10);
loc e11 (l11);
loc e12 (l12);
loc e13 (l13);

partial reachability = ((st A1 == est0) && (st A2 == est0));

network ev_loc_cons_pro_func_c (continuous)
```

```
  aut A1
    stt est0 to (est1) e8 e9 to (est2) e10 e11 to (est3) e12 e13
    stt est1 to (est0) e2
    stt est2 to (est0) e3
    stt est3 to (est0) e4

  aut A2
    stt est0 to (est1) e5 to (est0) e8 e10 e12
    stt est1 to (est2) e6 to (est1) e9 e11 e12
    stt est2 to (est0) e7 to (est2) e9 e10 e13

results

estar_a1_0_a2_0 = (st A1 == est0) && (st A2 == est0);
estar_a1_0_a2_1 = (st A1 == est0) && (st A2 == est1);
estar_a1_0_a2_2 = (st A1 == est0) && (st A2 == est2);
estar_a1_1_a2_0 = (st A1 == est1) && (st A2 == est0);
estar_a1_1_a2_1 = (st A1 == est1) && (st A2 == est1);
estar_a1_1_a2_2 = (st A1 == est1) && (st A2 == est2);
estar_a1_2_a2_0 = (st A1 == est2) && (st A2 == est0);
estar_a1_2_a2_1 = (st A1 == est2) && (st A2 == est1);
estar_a1_2_a2_2 = (st A1 == est2) && (st A2 == est2);
estar_a1_3_a2_0 = (st A1 == est3) && (st A2 == est0);
estar_a1_3_a2_1 = (st A1 == est3) && (st A2 == est1);
estar_a1_3_a2_2 = (st A1 == est3) && (st A2 == est2);
```

B.7 Rede SAN com evento local com taxa funcional e probabilidades funcionais construído utilizando primitivas GTA

```
identifiers

l1=( ((st A3 == est0)*6) + ((st A3 == est1)*4) ) ;
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=1;

pi1=( ((st A2 == est0)*0.8) + ((st A2 == est2)*0.2) );
pi2=( ((st A2 == est0)*0.2) + ((st A2 == est2)*0.8) );

events

loc e1 (l1);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
loc e10 (l10);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_loc_func_pro_func_f (continuous)

  aut A1
    stt est0 to (est1) e1(pi1) to (est2) e1(pi2)
    stt est1 to (est0) e4
    stt est2 to (est0) e3
```

```
  aut A2
    stt est0 to (est1) e5
    stt est1 to (est2) e6 to (est0) e8
    stt est2 to (est1) e7

  aut A3
    stt est0 to (est1) e9
    stt est1 to (est0) e10

results

estar_0_0_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_0_0_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_0_1_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_0_1_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_0_2_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_0_2_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_1_0_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_1_0_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_1_1_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_1_1_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_1_2_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_1_2_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_2_0_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_2_0_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_2_1_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_2_1_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_2_2_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_2_2_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
```

B.8 Rede SAN com evento local com taxa funcional e probabilidades funcionais convertido utilizando somente primitivas CTA

```
identifiers
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=1;
l11=4.8;
l12=1.2;
l13=3.2;
l14=0.8;
l15=1.2;
l16=4.8;
l17=0.8;
l18=3.2;

events

loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
loc e10 (l10);
syn e11 (l11);
syn e12 (l12);
syn e13 (l13);
syn e14 (l14);
syn e15 (l15);
syn e16 (l16);
syn e17 (l17);
```

```
syn e18 (l18);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_loc_func_pro_func_c (continuous)

  aut A1
    stt est0 to (est1) e11 e13 e15 e17 to (est2) e12 e14 e16 e18
    stt est1 to (est0) e4
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e5 to (est0) e11 e12 e13 e14
    stt est1 to (est2) e6 to (est0) e8
    stt est2 to (est1) e7 to (est2) e15 e16 e17 e18

  aut A3
    stt est0 to (est1) e9 to (est0) e11 e12 e15 e16
    stt est1 to (est0) e10 to (est1) e13 e14 e17 e18

results

estar_0_0_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_0_0_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_0_1_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_0_1_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_0_2_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_0_2_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_1_0_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_1_0_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_1_1_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_1_1_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_1_2_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_1_2_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_2_0_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_2_0_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_2_1_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_2_1_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_2_2_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
```

```
estar_2_2_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
```

B.9 Rede SAN com evento sincronizante com taxa funcional sem especificação de probabilidades construído utilizando primitivas GTA

```
identifiers
l1= ( ((st A3 == est0)*4) + ((st A3 == est2)*5) );
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;

events

syn e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_func_f (continuous)

  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e4
    stt est1 to (est0) e1
```

```
  aut A3
    stt est0 to (est1) e6
    stt est1 to (est2) e7 to (est0) e9
    stt est2 to (est1) e8

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
```

B.10 Rede SAN com evento sincronizante com taxa funcional sem especificação de probabilidades convertido utilizando somente primitivas CTA

```
identifiers
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=4;
l11=5;

events

loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);
syn e10 (l10);
syn e11 (l11);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_func_c (continuous)

  aut A1
    stt est0 to (est1) e10 e11
    stt est1 to (est2) e2
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e4
```

```
    stt est1 to (est0) e10 e11

  aut A3
    stt est0 to (est1) e6 to (est0) e10
    stt est1 to (est2) e7 to (est0) e9
    stt est2 to (est1) e8 to (est2) e11

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
```

B.11 Rede SAN com evento sincronizante com taxa funcional e probabilidades constantes construído utilizando primitivas GTA

```
identifiers
l1=( ((st A3 == est0)*5) + ((st A3 == est2)*3) );
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;

pi1=0.7;
pi2=0.3;

events

syn e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
loc e9 (l9);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_func_pro_cons_f (continuous)

  aut A2
    stt est0 to (est1) e1(pi1)
             to (est2) e1(pi2)
    stt est1 to (est0) e4
    stt est2 to (est0) e5
```

```
  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
    stt est2 to (est0) e3

  aut A3
    stt est0 to (est1) e6
    stt est1 to (est2) e7
             to (est0) e9
    stt est2 to (est1) e8

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_0_a2_2_a3_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_a1_0_a2_2_a3_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_a1_0_a2_2_a3_2 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_2_a3_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_a1_1_a2_2_a3_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_a1_1_a2_2_a3_2 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2);
estar_a1_2_a2_1_a3_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_2_a3_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_a1_2_a2_2_a3_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
```

```
estar_a1_2_a2_2_a3_2 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2);
```

B.12 Rede SAN com evento sincronizante com taxa funcional e probabilidades constantes convertido utilizando somente primitivas CTA

```
identifiers
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=3.5;
l11=1.5;
l12=2.1;
l13=0.9;

events

loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e7 (l7);
loc e6 (l6);
loc e8 (l8);
loc e9 (l9);
syn e10(l10);
syn e11(l11);
syn e12(l12);
syn e13(l13);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_func_pro_cons_c (continuous)

  aut A1
    stt est0 to (est1) e10 e11 e12 e13
    stt est1 to (est2) e2
```

```
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e10 e12
             to (est2) e11 e13
    stt est1 to (est0) e4
    stt est2 to (est0) e5

  aut A3
    stt est0 to (est1) e6
             to (est0) e10 e11
    stt est1 to (est2) e7
             to (est0) e9
    stt est2 to (est1) e8
             to (est2) e12 e13

results

estar_a1_0_a2_0_a3_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_a1_0_a2_0_a3_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_a1_0_a2_0_a3_2 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2);
estar_a1_0_a2_1_a3_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_a1_0_a2_1_a3_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_a1_0_a2_1_a3_2 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2);
estar_a1_0_a2_2_a3_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_a1_0_a2_2_a3_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_a1_0_a2_2_a3_2 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2);
estar_a1_1_a2_0_a3_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_a1_1_a2_0_a3_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_a1_1_a2_0_a3_2 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2);
estar_a1_1_a2_1_a3_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_a1_1_a2_1_a3_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_a1_1_a2_1_a3_2 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2);
estar_a1_1_a2_2_a3_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_a1_1_a2_2_a3_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_a1_1_a2_2_a3_2 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2);
estar_a1_2_a2_0_a3_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_a1_2_a2_0_a3_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_a1_2_a2_0_a3_2 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2);
```

```
estar_a1_2_a2_1_a3_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_a1_2_a2_1_a3_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_a1_2_a2_1_a3_2 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2);
estar_a1_2_a2_2_a3_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_a1_2_a2_2_a3_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
estar_a1_2_a2_2_a3_2 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2);
```

B.13 Rede SAN com evento sincronizante com taxa constante e probabilidades funcionais construído utilizando primitivas GTA

```
identifiers

l1=9;
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;

pi1=( ((st A2 == est0)*0.6) + ((st A2 == est1)*0.2) + ((st A2 == est2)*0.2) );
pi2=( ((st A2 == est0)*0.2) + ((st A2 == est1)*0.6) + ((st A2 == est2)*0.2) );
pi3=( ((st A2 == est0)*0.2) + ((st A2 == est1)*0.2) + ((st A2 == est2)*0.6) );

events

syn e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_cons_pro_func_f (continuous)

  aut A1
    stt est0 to (est1) e1(pi1) to (est2) e1(pi2) to (est3) e1(pi3)
    stt est1 to (est0) e2
    stt est2 to (est0) e3
    stt est3 to (est0) e4
```

```
  aut A2
    stt est0 to (est1) e5
    stt est1 to (est2) e6
    stt est2 to (est0) e7

  aut A3
    stt est0 to (est1) e1
    stt est1 to (est0) e8

results

estar_0_0_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_0_0_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_0_1_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_0_1_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_0_2_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_0_2_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_1_0_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_1_0_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_1_1_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_1_1_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_1_2_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_1_2_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_2_0_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_2_0_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_2_1_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_2_1_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_2_2_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_2_2_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
```

B.14 Rede SAN com evento sincronizante com taxa constante e probabilidades funcionais convertido utilizando somente primitivas CTA

```
identifiers

l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=5.4;
l10=1.8;
l11=1.8;
l12=5.4;
l13=1.8;
l14=5.4;

events

loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
loc e8 (l8);
syn e9 (l9);
syn e10 (l10);
syn e11 (l11);
syn e12 (l12);
syn e13 (l13);
syn e14 (l14);

partial reachability = ((st A1 == est0) && (st A2 == est0) && (st A3 == est0));

network ev_syn_cons_pro_func_c (continuous)
```

```
  aut A1
    stt est0 to (est1) e9 e11 e13 to (est2) e10 e12 e13 to (est3) e10 e11 e14
    stt est1 to (est0) e2
    stt est2 to (est0) e3
    stt est3 to (est0) e4

  aut A2
    stt est0 to (est1) e5 to (est0) e9 e10
    stt est1 to (est2) e6 to (est1) e11 e12
    stt est2 to (est0) e7 to (est2) e13 e14

  aut A3
    stt est0 to (est1) e9 e10 e11 e12 e13 e14
    stt est1 to (est0) e8

results

estar_0_0_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0);
estar_0_0_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1);
estar_0_1_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0);
estar_0_1_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1);
estar_0_2_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0);
estar_0_2_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1);
estar_1_0_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0);
estar_1_0_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1);
estar_1_1_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0);
estar_1_1_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1);
estar_1_2_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0);
estar_1_2_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1);
estar_2_0_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0);
estar_2_0_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1);
estar_2_1_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0);
estar_2_1_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1);
estar_2_2_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0);
estar_2_2_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1);
```

B.15 Rede SAN com evento sincronizante com taxa funcional e probabilidades funcionais construído utilizando primitivas GTA

```
identifiers
l1=( ((st A3 == est0)*5) + ((st A3 == est2)*3) ) ;
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=1;

pi1=( ((st A4 == est0)*0.7) + ((st A4 == est1)*0.3) );
pi2=( ((st A4 == est0)*0.3) + ((st A4 == est1)*0.7) );

events

syn e1 (l1);
loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
syn e8 (l8);
syn e9 (l9);
loc e10 (l10);

partial reachability = ((st A1 == est0) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est0));

network ev_loc_func_f (continuous)

  aut A1
    stt est0 to (est1) e1
    stt est1 to (est2) e2
```

```
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e1(pi1) to (est2) e1(pi2)
    stt est1 to (est0) e4
    stt est2 to (est0) e5

  aut A3
    stt est0 to (est1) e6
    stt est1 to (est2) e7 to (est0) e9
    stt est2 to (est1) e8

  aut A4
    stt est0 to (est1) e10
    stt est1 to (est0) e8 e9

results

estar_0_0_0_0 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est0);
estar_0_0_0_1 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est1);
estar_0_0_1_0 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est0);
estar_0_0_1_1 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est1);
estar_0_0_2_0 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est0);
estar_0_0_2_1 = (st A1 == est0) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est1);
estar_0_1_0_0 = (st A1 == est0) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est0);
estar_0_1_0_1 = (st A1 == est0) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est1);
estar_0_1_1_0 = (st A1 == est0) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est0);
estar_0_1_1_1 = (st A1 == est0) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est1);
estar_0_1_2_0 = (st A1 == est0) && (st A2 == est1) &&
```

```
(st A3 == est2) && (st A4 == est0);
estar_0_1_2_1 = (st A1 == est0) && (st A2 == est1) &&
(st A3 == est2) && (st A4 == est1);
estar_0_2_0_0 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est0) && (st A4 == est0);
estar_0_2_0_1 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est0) && (st A4 == est1);
estar_0_2_1_0 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est0);
estar_0_2_1_1 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est1);
estar_0_2_2_0 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est0);
estar_0_2_2_1 = (st A1 == est0) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est1);

estar_1_0_0_0 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est0);
estar_1_0_0_1 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est1);
estar_1_0_1_0 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est0);
estar_1_0_1_1 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est1);
estar_1_0_2_0 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est0);
estar_1_0_2_1 = (st A1 == est1) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est1);
estar_1_1_0_0 = (st A1 == est1) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est0);
estar_1_1_0_1 = (st A1 == est1) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est1);
estar_1_1_1_0 = (st A1 == est1) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est0);
estar_1_1_1_1 = (st A1 == est1) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est1);
estar_1_1_2_0 = (st A1 == est1) && (st A2 == est1) &&
(st A3 == est2) && (st A4 == est0);
estar_1_1_2_1 = (st A1 == est1) && (st A2 == est1) &&
```

```
(st A3 == est2) && (st A4 == est1);
estar_1_2_0_0 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est0) && (st A4 == est0);
estar_1_2_0_1 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est0) && (st A4 == est1);
estar_1_2_1_0 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est0);
estar_1_2_1_1 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est1);
estar_1_2_2_0 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est0);
estar_1_2_2_1 = (st A1 == est1) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est1);

estar_2_0_0_0 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est0);
estar_2_0_0_1 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est1);
estar_2_0_1_0 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est0);
estar_2_0_1_1 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est1) && (st A4 == est1);
estar_2_0_2_0 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est0);
estar_2_0_2_1 = (st A1 == est2) && (st A2 == est0) &&
(st A3 == est2) && (st A4 == est1);
estar_2_1_0_0 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est0);
estar_2_1_0_1 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est0) && (st A4 == est1);
estar_2_1_1_0 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est0);
estar_2_1_1_1 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est1) && (st A4 == est1);
estar_2_1_2_0 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est2) && (st A4 == est0);
estar_2_1_2_1 = (st A1 == est2) && (st A2 == est1) &&
(st A3 == est2) && (st A4 == est1);
estar_2_2_0_0 = (st A1 == est2) && (st A2 == est2) &&
```

```
(st A3 == est0) && (st A4 == est0);
estar_2_2_0_1 = (st A1 == est2) && (st A2 == est2) &&
(st A3 == est0) && (st A4 == est1);
estar_2_2_1_0 = (st A1 == est2) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est0);
estar_2_2_1_1 = (st A1 == est2) && (st A2 == est2) &&
(st A3 == est1) && (st A4 == est1);
estar_2_2_2_0 = (st A1 == est2) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est0);
estar_2_2_2_1 = (st A1 == est2) && (st A2 == est2) &&
(st A3 == est2) && (st A4 == est1);
```

B.16 Rede SAN com evento sincronizante com taxa funcional e probabilidades funcionais convertido utilizando somente primitivas CTA

```
identifiers
l2=1;
l3=1;
l4=1;
l5=1;
l6=1;
l7=1;
l8=1;
l9=1;
l10=1;
l11=3.5;
l12=1.5;
l13=1.5;
l14=3.5;
l15=2.1;
l16=0.9;
l17=2.1;
l18=0.9;

events

loc e2 (l2);
loc e3 (l3);
loc e4 (l4);
loc e5 (l5);
loc e6 (l6);
loc e7 (l7);
syn e8 (l8);
syn e9 (l9);
loc e10 (l10);
syn e11 (l11);
syn e12 (l12);
syn e13 (l13);
syn e14 (l14);
syn e15 (l15);
syn e16 (l16);
```

```
syn e17 (l17);
syn e18 (l18);

partial reachability = ((st A1 == est0) && (st A2 == est0) &&
(st A3 == est0) && (st A4 == est0));

network ev_loc_func_f (continuous)

  aut A1
    stt est0 to (est1) e11 e12 e13 e14 e15 e16 e17 e18
    stt est1 to (est2) e2
    stt est2 to (est0) e3

  aut A2
    stt est0 to (est1) e11 e13 e15 e17 to (est2) e12 e14 e16 e18
    stt est1 to (est0) e4
    stt est2 to (est0) e5

  aut A3
    stt est0 to (est1) e6 to (est0) e11 e12 e13 e14
    stt est1 to (est2) e7 to (est0) e9
    stt est2 to (est1) e8 to (est2) e15 e16 e17 e18

  aut A4
    stt est0 to (est1) e10 to (est0) e11 e12 e15 e16
    stt est1 to (est0) e8 e9 to (est1) e13 e14 e17 e18

results

estar_0_0_0_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est0);
estar_0_0_0_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est1);
estar_0_0_1_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est0);
estar_0_0_1_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est1);
estar_0_0_2_0 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est0);
```

```
estar_0_0_2_1 = (st A1 == est0) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est1);
estar_0_1_0_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est0);
estar_0_1_0_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est1);
estar_0_1_1_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est0);
estar_0_1_1_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est1);
estar_0_1_2_0 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est0);
estar_0_1_2_1 = (st A1 == est0) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est1);
estar_0_2_0_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est0);
estar_0_2_0_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est1);
estar_0_2_1_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est0);
estar_0_2_1_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est1);
estar_0_2_2_0 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est0);
estar_0_2_2_1 = (st A1 == est0) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est1);

estar_1_0_0_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est0);
estar_1_0_0_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est1);
estar_1_0_1_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est0);
estar_1_0_1_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est1);
estar_1_0_2_0 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est0);
estar_1_0_2_1 = (st A1 == est1) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est1);
```

```
estar_1_1_0_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est0);
estar_1_1_0_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est1);
estar_1_1_1_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est0);
estar_1_1_1_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est1);
estar_1_1_2_0 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est0);
estar_1_1_2_1 = (st A1 == est1) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est1);
estar_1_2_0_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est0);
estar_1_2_0_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est1);
estar_1_2_1_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est0);
estar_1_2_1_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est1);
estar_1_2_2_0 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est0);
estar_1_2_2_1 = (st A1 == est1) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est1);

estar_2_0_0_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est0);
estar_2_0_0_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est0) &&
(st A4 == est1);
estar_2_0_1_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est0);
estar_2_0_1_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est1) &&
(st A4 == est1);
estar_2_0_2_0 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est0);
estar_2_0_2_1 = (st A1 == est2) && (st A2 == est0) && (st A3 == est2) &&
(st A4 == est1);
estar_2_1_0_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est0);
```

```
estar_2_1_0_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est0) &&
(st A4 == est1);
estar_2_1_1_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est0);
estar_2_1_1_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est1) &&
(st A4 == est1);
estar_2_1_2_0 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est0);
estar_2_1_2_1 = (st A1 == est2) && (st A2 == est1) && (st A3 == est2) &&
(st A4 == est1);
estar_2_2_0_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est0);
estar_2_2_0_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est0) &&
(st A4 == est1);
estar_2_2_1_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est0);
estar_2_2_1_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est1) &&
(st A4 == est1);
estar_2_2_2_0 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est0);
estar_2_2_2_1 = (st A1 == est2) && (st A2 == est2) && (st A3 == est2) &&
(st A4 == est1);
```

B.17 Rede SAN do Padrão de mobilidade *Random Waypoint*

```
identifiers

Velocidade1 = 5 ; // velocidade do nodo 1 (m/s)
Parada1 = 0.1 ; // Tempo que o nodo permanece parado antes de iniciar
//novo movimento (s)

NumEspacos = 5 ; //
Esp = [0..4] ;

TamArea = 1000 ; // tamanho da area de movimentacao considerada (m)
TamEspaco  = (TamArea / NumEspacos) ; // Tamanho do espaco representado
//por um estado da cadeia de localizacao (m)

TaxaVel1 = Velocidade1 / TamEspaco ;  // Taxa de mudanca de localizacao
//do nodo na area considerada
TaxaDecDest1 = ( (st Dest_MN1) == (st Loc_MN1) ) * (1 / (Parada1*(NumEspacos-1)));
// Decidir por novo destino

TaxaAndarDir1 = TaxaVel1 * ( st Dest_MN1 > st Loc_MN1 ) ;
TaxaAndarEsq1 = TaxaVel1 * ( st Dest_MN1 < st Loc_MN1 ) ;

events

loc txm1(TaxaDecDest1);
loc txAndarD1(TaxaAndarDir1);
loc txAndarE1(TaxaAndarEsq1);

reachability = 1;

network mobilidade(continuous)

aut Dest_MN1
stt Dest0
to(Dest1) txm1 to(Dest2) txm1 to(Dest3) txm1 to(Dest4) txm1
stt Dest1
to(Dest0) txm1 to(Dest2) txm1 to(Dest3) txm1 to(Dest4) txm1
stt Dest2
```

```
to(Dest0) txm1 to(Dest1) txm1 to(Dest3) txm1 to(Dest4) txm1
stt Dest3
to(Dest0) txm1 to(Dest1) txm1 to(Dest2) txm1 to(Dest4) txm1
stt Dest4
to(Dest0) txm1 to(Dest1) txm1 to(Dest2) txm1 to(Dest3) txm1


aut Loc_MN1
    stt Esp0
        to(Esp1) txAndarD1
    stt Esp1
        to(Esp0) txAndarE1 to(Esp2) txAndarD1
    stt Esp2
        to(Esp1) txAndarE1 to(Esp3) txAndarD1
    stt Esp3
        to(Esp2) txAndarE1 to(Esp4) txAndarD1
    stt Esp4
        to(Esp3) txAndarE1


results
prob_MN1_E0 = (st Loc_MN1 == 0) ;
prob_MN1_E1 = (st Loc_MN1 == 1) ;
prob_MN1_E2 = (st Loc_MN1 == 2) ;
prob_MN1_E3 = (st Loc_MN1 == 3) ;
prob_MN1_E4 = (st Loc_MN1 == 4) ;
```

B.18 Rede SAN do Padrão de mobilidade *Random Waypoint* convertido utilizando somente primitivas CTA

```
identifiers

TaxaVel1 = 1;
TaxaDecDest1 = 2.5;

events

syn t_0(TaxaDecDest1);
syn t_1(TaxaDecDest1);
syn t_2(TaxaDecDest1);
syn t_3(TaxaDecDest1);
syn t_4(TaxaDecDest1);

syn t_d_1(TaxaVel1);
syn t_d_2(TaxaVel1);
syn t_d_3(TaxaVel1);
syn t_d_4(TaxaVel1);

syn t_e_0(TaxaVel1);
syn t_e_1(TaxaVel1);
syn t_e_2(TaxaVel1);
syn t_e_3(TaxaVel1);

reachability = 1;

network mobilidade(continuous)

aut Dest_MN1
stt Dest0
        to(Dest0) t_e_0 t_e_1 t_e_2 t_e_3
to(Dest1) t_0 to(Dest2) t_0 to(Dest3) t_0 to(Dest4) t_0
stt Dest1
        to(Dest1) t_d_1 t_e_1 t_e_2 t_e_3
to(Dest0) t_1 to(Dest2) t_1 to(Dest3) t_1 to(Dest4) t_1
stt Dest2
        to(Dest2) t_d_1 t_d_2 t_e_2 t_e_3
```

```
to(Dest0) t_2 to(Dest1) t_2 to(Dest3) t_2 to(Dest4) t_2
stt Dest3
        to(Dest3) t_d_1 t_d_2 t_d_3 t_e_3
to(Dest0) t_3 to(Dest1) t_3 to(Dest2) t_3 to(Dest4) t_3
stt Dest4
        to(Dest4) t_d_1 t_d_2 t_d_3 t_d_4
to(Dest0) t_4 to(Dest1) t_4 to(Dest2) t_4 to(Dest3) t_4


aut Loc_MN1
    stt Esp0
        to(Esp0) t_0
        to(Esp1) t_d_1
    stt Esp1
        to(Esp1) t_1
        to(Esp0) t_e_0 to(Esp2) t_d_2
    stt Esp2
        to(Esp2) t_2
        to(Esp1) t_e_1 to(Esp3) t_d_3
    stt Esp3
        to(Esp3) t_3
        to(Esp2) t_e_2 to(Esp4) t_d_4
    stt Esp4
        to(Esp4) t_4
        to(Esp3) t_e_3


results
prob_MN1_E0 = (st Loc_MN1 == 0) ;
prob_MN1_E1 = (st Loc_MN1 == 1) ;
prob_MN1_E2 = (st Loc_MN1 == 2) ;
prob_MN1_E3 = (st Loc_MN1 == 3) ;
prob_MN1_E4 = (st Loc_MN1 == 4) ;
```