4

Avaliação de Desempenho de Sistemas Paralelos[1]

Leonardo Brenner[2] *(PUCRS, lbrenner@inf.pucrs.br)*
Paulo Fernandes[3] *(PUCRS, paulof@inf.pucrs.br)*
Afonso Sales[4] *(PUCRS, asales@inf.pucrs.br)*

Resumo:

Este curso introduz os conceitos de avaliação de desempenho através de métodos analíticos, mais especificamente, avaliação de desempenho de sistemas paralelos usando o formalismo de Redes de Autômatos Estocásticos (SAN).

O formalismo SAN define um sistema como um conjunto de subsistemas que interagem ocasionalmente. Para isso, o formalismo define primitivas de sincronismo e paralelismo.

No decorrer deste curso apresenta-se uma definição informal do formalismo SAN, bem como técnicas e exemplos de modelagem de sistemas paralelos em SAN. Adicionalmente, a codificação de exemplos para a ferramenta PEPS é apresentada, assim como a gramática utilizada na ferramenta.

[1]Os autores são parcialmente suportados pelo Projeto CASCO (T.A. 24) do convênio HP Brasil-PUCRS.

[2]Mestre em Ciência da Computação pela PUCRS (2003) e Bacharel em Ciência da Computação pela mesma universidade (2001). Pesquisador do Projeto CASCO no convênio HP Brasil-PUCRS.

[3]Doutor em Informática pelo Institut National Polytechnique de Grenoble (1998). Mestre em Ciência da Computação pela UFRGS (1990) e Bacharel em Ciência da Computação pela mesma universidade (1987). Professor da Faculdade de Informática da PUCRS e coordenador do Programa de Pós-Graduação em Ciência da Computação na mesma universidade.

[4]Mestre em Ciência da Computação pela PUCRS (2003) e Bacharel em Informática pela mesma universidade (1999). Pesquisador do Projeto CASCO no convênio HP Brasil-PUCRS.

4.1. Introdução

Avaliação de desempenho é incontornável para o desenvolvimento racional de aplicações paralelas de alto desempenho. Apesar de ser a opção mais freqüente, a avaliação através de monitoração não aprensenta uma confiabilidade excepcional nem é economicamente uma opção interessante. Os benchmarks, apesar de bastante populares, são um pálido indicativo do desempenho de uma máquina em situações pouco reais. Porém, a grande desvantagem da monitoração reside na necessidade de uma implementação física completa do sistema para que este possa ser avaliado.

Por sua vez, os métodos analíticos apresentam a grande vantagem de fornecer informações sobre o modelo teórico do funcionamento de máquinas e programas. A grande desvantagem do uso de métodos analíticos reside na descrição dos modelos teóricos. Esta desvantagem pode ser atenuada através do uso de formalismos que facilitem a modelagem, como por exemplo: Cadeias de Markov, Redes de Autômatos Estocásticos (SAN), Redes de Petri Estocásticas (SPN), entre outros.

Este curso tem como objetivo apresentar especificamente o formalismo SAN, o qual possui primitivas adequadas à descrição de processos paralelos. Com o uso deste formalismo é possível extrair índices de desempenho estacionários, como: utilização média de processadores, tempos médios de atraso, carga dos canais de comunicação, etc.

Especificamente, serão vistos modelos teóricos descrevendo máquinas paralelas, bem como algoritmos paralelos. Os exemplos vistos serão codificados para a ferramenta PEPS que faz análise estacionária de modelos SAN.

As seções seguintes apresentam uma definição informal do formalismo SAN (Seção 4.2.) e exemplos modelados neste formalismo (Seção 4.3.). Por último são tecidas algumas considerações sobre avaliação de desempenho através de métodos analíticos.

4.2. Redes de Autômatos Estocásticos

O formalismo SAN foi inicialmente proposto por Plateau em 1984 [PLA 84]. No início da década de 90 as primeiras soluções foram formalizadas para modelos em escala de tempo contínua e discreta [PLA 91]. Na virada do século, o formalismo SAN foi novamente revisto face aos eficientes algoritmos para modelos a escala de tempo contínua [FER 98], na mesma ocasião foi disponibilizada a versão 2.0 da ferramenta PEPS , a qual implementa métodos iterativos[5] para resolução de modelos SAN.

A idéia principal do formalismo SAN é modelar um sistema em vários subsistemas, ou seja, um sistema composto de módulos "quase independentes". A expressão "quase independente" denota a possibilidade de ocorrer interação entre cada subsistema. SAN é um formalismo para modelagem de sistemas com grande espaço de estados. Essa modularização definida pelo formalismo SAN permite o armazenamento e a solução eficiente de sistemas complexos por evitar os prejuízos da *explosão do espaço de estados* que ocorre no formalismo de Cadeias de Markov, o qual SAN tem equivalência de representação.

[5]Para resolução computacional os métodos iterativos de resolução tais como *Método da Potência*, *Método de Arnoldi* e *GMRES*, implementados no PEPS, são mais eficientes que os métodos diretos, *Eliminação de Gauss*, por exemplo, por aproveitar as características das técnicas de armazenamento esparsas e não requerer tanta memória quanto os métodos diretos.

Cada subsistema é representado por um *autômato estocástico* e por *transições* entre os estados deste autômato. As transições entre os estados de cada autômato são modeladas por um processo estocástico de tempo contínuo ou discreto definidos por distribuições exponenciais ou geométricas respectivamente.

É interessante ressaltar que toda SAN pode ser representada por um único autômato estocástico que contém todos os estados possíveis do sistema. Esse único autômato corresponde à cadeia de Markov equivalente ao modelo SAN.

Note que a visão geral de modelagem em SAN apresentada nesta seção se aplica tanto para a escala de tempo contínua como para a escala de tempo discreta. Entretanto, as explicações e exemplos apresentados ao longo deste curso fazem referência à escala de tempo contínua (*taxas de ocorrência*) e não à escala de tempo discreta (*probabilidades de ocorrência*). A diferenciação entre as duas escalas de tempo dá-se apenas na obtenção do *Descritor Markoviano* de cada modelo. Enquanto um modelo em SAN à escala de tempo contínua gera uma cadeia de Markov à escala de tempo contínua (CTMC - *Continuous Time Markov Chain*), um modelo em SAN descrito em escala de tempo discreta gera uma cadeia de Markov à escala de tempo discreta (DTMC - *Discrete Time Markov Chain*).

No decorrer deste curso adota-se a seguinte notação para a definição dos modelos em SAN:

Sejam

$\mathcal{A}^{(i)}$ o i-ésimo autômato de um modelo SAN, numerando o primeiro autômato como $\mathcal{A}^{(1)}$;

$x^{(i)}$ o x-ésimo estado do autômato $\mathcal{A}^{(i)}$, numerando o primeiro estado do primeiro autômato como $0^{(1)}$;

e identificador de um evento (local ou sincronizante);

$e(\pi_g)$ identificador de um evento e com probabilidade constante de rotação π_g[6];

$e(f_g)$ identificador de um evento e com probabilidade funcional de rotação definida pela função f_g;

As probabilidades alternativas de rotação (π_g ou f_g) são utilizadas quando um evento tem duas ou mais alternativas de transição. Dessa maneira, probabilidades de rotação são utilizadas para indicar em que proporções o evento seguirá por uma transição ou por outra. A probabilidade de rotação pode ser omitida caso esta seja igual a 1.0. Outro ponto importante é que a soma das probabilidades de rotação de um evento deve ser sempre igual a 1.0.

A seguir, cada primitiva de modelagem é explicada detalhadamente.

4.2.1. Autômatos Estocásticos

Autômato estocástico é um modelo matemático de um sistema que possui entradas e saídas discretas. O sistema pode se encontrar em qualquer um dentre o número finito dos estados do sistema ou das configurações internas. O estado interno em que o sistema se encontra sumariza as informações sobre as entradas anteriores e indica ainda o

[6]O índice g não tem nenhuma semântica particular para qualquer uma das notações definidas nesta seção.

que é necessário para determinar o comportamento do sistema para as entradas seguintes [HOP 79].

Baseado nessa definição, pode-se descrever um autômato estocástico como um *conjunto finito de estados* e um *conjunto finito de transições* entre esses estados. A denominação de estocásticos atribuída a esses autômatos dá-se pela razão do tempo ser tratado como uma variável aleatória, a qual obedece a uma distribuição exponencial para a escala de tempo contínua e uma distribuição geométrica para a escala de tempo discreta.

O *estado local* do sistema modelado em SAN é o estado individual de cada autômato do modelo. Por sua vez, o *estado global* do mesmo é definido pela combinação dos estados locais de todos os autômatos que compõem o modelo. A mudança do estado global do sistema dá-se pela mudança do estado local de qualquer um dos autômatos do modelo.

A mudança de um determinado estado local para outro é feita através de transições. As transições são construções que indicam a possibilidade de mudança entre um estado e outro. No entanto, cada transição necessita ter ao menos um *evento* associado a ela para que essa possa ser disparada.

A Figura 4.1 apresenta um modelo em SAN com dois autômatos completamente independentes.

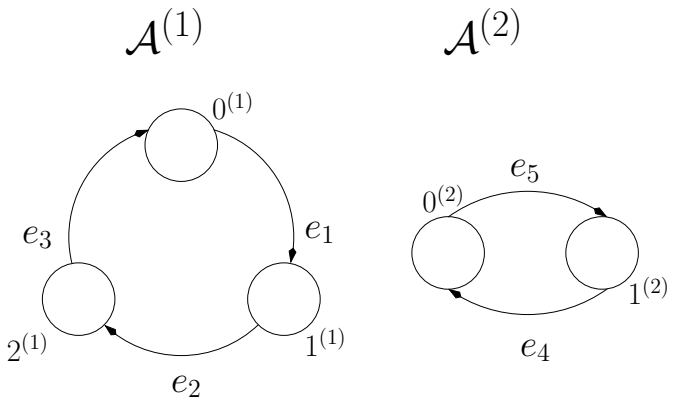


Figura 4.1: Modelo em SAN **com 2 autômatos independentes.**

Neste primeiro exemplo, o autômato $\mathcal{A}^{(1)}$ do modelo possui três estados $0^{(1)}$, $1^{(1)}$ e $2^{(1)}$, enquanto o autômato $\mathcal{A}^{(2)}$ possui apenas dois estados $0^{(2)}$ e $1^{(2)}$. Dos cinco eventos que são modelados neste exemplo, três eventos (e_1, e_2 e e_3) ocorrem no autômato $\mathcal{A}^{(1)}$, enquanto outros dois eventos (e_4 e e_5) ocorrem no autômato $\mathcal{A}^{(2)}$.

Evento	Taxa de Ocorrência
e_1	τ_1
e_2	τ_2
e_3	τ_3
e_4	τ_4
e_5	τ_5

Tabela 4.1: Taxa de ocorrência dos eventos do modelo SAN **da Figura 4.1.**

Apresenta-se na Figura 4.2 a CTMC equivalente ao modelo em SAN da Figura 4.1.

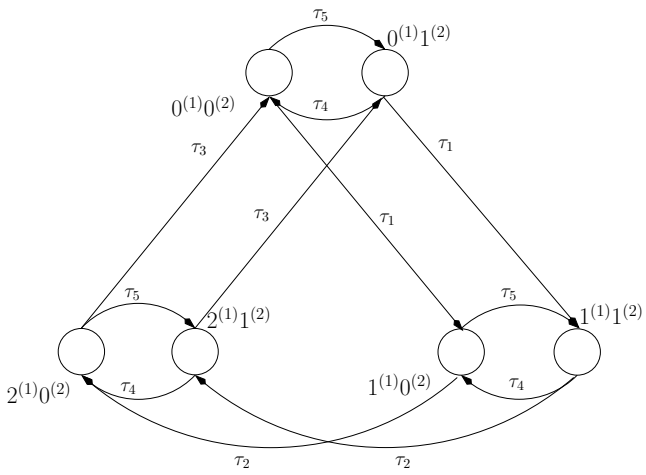


Figura 4.2: CTMC equivalente ao modelo da Figura 4.1.

Note que no modelo da Figura 4.1 não há interação entre os dois autômatos, *i.e.*, existe apenas *eventos locais* em cada um deles. Na seção a seguir, é vista a definição e os tipos de eventos que podem ser utilizados nos modelos em SAN.

4.2.2. Eventos

Evento é a entidade do modelo responsável pela ocorrência de uma transição, a qual muda o *estado global* do modelo. Um ou mais eventos podem estar associados a uma transição e esta é disparada através da ocorrência de qualquer um dos eventos a ela associada.

Dois tipos de eventos podem ser modelados no formalismo SAN. Um evento pode ser classificado como evento *local* ou *sincronizante*.

4.2.2.1. Eventos Locais

Os eventos locais são utilizados em SAN para alterar o estado local de um único autômato, sem que essa alteração ocasione uma mudança de estado em qualquer outro autômato do modelo. Esse tipo de evento é particularmente interessante, pois permite que vários autômatos tenham um comportamento paralelo, trabalhando independentemente sem que haja interação entre eles.

Na Figura 4.1, pode-se observar exemplos de eventos locais, a qual é composta exclusivamente por esse tipo de evento.

4.2.2.2. Eventos Sincronizantes

Os eventos sincronizantes alteram o estado local de dois ou mais autômatos simultaneamente, *i.e.*, a ocorrência de um evento sincronizante em um autômato *força* a ocorrência deste mesmo evento nos outros autômatos envolvidos. Através dos eventos

sincronizantes, é possível fazer a interação entre autômatos. Essa interação dá-se sob a forma de sincronismo no disparo das transições.

A classificação de um evento como *local* ou *sincronizante* é dada pela aparição do identificador do evento e no conjunto de eventos de um autômato. Caso o identificador do evento apareça apenas no conjunto de eventos de um único autômato, o evento é classificado como *evento local*. Se o mesmo identificador aparecer no conjunto de eventos de *vários* autômatos, o evento é classificado como *evento sincronizante*.

Cada evento deve possuir uma taxa de ocorrência e uma probabilidade de rotação associada ao mesmo. Tanto a taxa de ocorrência como a probabilidade de rotação podem ter associados valores constantes ou valores funcionais. Taxas e probabilidades funcionais assumem valores diferentes conforme os estados dos outros autômatos do modelo.

4.2.3. Taxas e Probabilidades Funcionais

Taxas e probabilidades funcionais constituem a segunda possibilidade de interação entre autômatos nos modelos em SAN. Outra possibilidade é a utilização de eventos sincronizantes[7]. A utilização de funções para definir taxas e/ou probabilidades permite associar a um mesmo evento diferentes valores conforme o estado global do modelo.

As taxas e probabilidades funcionais são expressas por funções que levam em consideração os estados atuais dos autômatos do modelo, podendo desta forma variar seu valor conforme os estados em que se encontram os autômatos envolvidos na função.

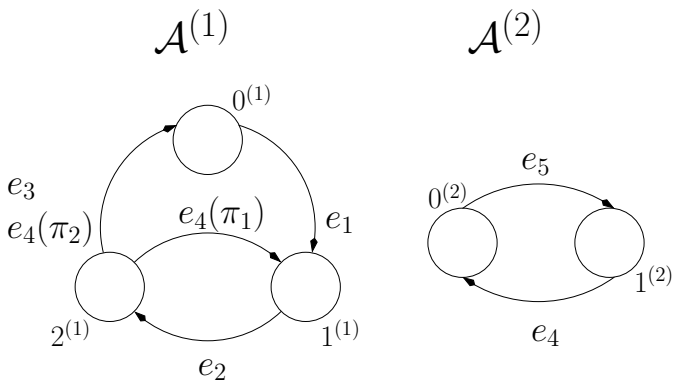


Figura 4.3: Modelo em SAN **com 2 autômatos, 1 evento sincronizante e uma taxa funcional.**

A Figura 4.3 apresenta um modelo em SAN com 2 autômatos de 3 e 2 estados respectivamente. Da mesma forma que o modelo em SAN da Figura 4.1, também utiliza-se cinco eventos no modelo em SAN da Figura 4.3. Entretanto, o evento e_4 é um evento sincronizante, visto que o mesmo está associado a mais de um autômato do modelo. Este evento ainda possui probabilidades associadas a diferentes transições no autômato $\mathcal{A}^{(1)}$. Além disso, o evento e_5 possui agora a função f_1 associada à sua taxa de ocorrência. Por

[7]A utilização de taxas e probabilidades funcionais não está limitada aos eventos locais e podem ser empregadas nos eventos sincronizantes exatamente como nos eventos locais.

Evento	Taxa de Ocorrência
e_1	τ_1
e_2	τ_2
e_3	τ_3
e_4	τ_4
e_5	f_1

Tabela 4.2: Taxa de ocorrência dos eventos do modelo SAN da Figura 4.3.

sua vez, a função f_1 é definida como:

$$f_1 = \begin{cases} \lambda_1 & \text{se autômato } \mathcal{A}^{(1)} \text{ está no estado } 0^{(1)}; \\ 0 & \text{se autômato } \mathcal{A}^{(1)} \text{ está no estado } 1^{(1)}; \\ \lambda_2 & \text{se autômato } \mathcal{A}^{(1)} \text{ está no estado } 2^{(1)}. \end{cases}$$

Como pode-se observar na definição da função f_1, a taxa de ocorrência da transição do estado $0^{(2)}$ para o estado $1^{(2)}$ é igual a λ_1 (caso o autômato $\mathcal{A}^{(1)}$ esteja no estado $0^{(1)}$), igual a λ_2 (caso o autômato $\mathcal{A}^{(1)}$ esteja no estado $2^{(1)}$), e a transição não ocorrerá caso o autômato $\mathcal{A}^{(1)}$ esteja no estado $1^{(1)}$.

A Figura 4.4 apresenta a CTMC equivalente ao modelo em SAN da Figura 4.3.

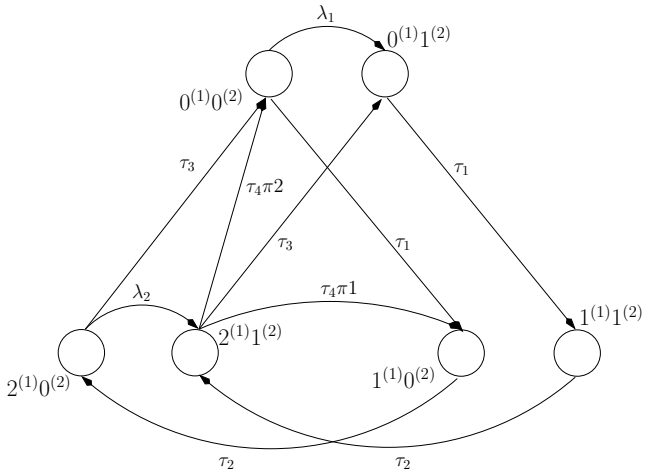


Figura 4.4: CTMC equivalente ao modelo da Figura 4.3.

Assim como as taxas de ocorrência podem ser expressas por funções, o mesmo pode ocorrer com as probabilidades alternativas de rotação de cada evento. A definição de funções usadas para expressar as probabilidades funcionais de rotação são exatamente iguais as funções usadas para definir as taxas de ocorrência.

4.2.3.1. Outras Funções

Outros dois tipos de funções ainda são utilizadas em SAN: *Função de Atingibilidade* e *Funções de Integração*. As expressões que definem a função de atingibilidade e

as funções de integração são descritas da mesma forma que as taxas e probabilidades funcionais. Porém, esses dois tipos de funções desempenham papéis diferenciados conforme explicados a seguir.

Função de Atingibilidade Devido à representação em SAN ser de forma modular e o autômato global (equivalente à cadeia de Markov) se constituir pela combinação de todos os autômatos do modelo, é necessário especificar uma função que defina quais os *estados atingíveis* deste autômato global que representam a SAN .

A definição de quais destes estados podem ser *atingidos* ou *alcançados* em SAN é dada pela *função de atingibilidade*. Essa função é definida usando-se as mesmas regras adotadas para a definição de taxas e probabilidades funcionais.

A noção de função de atingibilidade fica mais clara ao se imaginar, por exemplo, um modelo de compartilhamento de recursos, onde se tem N clientes disputando R recursos. Este sistema pode ser modelado em SAN usando um autômato com dois estados para cada cliente. O estado $0^{(n)}$ representa que o recurso não está sendo utilizado pelo cliente, enquanto o estado $1^{(n)}$ representa que o recurso está em uso pelo cliente. É fácil imaginar que, se o número R de recursos for menor do que o número N de clientes, o estado global que representa todos os clientes utilizando um recurso não poderá ser atingido, pois este estado não corresponde a realidade do modelo. Os estados que possuem tal característica são chamados de *estados inatingíveis* e devem ser eliminados do modelo através da *função de atingibilidade,* pois a probabilidade do modelo encontrar-se em algum destes estados é igual a *zero*.

A função de atingibilidade correta para o modelo de compartilhamento de recursos descrito acima é:

$$reachability = \sum_{n=1}^{N} st(A^{(n)}) \leq R$$

Funções de Integração Define-se *funções de integração* para a obtenção de resultados numéricos sobre o modelo em SAN. As funções de integração avaliam qual a probabilidade do modelo em SAN encontrar-se em um determinado estado.

Com isso, pode-se compor funções de integração que levem em conta a probabilidade do modelo se encontrar em um conjunto de estados, podendo assim se obter índices de desempenho e confiabilidade do modelo. Essas funções de integração são avaliadas sobre o *vetor de probabilidades* que contém a probabilidade do modelo se encontrar em cada um dos estados pertencente a ele.

Um exemplo de função de integração, tendo em mente o modelo de compartilhamento de recursos exposto anteriormente, é dado pela função u, onde se quer descobrir a probabilidade do autômato $\mathcal{A}^{(1)}$ não estar utilizando o recurso, *i.e.*, encontrar-se no estado $0^{(1)}$.

$$u = st(\mathcal{A}^{(1)}) == 0$$

Via de regra, todas as funções são modeladas em SAN da mesma forma, o que as diferenciam é como elas são empregadas no modelo.

4.3. Exemplos de modelagem

Esta seção apresenta alguns exemplos de modelagem em SAN visando aplicar os conceitos e as primitivas de modelagem vistas na Seção 4.2.. Para isso, é feita a apresentação dos modelos SAN de cada exemplo exposto, bem como a respectiva representação textual para a ferramenta PEPS.

A Seção 4.3.1. apresenta um modelo de compartilhamento de recurso. Na Seção 4.3.2. é exposto uma rede de filas de espera modelada, enquanto os dois últimos modelos descrevem uma situação de fontes *On/Off* e um *cluster* com protocolo de comunicação UDP.

4.3.1. Compartilhamento de Recursos

O primeiro exemplo apresentado descreve uma situação de compartilhamento de P recursos indistinguíveis disputados por N clientes com o mesmo padrão de comportamento. Neste sentido, é preciso modelar cada cliente e recurso disponível. Para modelar esta situação em SAN utiliza-se N autômatos idênticos (replicados). Cada autômato replicado representa um cliente do sistema. Um autômato é composto de dois estados ($0^{(n)}, 1^{(n)}$) onde o estado $0^{(n)}$ (*ocioso*) significa que o cliente n não está de posse de um recurso enquanto o estado $1^{(n)}$ (*ativo*) significa que este cliente está usando um dos recurso disponíveis. Esse modelo pode ser visto na Figura 4.5.

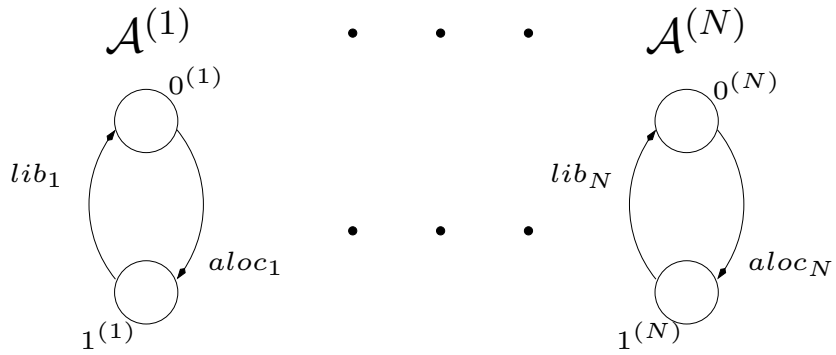


Figura 4.5: Modelo SAN **para compartilhamento de recursos.**

A liberação de um recurso pelo cliente n ($1^{(n)} \rightarrow 0^{(n)}$) é representada por um evento local lib_n, esse evento ocorre com taxa μ de forma completamente independente, ou seja, a liberação de um recurso não depende do estado do sistema (estado dos demais autômatos). Por outro lado, a alocação de um recurso ($0^{(n)} \rightarrow 1^{(n)}$) é representada por um evento local $aloc_n$ o qual é disparado por uma taxa de ocorrência funcional (f). É esta taxa f quem regula (modela) a contenção de recursos. Esta função deve resultar um valor nulo (0.0) quando não houver nenhum recurso disponível e uma taxa não-nula (λ) caso contrário. A expressão desta taxa funcional é mostrada na função abaixo.

$$f = [(nb\ \ ativo) < P] * \lambda;$$

Note-se que neste modelo temos representado de forma compacta 2^N estados globais, porém nem todos estes estados são atingíveis[8]. Na verdade, todos os estados globais cuja composição de estados locais represente mais de P recursos utilizados são considerados inatingíveis.

Este modelo descreve toda a interação entre os autômatos através da taxa funcional (f), logo não foi necessário utilizar nenhum evento sincronizante.

4.3.1.1. Arquivo SAN para o PEPS

Criou-se o seguinte arquivo `san` para modelo de compartilhamento de recursos apresentado neste exemplo. Foi define para este arquivo o número de 4 recursos e 20 clientes.

Definição dos identificadores e domínio

```
identifiers
  P      = 4;                        // número de recursos disponíveis
  N      = [0..19];                  // domínio de replicações
  mu     = 9;                        // taxa para liberar o recurso
  lambda = 6;                        // taxa para alocar o recurso
  f      = (nb ativo < P) * lambda; // função de acesso ao recurso
```

Nesta primeira parte do arquivo definiu-se cinco identificadores e domínios, *P*, *N*, *mu*, *lambda* e *f*, dos quais *P* e *N* definem as características do modelo, como o número de recursos disponíveis e o número de clientes disputando os recursos, calculado a partir do domínio de replicações *N*. Os três últimos identificadores representam as taxas liberação (*mu*) e alocação (*lambda*) de recurso no modelo e a taxa funcional para alocação de recursos (*f*), a qual retorna λ se houver ao menos um recurso disponível e 0.0 caso contrário.

Definição dos eventos

```
events
  loc aloc (f);  // evento associado a alocação de recursos
  loc lib  (mu); // evento associado a liberação de recursos
```

Após a definição de identificadores, define-se o conjunto de eventos que irá disparar uma transição. Ao evento $aloc$ associou-se o identificador f que define uma taxa para a alocação de recursos, enquanto para o evento lib associou-se o identificador mu que define a taxa de liberação de um recurso.

Função de atingibilidade

```
reachability = (nb ativo <= P);
```

A função de atingibilidade é definida permitindo que o número máximo de autômato no estado *ativo* seja igual a *P*, ou seja, um estado global é atingível caso o número de clientes utilizando recursos seja menor ou igual a *P*.

[8] O número de estados atingíveis para um modelo com N clientes compartilhando P recursos é igual a $\sum_{i=0}^{P} \binom{N}{i}$. Onde $\binom{N}{i}$ representa o número de combinações não ordenadas de i elementos tirados de um conjunto contendo um total de N elementos.

Definição dos autômatos

```
network cr (continuous)
  aut cliente[N]  // o autômato cliente é replicado N vezes
    stt ocioso    // o cliente está ocioso
      to (ativo)  // a passagem de ocioso para ativo ocorre com o
        aloc      //   evento local aloc
    stt ativo     // o cliente está usando 1 recurso
      to (ocioso) // a passagem de ativo para ocioso ocorre com o
        lib       //   evento local lib
```

Definiu-se para esse modelo apenas um autômato, o qual é replicado 20 vezes ($[0..19]$). Apenas dois estados foram modelados, que são *ocioso* e *ativo*. Associou-se o evento $aloc$ à transição que passa o autômato do estado *ocioso* para o estado *ativo*, enquanto o evento lib foi associado à transição que passa o autômato do estado *ativo* para o estado *ocioso*.

Funções de integração

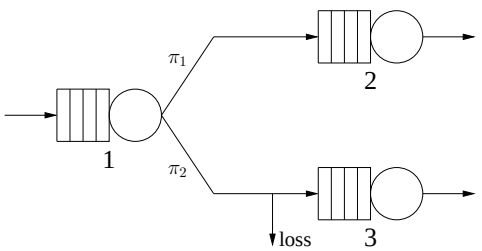
```
results
  uso4  = nb ativo==4; //probabilidade de ter 4 recursos ocupados
  uso3  = nb ativo==3; //probabilidade de ter 3 recursos ocupados
  uso2  = nb ativo==2; //probabilidade de ter 2 recursos ocupados
  uso1  = nb ativo==1; //probabilidade de ter 1 recurso ocupado
  uso0  = nb ativo==0; //probabilidade de nenhum recurso estar ocupado
  media = nb ativo;    //número médio de recursos ocupados
```

Seis resultados (funções de integração) foram criados. Esses resultados nos dizem a probabilidade de que 1, 2, 3 ou 4 recursos estejam em uso, de que nenhum recurso esteja em uso e ainda o número médio de recursos em uso.

4.3.2. Rede de Filas de Espera

O segundo exemplo apresentado descreve uma rede de filas de espera aberta com três filas com capacidade limitada.

Neste exemplo admite-se a rede de filas descrita na Figura 4.6.



$$
\begin{array}{llll}
|\mathcal{M}| = 3 & K_1 = 3 & C_1 = 1 & S_1 = \frac{1}{\tau_1} \\
R = 1 & K_2 = 3 & C_2 = 2 & S_2 = \frac{1}{\tau_2} \\
 & K_3 = 2 & C_3 = 1 & S_3 = \frac{1}{\tau_3} \\
 & P_{1,2} = \pi_1 & B_{1,2} = blocking & \\
 & P_{1,3} = \pi_2 & B_{1,3} = loss & \\
 & L_1 = \lambda_1 & &
\end{array}
$$

Figura 4.6: Rede de filas de espera aberta com capacidade limitada.

Neste modelo utiliza-se um autômato para representar cada uma das filas. As chegadas de clientes na primeira fila e as saídas da última fila são representadas por eventos locais (l_1, l_2 e l_3). A passagem de clientes entre as filas é representada por eventos sincronizantes ($e_1 2$ e $e_1 3$).

A Figura 4.7 representa o modelo SAN para a rede de filas de espera aberta descrita na Figura 4.6.

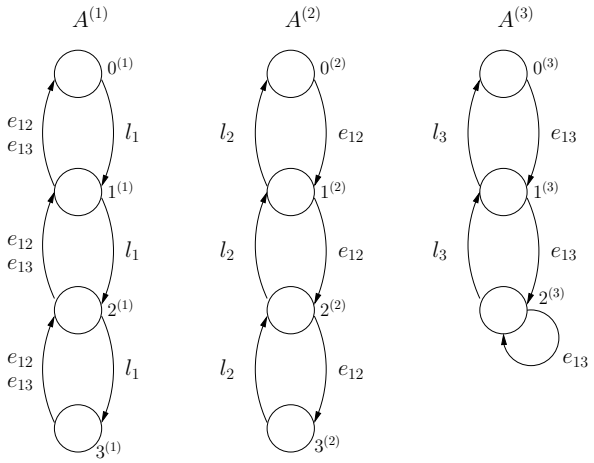


Figura 4.7: Modelo SAN **para a rede de filas de espera aberta.**

Neste exemplo não foi necessária a utilização de taxas ou probabilidades funcionais e também se não tem estados globais inatingíveis no modelo, ao contrário do exemplo anterior. Cabe salientar que a ausência de probabilidades e taxas funcionais não está relacionada com a ausência de estados inatingíveis.

4.3.2.1. Arquivo SAN para o PEPS

O arquivo `san` que implementa o modelo de rede de filas de espera como descrito neste exemplo segue abaixo.

Definição dos identificadores e domínios

```
identifiers
  Ci1 = 2; //número de servidores da fila 1
  Ci2 = 1; //número de servidores da fila 2
  Ci3 = 2; //número de servidores da fila 3

  Ki1 = [0..3]; //capacidade de fila 1
  Ki2 = [0..3]; //capacidade de fila 2
  Ki3 = [0..2]; //capacidade de fila 3

  lambda1 = 5; //taxa de chegada de clientes na fila 1
  Si1 = 0.2; //tempo de serviço da fila 1
  Si2 = 0.3; //tempo de serviço da fila 2
  Si3 = 0.1; //tempo de serviço da fila 3

  mu1 = (min [Ci1, st q1])/Si1; //taxa de serviço da fila 1
  mu2 = (min [Ci2, st q2])/Si2; //taxa de serviço da fila 2
  mu3 = (min [Ci3, st q3])/Si3; //taxa de serviço da fila 3

  rot_1to2 = 0.3 * mu1;
  rot_1to3 = 0.7 * mu1;
```

A definição dos identificadores e domínios incluem todas as especificações feitas na Figura 4.6.

Definição dos eventos

```
events
  loc l1  (lambda1);    // evento associado a chegada de clientes
  loc l2  (mu2);        // evento associado a saída de clientes da fila 2
  loc l3  (mu3);        // evento associado a saída de clientes da fila 3
  syn r1to2 (rot_1to2); // evento sinc. que modela a rotação de clientes
  syn r1to3 (rot_1to3); // evento sinc. que modela a rotação de clientes
```

Os eventos associam as taxas definidas nos identificadores do bloco anterior aos eventos que irão disparar cada transição.

Função de atingibilidade

```
reachability = 1;
```

Diferentemente do modelo anterior, neste modelo tem-se todos os estados atingíveis. Dessa maneira, a função de atingibilidade é definida com 1.

Definição dos autômatos

```
network qn (continuous)
  aut q1        // fila 1
    stt c[K1]   // capacidade K1
      to (++)   // chegada de clientes
        l1      //   evento local
      to (--)   // atendimento de clientes
        e12 e13 //   eventos sincronizantes

  aut q2        // fila 2
    stt c[K2]   // capacidade K2
      to (++)   // chegada de clientes
        e12     //   evento sincronizante
      to (--)   // atendimento de clientes
        l2      //   evento local

  aut q3        // fila 3
    stt c0[K3]  // capacidade K3
      to (++)   // chegada de clientes
        e13     //   evento sincrconizante
      to (--)   // atendimento de clientes
        l3      //   evento local
```

Na descrição da rede foram definidos três autômatos (q1, q2 e q3), que, diferentemente do exemplo de compartilhamento de recursos, esses autômatos não foram replicados por terem características diferentes entre si. No entanto, neste modelo foi utilizada a replicação dos estados de cada autômato.

Para isso, introduziu-se a utilização dos operadores “++” e “--”, para definir uma transição para o estado posterior (++) e para o estado anterior (--).

Funções de integração

```
results
  n1 = st q1; //população média da fila 1
  n2 = st q2; //população média da fila 2
  n3 = st q3; //população média da fila 3
```

Solicitou-se nas funções de integração (n1, n2 e n3) a população média de cada um dos autômatos representando cada autômato uma fila do modelo.

4.3.3. Modelo de Fontes On/Off

O modelo de fontes On/Off é definido por uma fila de capacidade K alimentada por N fontes do tipo On/Off. Cada fonte On/Off se caracteriza por enviar uma quantidade constante de pacotes quando se encontra *ligada* (on) e por não enviar pacotes quando encontra-se *desligada* (off). Cada fonte é independente e pode alterar seu estado de ligada para desligada sem depender do estado em que se encontram as outras fontes. A fila, por sua vez, é caracterizada por uma taxa de chegada dependente do número de fontes ativas (no estado ligado), ou seja, quanto mais fontes ativas mais pacotes chegam a fila, que por sua vez enche mais rapidamente. Por outro lado, se nenhuma fonte estiver ligada a fila não recebe pacotes e só se esvazia. A taxa de atendimento dos pacotes na fila tem uma taxa constante não dependendo do estado da fila ou das fontes.

O modelo SAN para este exemplo compreende $N + 1$ autômatos. N autômatos modelam as fontes enquanto o último autômato modela a fila (Figura 4.8).

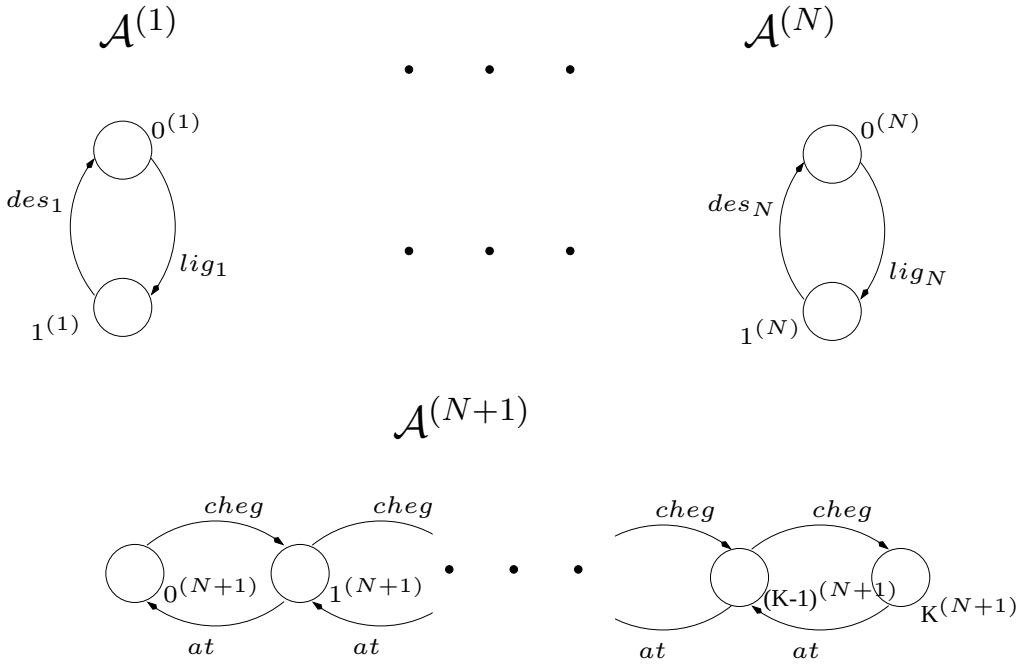


Figura 4.8: Modelo SAN **de** N **fontes On/Off e uma fila com capacidade** K**.**

Os N autômatos que modelam as fontes são replicados e cada um possui dois estados, o estado $0^{(n)}$ representa a fonte em estado *off* enquanto o estado $1^{(n)}$ representa a fonte em estado *on*. Uma fonte n é ligada ($0^{(n)} \rightarrow 1^{(n)}$) quando um evento local lig_n com taxa de ocorrência λ ocorre. De forma semelhante o evento local des_n com taxa δ provoca o desligamento da fonte ($1^{(n)} \rightarrow 0^{(n)}$).

O autômato da fila possui $K + 1$ estados, um estado ($0^{(N+1)}$) representando a fila vazia mais K estados ($1^{(N+1)}$ a $K^{(N+1)}$) representando o número de pacotes na fila. Um pacote é atendido ($k^{(N+1)} \rightarrow (k-1)^{(N+1)}$) quando ocorre um evento local at, o qual possui uma taxa de ocorrência constante μ. Da mesma maneira, a chegada de um pacote ($k^{(N+1)} \rightarrow (k+1)^{(N+1)}$) na fila é modelado pelo evento local $cheg$ com uma taxa de ocorrência funcional f. A taxa de ocorrência do evento $cheg$ é dada por uma taxa constante α, que modela a taxa de geração de pacotes por fonte, multiplicada pelo número de fontes ativas. Essa função f é expressa por:

$$f = \alpha * (nb\ [\mathcal{A}^{(0)}..\mathcal{A}^{(N)}]\ on);$$

4.3.3.1. Arquivo SAN para o PEPS

Para o modelo de fontes On/Off apresentado neste exemplo, foi criado o seguinte arquivo `san` para descrever o modelo. Adotando 20 fontes On/Off e uma fila de capacidade 50.

Definição dos identificadores e domínios

```
identifiers
  N      = [0..19]; // número de fonte On/Off replicadas
  K      = [0..50]; // capacidade da fila
  lambda = 3;       // taxa para ligar uma fonte
  delta  = 4;       // taxa para desligar uma fonte
  alfa   = 22;      // taxa de geração de pacotes por fonte
  f      = (nb [fonte[0]..fonte[19]] on) * alfa;
                    // taxa total de geração de pacotes
  mu     = 47;      // taxa de atendimento de pacotes
```

Dos sete identificadores e domínios declarados neste bloco, os dois primeiros (domínios *N* e *K*) fazem referência ao número de fontes no modelo, neste caso 20 e a capacidade total da fila, no caso 50. Os identificadores *lambda* e *delta* informam com que freqüência uma fonte alterna seu estado. Os três últimos identificadores informam a taxa individual de geração de pacotes de uma fonte quando esta se encontra ligada (*alfa*), enquanto *f* informa a taxa total de geração de pacotes, a qual varia de acordo com o número de fontes ligadas. A taxa de atendimento de pacotes pelo servidor da fila é definida pelo identificador *mu*.

Definição dos eventos

```
events
  loc lig  (lambda); // evento que liga uma fonte
  loc des  (delta);  // evento para desligar uma fonte
  loc cheg (f);      // evento que modela a chegada de um pacote
  loc at   (mu);     // evento que representa o atendimento de um pacote
```

Os dois primeiros eventos (lig e des) são associados as transições que alternam o estado de um fonte, enquanto os dois últimos eventos ($cheg$ e at) são associados as transições que alteram o estado da fila.

Função de atingibilidade

```
reachability = 1;
```

Neste modelo todos os estados são atingíveis por isso informa-se uma probabilidade 1 (*verdadeiro*) para todos os estados globais do modelo.

Definição dos autômatos

```
network onoff (continuous)
  aut fonte[N] // 20 replicas do autômato fonte
    stt off    // fonte desligada
      to(on)   // transição que liga a fonte
        lig
    stt on     // fonte ligada
      to(off)  // transição de desliga a fonte
        des

  aut fila
    stt s[K]   // o estado s é replicado 51 vezes
      to(--)   // transição de atendimento
        at
      to(++)   // transição de chegada
        cheg
```

O primeiro autômato declarado (*fonte*) é replicado 20 vezes e representa as fontes do sistema. Já o segundo autômato (*fila*) que representa a fila de atendimento não é replicado, entretanto tem o estado *s* que representa o número de pacotes esperando atendimento, replicado 51 vezes, 50 estados representando a capacidade da fila mais um estado representando a fila vazia. As transições para o estado anterior ou posterior, a partir de um estado qualquer, é representada por (--) e (++), respectivamente.

Funções de integração

```
results
  f_on   = nb [fonte[0]..fonte[19]] on; // número médio de fontes ligadas
  u_fila = st fila;                     // utilização média da fila
  p_fila = f - ((st fila !=0)*mu);      // probabilidade média de perda
```

O primeiro resultado calcula qual o número médio de fontes ligadas no sistema. Os dois últimos resultados refere-se ao autômato *fila*, indicando a utilização média da fila e a probabilidade média de perda de pacotes pela fila estar cheia.

4.3.4. *Cluster* com Protocolo de Comunicação UDP

Este último exemplo apresenta um modelo de um *cluster* de computadores conectados por um barramento único. Neste modelo, N computadores disputam o direito de transmissão no barramento. Note-se que, apesar de um único computador poder transmitir pacotes no barramento a cada vez, vários outros podem estar recebendo estes pacotes. Eventualmente os pacotes podem estar sendo perdidos, caso nenhum computador esteja recebendo. Entretanto, um computador somente poderá iniciar uma recepção se existir outro computador transmitindo naquele momento. Tal comportamento de comunicação pode caracterizar de maneira genérica o protocolo UDP [BRE 2002], por exemplo.

O modelo SAN para este exemplo é modelado por um autômato de quatro estados replicado N vezes, como é visto na Figura 4.9. Cada um dos N autômatos modela um computador conectado ao barramento. Cada computador i pode estar *ocioso*, *processando*, *transmitindo* ou *recebendo*, definindo assim os quatro estados de cada autômato, respectivamente, $0^{(i)}$, $1^{(i)}$, $2^{(i)}$ e $3^{(i)}$.

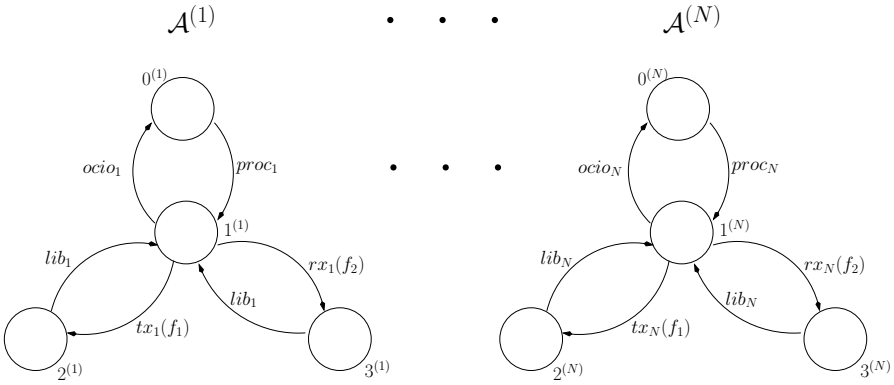


Figura 4.9: Modelo SAN para o exemplo de cluster com protocolo comunicação UDP.

Se o computador i estiver ocioso, este pode passar para processando ($0^{(i)} \rightarrow 1^{(i)}$), pelo disparo do evento local $proc_i$ com taxa de ocorrência σ. Uma vez que o computador esteja processando, este pode voltar a ficar ocioso ($1^{(i)} \rightarrow 0^{(i)}$), essa transição é feita pelo disparo do evento local $ocio_i$ que ocorre com taxa δ. Do estado processando o computador pode ainda iniciar uma transmissão de pacotes ($1^{(i)} \rightarrow 2^{(i)}$), a qual é iniciada pela ocorrência do evento tx_i disparado com taxa funcional f_1, a qual retorna um valor nulo (0.0) caso o barramento esteja ocupado e λ caso contrário. A função f_1 é expressa por:

$$f_1 = (nb\ \ transmitindo == 0) * \lambda;$$

Do estado processando, o computador i também pode iniciar a recepção de pacotes ($1^{(i)} \rightarrow 3^{(i)}$). A recepção de pacotes por um computador i é disparada pelo evento rx_i. Tal evento possui uma taxa de ocorrência funcional f_2. A função f_2 retorna μ caso exista algum computador transmitindo pacotes no barramento, caso contrário retorna um valor nulo (0.0). A função f_2 é expressa por:

$$f_2 = (nb\ \ transmitindo == 1) * \mu;$$

A taxa funcional dada pela função f_2 impede um computador qualquer inicie uma recepção de pacotes quando não existe nenhum computador transmitindo. Porém, o computador pode ficar no estado recebendo após o computador que estava transmitindo acabar a transmissão. Note porém que a taxa de liberação $alfa$, associada aos eventos locais lib, libera o meio de comunicação, tanto pela transmissão ($2^{(i)} \rightarrow 1^{(i)}$) quanto pela recepção ($3^{(i)} \rightarrow 1^{(i)}$), é a mesma, ou seja, o tempo que um computador fica transmitindo e que o tempo que outro fica recebendo é o mesmo, porém a recepção pode ser iniciada um pouco depois e também acabar um pouco depois. Isso modela a *latência* do meio de comunicação.

4.3.4.1. Arquivo SAN para o PEPS

Criou-se o seguinte arquivo `san` para este modelo de cluster com protocolo de comunicação UDP, como apresentado neste exemplo, adotando 13 computadores.

Definição dos identificadores e domínios

```
identifiers
  N      = [0..12];               // intervalo de replicações
  sigma  = 2;                     // taxa de saída do estado ocioso
  delta  = 3;                     // taxa de volta para o estado ocioso
  lambda = 6;                     // taxa de início de uma transmissão
  mu     = 10;                    // taxa de início de uma recepção
  alfa   = 6;                     // taxa de fim de transmissão/recepção
  f1     = (nb tx == 0) * lambda; // avalia se o meio está ocioso
  f2     = (nb tx == 1) * mu;      // avalia se o meio está sendo usado
```

Do conjunto de identificadores, os quais definem o domínio das réplicas (*N*), as taxas constantes de transição (*sigma*, *delta*, *lambda*, *mu* e *alfa*), dá-se destaque especial para as funções *f1* e *f2* que definem as taxas funcionais as quais restringem o acesso ao meio de comunicação.

Definição dos eventos

```
events
  loc proc  (sigma);
  loc ocio  (delta);
  loc trans (f1);
  loc rec   (alfa);
  loc lib   (f2);
```

O conjunto de eventos associa cada identificador declarado no bloco anterior a um evento. Note-se que a liberação do meio, seja por uma recepção ou por uma transmissão, dá-se com a mesma taxa *mu* associada ao evento lib, isso ocorre por que logo após o transmissor acabar a transmissão a recepção termina a recepção, mas não necessariamente ao mesmo tempo.

Função de atingibilidade

```
reachability = (nb tx <= 1);
```

A função de atingibilidade de modelo define como estados atingíveis todos estados globais cujo número de autômatos em *tx* é menor ou igual a 1.0.

Definição dos autômatos

```
network udp (continuous)
  aut comp[N]     // o autômato computador é replicado N vezes
    stt ocioso    // computador está ocioso
      to(pr)      // sai do ócio
        proc
    stt pr        // computador está processando
      to(ocioso)  // volta para o ócio
        ocio
      to(tx)      // vai transmitir
        trans     // impede transmissão se o meio já está usado
      to(rx)      // vai receber
        rec       // impede recepção se ninguém está transmitindo
    stt tx        // computador está transmitindo
      to(pr)      // libera o meio e volta a processar
        lib
    stt rx        // computador está recebendo
      to(pr)      // acaba recepção e volta a processar
        lib
```

Para esse modelo definiu-se apenas um autômato replicado 13 vezes, visto que os computadores na rede têm as mesmas características. Como já mencionado, a restrição de acesso ao meio é feita pelas duas taxas funcionais associadas aos eventos rec e $trans$, os quais, por sua vez, estão associados as transições de recepção e transmissão, respectivamente.

Funções de integração

```
results
   u_meio = nb tx; // uso médio do meio de comunicação
   u_pr   = nb pr; // número meio de computadores processando
```

As duas funções buscam descobrir os gargalos[9] do sistema. A função *u_meio* analisa a utilização média do meio de comunicação enquanto a função *u_pr* analisa o uso dos processadores.

4.4. Gramática da Ferramenta PEPS

A ferramenta de software PEPS (*Performance Evaluation of Parallel Systems*) é uma ferramenta acadêmica para descrição e resolução de modelos SAN. A primeira versão da ferramenta foi proposta por Plateau, Fourneau e Lee [PLA 88] em 1988, desde então a ferramenta tem incorporado novos métodos e facilidades, como por exemplo, métodos de solução iterativa, funções integrações, definição do modelo em alto nível entre outros.

[9]Considera-se um gargalo do sistema a parte do sistema que mais impede o fluxo dos dados.

A gramática para o PEPS2003 é um formato para representação do formalismo SAN, definido através de um arquivo textual.

Será apresentada aqui apenas uma visão superficial da gramática para o PEPS2003, sem entrar no domínio de cada termo, uma descrição mais detalhes pode ser encontrada em [BEN 2003].

O formato de arquivo reconhecido pelo PEPS2003 pode ser dividido em alguns blocos (*identifiers*, *events*, *reachability*, *network* e *results*), como é apresentado a seguir.

4.4.1. Declaração de Identificadores e Domínios

Este primeiro bloco contém as declarações e a inicialização dos identificadores (constantes ou funções) e domínios (intervalos contínuos) que serão utilizados no modelo SAN.

Caso não necessite definir num identifidor ou domínio para uso no modelo, este bloco pode ser omitido. Estes identificadores e domínios podem ser utilizados para representar as taxas de ocorrência e as probabilidades de rotação.

Declaração de Identificadores e Domínios - PEPS2003

Identifiers
identificador = <*expressão*>;
identificador1 = <*expressão*>;
...
dominio = [<*inicio*> .. <*fim*>];
dominio1 = [<*inicio*> .. <*fim*>];
...

4.4.2. Declaração de Eventos

O bloco de declaração de eventos define todos os eventos usados no modelo. Para cada evento deve ser define o tipo de evento (*loc* ou *syn*), o nome do evento e a taxa de disparo associada. A taxa de disparo pode ser tanto um identificador declarado no bloco anterior como um valor numérico qualquer.

Declaração de Eventos - PEPS2003

events
loc *nome do evento* [*dominio*] (<*identificador*>);
syn *nome do evento1* [*dominio1*] (<*identificador1*>);
...

4.4.3. Função de Atingibilidade

Neste bloco é modelada a função de atingibilidae que definirá qual é o espaço de estados atingíveis do modelo SAN. A função de atingibilidade é um função booleana que retorna 1 caso o estado possa ser atingido e 0 caso não possa.

Função de Atingibilidade - PEPS2003

partial reachability = <*expressão*>;

A palavra reservada **partial** define que a expressão utilizada para definir o conjunto de estados atingíveis, não indica todos os estados, apenas uma parte deles. A ferramenta PEPS tem condições de descobrir os outros estados atingíveis a partir do conjunto definido e gerar a função de atingibilidade completa.

4.4.4. Descrição do Modelo

A parte de definição do modelo propriamente dito, é feito no bloco de *Descrição do Modelo*. Este bloco tem uma estrutura hierarquizada onde são descritos os autômatos pertencentes à rede. Além disto, neste bloco também é definido o nome do modelo e a escala de tempo utilizada.

Descrição do Modelo - PEPS2003
network *nome do modelo* (*tipo do modelo*) Descrição dos Autômatos

Os tipos de modelos aceitos pela gramática são *discrete* e *continuous*, porém a ferramenta PEPS2003 resolve apenas modelos à escala de tempo contínua.

4.4.4.1. Descrição dos Autômatos

Para cada autômato é especificado o nome do autômato e o número de replicas (caso não haja réplica do autômato, não é necessário informar este parâmetro) dentro da rede. São descritos também os estados em que o autômato pode estar.

Descrição dos Autômatos - PEPS2003
aut *nome do autômato* [*número de réplicas*] Descrição dos Estados ...

4.4.4.2. Descrição dos Estados

A *Descrição dos Estados* descreve cada estado dentro do autômato. Deve definir-se o nome do estado e, caso existe, a recompensa e o número de replicações deste estado, dentro do autômato. Descreve-se também as transições deste estado para outro.

Descrição dos Estados - PEPS2003
stt *nome do estado* [*número de réplicas*] (*recompensa*) Descrição das Transições ...

4.4.4.3. Descrição das Transições

A *Descrição das Transições* define a transição entre os estados do autômato. Dois tipos de transições são definidas no PEPS2003. O primeiro tipo de transição assume como estado origem o estado que está sendo declarado. Esse tipo de transição é definido pela palavra reservada **to** e deve indicar o estado destino e os eventos associados a ela. O segundo tipo de transição é definido inicialmente pela palavra **from**, que indica o estado

origem, após utiliza-se a palavra **to** para definir o estado destino e os eventos associados a transição. Este segundo tipo de transição é utilizado no caso de replicações de estados, quando necessita-se uma transição para/de um estado especifico do conjunto de replicações.

Descrição das Transições - PEPS2003

to (*nome do estado destino*)
Descrição dos Eventos
...
from (*nome do estado origem*)
to (*nome do estado destino*)
Descrição dos Eventos
...

4.4.4.4. Descrição dos Eventos

Neste nível os eventos declarados anteriormente são associados a uma transição. Caso haja probabilidades de rotação diferentes de 1.0 estas são definidas entre parênteses após nome do evento.

Descrição de Eventos - PEPS2003

nome do evento
nome do evento (*probabilidade*)
...

Para todos os tipos de eventos pode-se omitir a probabilidade de rotação quando está for igual a 1.0.

É interessante ressaltar que nem todos eventos precisam aparecer em todas as transições, mas cada transição deve ter pelo menos um evento associado.

4.4.5. Descrição dos Resultados

Neste bloco são definidas as funções para obtenção dos índices de desempenho interessantes ao modelo. Estes índices são descritos através de um nome e de uma expressão matemática.

Descrição dos Resultados - PEPS2003

Results
nome da função = <*expressão*>;
...

4.4.6. Defi nição das Expressões

A gramática utilizada pela ferramenta PEPS2003 define vários operadores matemáticos, lógicos e relacionais. A ferramanta PEPS2003 não define ordem de precedência entre os operadores e a avaliação das expressões é feita da esquerda para a direita, para definir precedência nas operações usa-se parênteses.

Os operadores matemáticos aceitos são “+”, “-”, “*”, “/”, “min” e “max”. Para operadores relacionais a ferramenta PEPS2003 suporta os operadores “==”, “!=”, “>”, “<”, “>=” e “<=”. Da mesma forma os seguintes operadores lógicos são aceitos pela ferramenta, “e” com o código “&&”, “ou” com “||”, “negação” com “!”, “ou exclusivo” por “^”, “implicação” usando “->” e “dupla implicação” como “<->”.

Além dos operadores matemáticos, lógicos e relacionais já conhecidos, a ferramenta PEPS2003 define primitivas para operação com autômatos. A definição semântica destes operadores é a seguinte:

- **st** <*id_aut*>: retorna o estado corrente do autômato *id_aut*, onde *id_aut* é o identificador do autômato;
- **nb** <*id_stt*>: retorna o número total de autômatos do modelo SAN que se encontram no estado <*id_stt*>, onde <*id_stt*> é o identificador de um estado;
- **nb** [<id_aut> .. <id_aut>] <*id_stt*> : retorna o número de autômatos no estado <id_stt> no intervalo [<id_aut> ..<id_aut>];
- **rw** <*id_aut*>: retorna a recompensa associada ao estado do autômato <*id_aut*>, caso não haja recompensa associada a funcão **rw** <*id_aut*> é idêntica a função **st** <*id_aut*>;
- **sum_rw** [<*id_aut*> .. <*id_aut*>]: retorna o somatório das recompensa associadas aos estados correntes dos autômatos do intervalo [<*id_aut*>.. <*id_aut*>];
- **sum_rw** [<*id_aut*>.. <*id_aut*>] <*id_stt*> : retorna o somatório das recompensa associadas aos autômatos do intervalo [<*id_aut*>.. <*id_aut*>] que se encontram no estado <*id_stt*>;
- **prod_rw** [<*id_aut*>.. <*id_aut*>] ou **prod_rw** <*id_stt*> [<*id_aut*>.. <*id_aut*>]: funcionam de maneira semelhante ao **sum_rw**, porém retornam o produtório invés do sumátorio.

4.5. Considerações Finais

O enfoque principal deste curso foi a utilização do formalismo de Redes de Autômatos Estocásticos (SAN) no desenvolvimento de modelos paralelos. Entretanto as vantagens de SAN não se restringem apenas ao nível de modelagem. É possível empregar uma série de técnicas para a resolução eficiente dos modelos. Cabe lembrar que entende-se como resolução de modelos estocásticos de avaliação de desempenho a obtenção de probabilidades estimadas para as diversas situações em que o sistema possa se encontrar. No caso de modelos baseados em Cadeias de Markov, como nas SAN, a solução se expressa através do vetor de probabilidade dos estados.

Em comparação a outros formalismos, como por exemplo, Cadeias de Markov [STE 94] e Redes de Petri Estocásticas [MAR 95], a grande vantagem do formalismo SAN é a utilização de álgebra tensorial generalizada [FER 98a] que permite a obtenção de um formato tensorial (descritor Markoviano) que se mostra extremamente econômico face a outros métodos de armazenamento e resolução de sistemas.

Por se tratar de um formalismo relativamente recente, as SAN continuam sendo um fértil campo de pesquisa, sempre englobando novos conceitos e técnicas para armazenamento e soluções de sistemas.

4.6. Bibliografia

[BEN 2003] BENOIT, A. et al. The PEPS Software Tool. In: INTERNATIONAL CONFERENCE ON MODELLING TECHNIQUES AND TOOLS FOR COMPUTER PERFORMANCE EVALUATION, 13., 2003, Urbana and Monticello, Illinois, USA. **Proceedings...** Springer-Verlag, 2003.

[BRE 2002] BRENNER, L.; FERNANDES, P.; DE ROSE, C. A. F. An analytical model to evaluate the performance of cluster architecture. In: THE LINUX CLUSTERS: THE HPC REVOLUTION 2002 CONFERENCE, 2002, St. Petersburg, Florida, USA. **Anais...** [S.l.: s.n.], 2002. p.1–11.

[FER 98] FERNANDES, P.; PLATEAU, B.; STEWART, W. Efficient descriptor-vector multiplication in stochastic automata networks. **Journal of the ACM**, v.45, n.3, p.381–414, 1998.

[FER 98a] FERNANDES, P. **Méthodes numériques pour la solution de systèmes markoviens à grand espace d'états**. 1998. Tese (Doutorado em Ciência da Computação) — Institut National Polytechnique de Grenoble, Grenoble.

[HOP 79] HOPCROFT, J.; ULLMAN, J. **Introduction to automata theory, languages and computation**. [S.l.]: Addison-Welsey, 1979.

[MAR 95] MARSAN, M. A. et al. **Modelling with generalized stochastic petri nets**. [S.l.]: John Wiley and Sons, 1995.

[PLA 91] PLATEAU, B.; ATIF, K. Stochastic automata networks for modelling parallel systems. **IEEE transactions on software engineering**, v.17, n.10, p.1093–1108, 1991.

[PLA 88] PLATEAU, B.; FOURNEAU, J.; LEE, K. Peps: a package for solving complex markov models of parallel systems. In: INTERNATIONAL CONFERENCE ON MODELLING TECHNIQUES AND TOOLS FOR COMPUTER PERFORMANCE EVALUATION, 4., 1988, Palma, Spain. **Proceedings...** [S.l.: s.n.], 1988.

[PLA 84] PLATEAU, B. **De l´evaluation du parallélisme et de la synchronisation**. 1984. Tese (Doutorado em Ciência da Computação) — Paris-Sud, Orsay.

[STE 94] STEWART, W. J. **Introduction to the numerical solution of markov chains**. [S.l.]: Princeton University Press, 1994.