Técnicas de modelagem para a análise de desempenho de processos de negócio

Kelly Rosa Braghetto

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Doutor em Ciências

Programa: Ciência da Computação

Orientador: Prof. Dr. João Eduardo Ferreira

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq, da FAPESP e da CAPES

São Paulo, julho de 2011

Técnicas de modelagem para a análise de desempenho de processos de negócio

Esta tese trata-se da versão original
da aluna Kelly Rosa Braghetto.

Agradecimentos

Agradeço ao meu orientador, o professor João Eduardo Ferreira, por toda a atenção e paciência, pelos vários aconselhamentos, pelo apoio nos momentos adversos e por compartilhar do meu entusiasmo pela pesquisa na área de modelagem de processos de negócio.

Agradeço ao professor Jean-Marc Vincent por ter me acolhido no *Laboratoire de Informatique de Grenoble* (LIG) durante o meu estágio de doutorado na França, pela generosidade com que compartilhou seus valiosos conhecimentos na área de análise de desempenho e por sua colaboração intensiva na orientação deste trabalho.

Agradeço aos professores Roberto Cesar Jr., Ronaldo Hashimoto e Roberto Hirata Jr., que integraram a banca do meu exame de qualificação, pelos comentários e sugestões sobre a minha proposta inicial de trabalho. As observações feitas por eles me ajudaram a melhor direcionar minhas pesquisas.

Agradeço aos meus amigos Fernanda Almeida, Jesús Mena-Chalco, Marcos Broinizi e Pedro Paulo da Silva, pela nossa agradável convivência no IME–USP. Em especial, agradeço a Pedro Takecian e Márcio Oikawa, que sempre estiveram mais próximos, pelos vários aconselhamentos que me deram e pelas inúmeras discussões técnicas que tivemos.

Agradeço também aos colegas do LIG, com quem convivi durante um ano e meio, por terem tornado a experiência de estudar em outro país ainda mais rica. Em especial, agradeço a Afonso Sales e Leonardo Brenner, que diretamente me auxiliaram em minha pesquisa por meio do seu *expertise* em SAN.

Agradeço aos meus amigos Ana Beatriz Graciano, Eliane Matsuda, Felipe Rosário, Gustavo Halasi, Jonatas de Moraes, Selma Shimono, Silvia Ferreira e Tissiano da Silva – por entremear em minha rotina de trabalho/estudos momentos de "vida normal".

Agradeço especialmente a Daniel Cordeiro (apesar de eu crer que não há agradecimentos suficientes para ele), pelo seu apoio incansável durante todos esses anos do meu doutorado e por participar tão ativamente em minha vida, de tantas maneiras distintas.

Finalmente, agradeço aos meus familiares, em especial aos meus pais Aparecida e Laerte Braghetto, pelo apoio incondicional que sempre me devotaram.

Resumo

As recentes pesquisas na área de Gestão de Processos de Negócio (GPN) vêm contribuindo para aumentar a eficiência nas organizações. A GPN pode ser compreendida como o conjunto de métodos, técnicas e ferramentas computacionais desenvolvidas para amparar os processos de negócios. Tipicamente, a GPN é fundamentada por *modelos de processos*. Esses modelos, além de permitir a automação da configuração e execução, aumentam a analisabilidade dos processos de negócio.

Apesar de auxiliar os especialistas de negócio nas diferentes fases envolvidas no ciclo de vida de um processo de negócio, os modelos definidos em linguagens específicas de domínio, como a BPMN (*Business Process Model and Notation*), não são os mais apropriados para amparar a fase de análise. De formal geral, esses modelos não possuem uma semântica operacional formalmente definida (o que limita o seu uso para a verificação e validação dos processos) e nem mecanismos para quantificar o comportamento modelado (o que impossibilita a *análise de desempenho*).

Neste trabalho de doutorado, nós desenvolvemos um arcabouço que ampara e automatiza os principais passos envolvidos na análise de desempenho de processos de negócio via modelagem analítica. Nós estudamos a viabilidade da aplicação de três formalismos Markovianos na modelagem de processos de negócio: as *Redes de Petri Estocásticas*, as *Álgebras de Processo Estocásticas* e as *Redes de Autômatos Estocásticos* (SAN, do inglês *Stochastic Automata Networks*). Escolhemos SAN como formalismo base para o método proposto neste trabalho.

Nosso arcabouço é constituído por: (i) uma notação para enriquecer modelos de processos de negócio descritos em BPMN com informações sobre o seu gerenciamento de recursos, e (ii) um algoritmo que faz a conversão automática desses modelos não-formais de processos para modelos estocásticos em SAN. Com isso, somos capazes de capturar o impacto causado pela contenção de recursos no desempenho de um processo de negócio. A partir de um modelo em SAN gerado com o nosso arcabouço, podemos predizer variados índices de desempenho que são boas aproximações para o desempenho esperado do processo de negócio no mundo real.

Palavras-chave: Processos de Negócio, Modelagem, Análise de Desempenho, Redes de Autômatos Estocásticos.

Abstract

Recent results in the research field of *Business Process Management* (BPM) are contributing to improve efficiency in organizations. BPM can be seen as a set of methods, techniques and tools developed to support business processes in their different requirements. Usually, the BPM techniques are based on a *process model*. In addition to enabling automated process configuration and execution, these models also increase the analizability of business processes.

Despite being able to support business specialists in different phases of the life cycle of a business process, the models created in domain-specific languages, such as BPMN (*Business Process Model and Notation*), are not the most appropriated ones to support the analysis phase. Generally, these models have neither a formally defined operational semantics (which hinders their use for *verification* and *validation*), nor mechanisms to quantify the modeled behavior (which hinders their use for *performance analysis*).

In this PhD research, we developed a framework to support and to automatize the main steps involved in the analytical modeling of business processes aiming performance evaluation. We studied the viability of applying three Markovian formalisms in business process modeling: *Stochastic Petri Nets*, *Stochastic Process Algebras* and *Stochastic Automata Networks* (SAN). We have chosen SAN to support the method proposed in this work.

Our framework is composed of: (i) a notation to enrich BPMN business process models with information concerning the associated resource management and (ii) an algorithm that automatically converts these non-formal business process models in SAN stochastic models. With this, we are able to capture the impact caused by resource contention in the performance of a business process. From a model generated through our framework, we are able to extract varied performance indices that are good approximations for the expected process performance in the real world.

Keywords: Business Processes, Modeling, Performance Analysis, Stochastic Automata Networks.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

APE Álgebra de Processos Estocástica

ATC Álgebra Tensorial Clássica

ATG Álgebra Tensorial Generalizada

BPMN *Business Process Model and Notation*

CMTC Cadeia de Markov de Tempo Contínuo

CMTD Cadeia de Markov de Tempo Discreto

CSP *Calculus of Sequential Processes*

CR Conjunto de Recursos

FNC Forma Normal Conjuntiva

FND Forma Normal Disjuntiva

GD Grafo de Derivação

GPN Gestão de Processos de Negócio

GSPN *Generalized Stochastic Petri Nets*

PEPA *Performance Evaluation Process Algebra*

PEPS *Performance Evaluation of Parallel Programs*

RdP Rede de Petri

RdPE Rede de Petri Estocástica

RdPT Rede de Petri Temporizada

RDR Requisitos Disjuntos de Recursos

RR Requisitos de Recursos

RRS Requisito de Recurso Simples

SAN *Stochastic Automata Networks*

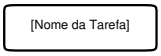
SMART *Stochastic Model checking Analyzer for Reliability and Timing*

WfMS *Workflow Management Systems*

UML *Unified Modeling Language*

Lista de Símbolos

BPMN 2.0

	Evento de Início
	Evento de Fim
	Tarefa Atômica
	Desvio Condicional Exclusivo (Divisão/Junção **OU-Exclusivo**)
	Ativação Incondicional em Paralelo (Divisão/Junção **E**)
	Ativação Inclusiva Condicional (Divisão/Junção **OU**)
	Anotação de Texto
	Fluxo de Sequência
	Associação

SAN

$\mathcal{A}$	Um Autômato
$\mathcal{A}[1..n]$	Um Conjunto de n Réplicas do Autômato $\mathcal{A}$
e	Um Evento
$e[1..n]$	Um Conjunto de n Réplicas do Evento e
$st(\mathcal{A})$	Devolve o Estado Corrente do Autômato $\mathcal{A}$
$nb(\mathcal{A}[1..n], s)$	Devolve o Número de Autômatos em $\mathcal{A}[1..n]$ que estão no Estado s

Capítulo 1

Introdução

1.1 Contexto e Motivação

As recentes pesquisas na área de Gestão de Processos de Negócio (GPN) vêm contribuindo para aumentar a eficiência nas organizações. A GPN pode ser compreendida como um conjunto de métodos, técnicas e ferramentas computacionais desenvolvidas para amparar as diferentes fases do ciclo de vida de um processo de negócio [vdAtHW03]. Ela contribui para a compreensão de como os processos organizacionais funcionam e de como eles podem ser otimizados.

Os sistemas de apoio à GPN atuais podem ser vistos como uma "evolução natural" dos sistemas de gerenciamento de *worflows* (WfMS, do termo inglês *Workflow Management Systems*). Quando os primeiros WfMS apareceram na década de 80, o seu principal objetivo era a automação de processos. Mas com a evolução das linguagens e ferramentas de apoio criadas especificamente para esse domínio de aplicação, a análise dos processos operacionais quase negligenciada nos WfMS começou a receber mais atenção com o surgimento dos sistemas de apoio à GPN.

Apesar da grande diversidade, as tecnologias computacionais existentes atualmente para o apoio à gestão de processos possuem uma base operacional comum: um *modelo de processo* [BRvU00]. Além de permitir a automação de sua configuração e execução, um modelo aumenta significativamente a analisabilidade de um processo de negócio.

Os processos de negócio podem ser analisados sob duas perspectivas distintas: a *qualitativa* e a *quantitativa*. Podemos citar como exemplos de análise qualitativa a *validação* e a *verificação*. A primeira avalia se um processo se comporta como o esperado (correção semântica), enquanto a segunda avalia se um processo respeita alguns critérios estruturais (correção sintática).

O exemplo mais representativo de análise quantitativa é a *análise de desempenho*. A análise de desempenho de processos de negócio avalia a habilidade do processo em satisfazer requisitos não funcionais relacionados a diferentes índices, como, por exemplo, tempos de execução, tempos de espera, taxas de utilização de recursos e níveis de serviço. Ela nos permite melhorar a qualidade de serviço de um processo de negócio por meio da identificação de ineficiências, tais como as causadas por uma má definição do fluxo de tarefas ou pelo mal provisionamento dos recursos (e.g., gargalos na execução do processo ou recursos ociosos).

As tarefas de um processo de negócio geralmente dependem de recursos para serem executadas. Na prática, esses recursos são finitos e precisam ser compartilhados entre as diferentes instâncias do processo de negócio que podem estar concorrentemente em execução. Sendo assim, o desempenho de um processo de negócio está intrinsecamente associado ao seu gerenciamento de recursos.

Muitos esforços têm sido feitos para padronizar as linguagens de modelagem de processos de negócio, visando a interoperabilidade das ferramentas desenvolvidas para a área de GPN. O resul-

tado mais importante desses esforços é a *Business Process Model and Notation* (BPMN) [OMG11b], uma notação padrão para a representação gráfica de processos de negócio desenvolvida pela *Business Process Management Initiative* (BPMI) [BPM] e mantida pelo *Object Management Group* (OMG) [OMG].

Apesar de auxiliar os especialistas de negócio nas diferentes fases envolvidas no ciclo de vida de um processo de negócio, os modelos em BPMN não são os mais apropriados para amparar a fase de análise. Assim como as demais linguagens específicas de domínio cujos conceitos inspiraram sua criação, a BPMN não possui uma semântica formalmente definida e, por essa razão, não é adequada para a validação e verificação de modelos.

Além disso, modelos em BPMN não possuem elementos para a quantificação do esforço automatizado ou manual necessário para se desempenhar as tarefas que compõem um processo de negócio. Essa deficiência impossibilita o uso direto de modelos em BPMN para a avaliação de desempenho.

As técnicas para a análise de desempenho de sistemas computacionais se baseiam em três abordagens distintas: *modelagem analítica*, *simulação* e *medição*. Cada uma dessas abordagens possui os seus prós e contras. A viabilidade de cada uma delas depende do contexto de aplicação e dos objetivos da análise.

O tema deste trabalho de doutorado é a análise de desempenho de processos de negócio via modelagem analítica. Além de possuir um caráter preditivo, a modelagem analítica fornece uma boa percepção dos efeitos causados pela variação dos parâmetros do sistema e por suas interações [Jai91]. Essas características são particularmente interessantes no domínio da GPN.

Modelos analíticos geralmente baseiam-se em modelos estocásticos que, em muitos casos, supõe-se que são processos de Markov. A maior limitação dos modelos Markovianos é o conhecido problema da explosão do espaço de estados associado a modelos de processos complexos ou de grande escala. Modelos com grandes espaços de estados implicam dificuldades computacionais para o cálculo da solução numérica (*e.g.*, elevados requisitos em termos tanto de consumo de memória quanto de tempo de processamento). Esse problema restringe a aplicação de modelos Markovianos na análise de muitos sistemas existentes no mundo real.

Formalismos Markovianos de alto nível foram criados com o objetivo de amenizar a complexidade da modelagem e da análise de sistemas representados como processos de Markov. Eles permitem que um sistema seja modelado em um nível de abstração maior que o provido pelas cadeias de Markov. Frequentemente, esses formalismos contam com técnicas de extração de medidas de desempenho a partir dos modelos sem mesmo passar pela geração de suas cadeias de Markov subjacentes, o que torna a análise mais eficiente.

Como frequentemente acontece em outros tipos de sistemas paralelos e distribuídos, os processos de negócio são difíceis de serem manualmente modelados usando os métodos Markovianos tradicionais, devido à complexidade de seus requisitos. Mas são ainda mais difíceis de serem analisados, devido ao seu potencialmente grande espaço de estados.

Neste trabalho de doutorado, nós desenvolvemos um arcabouço que ampara e automatiza os principais passos envolvidos na análise de desempenho de um processo de negócio via modelagem analítica. Esses passos, que aparecem sintetizados no processo sequencial da Figura 1.1, são:

1. a criação do modelo de processo de negócio usando uma linguagem específica do domínio (a BPMN);

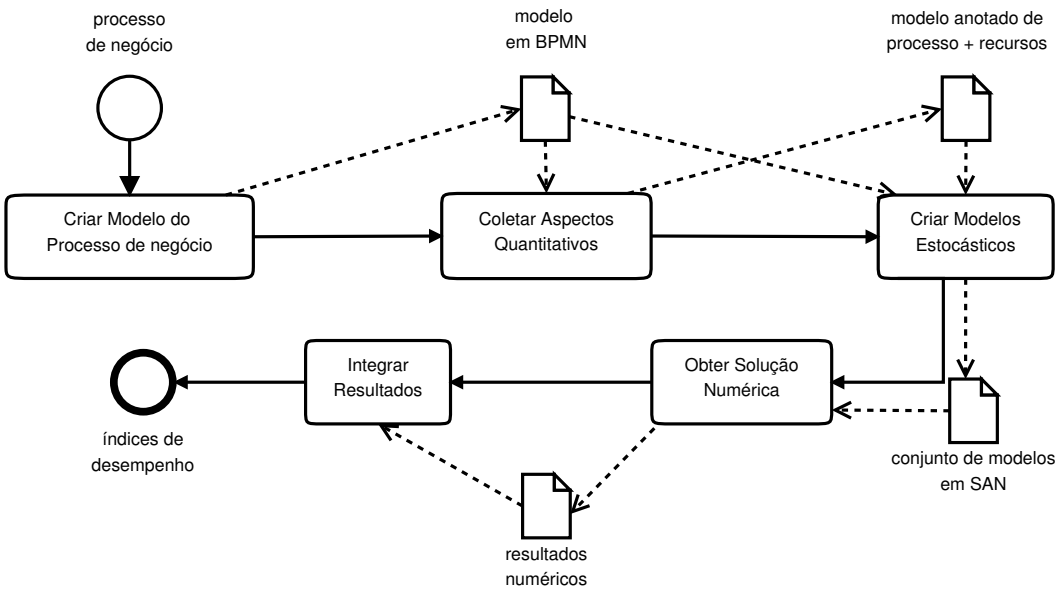


Figura 1.1: *Passos envolvidos na análise de desempenho de processos de negócio via modelagem analítica.*

2. a coleta de aspectos quantitativos sobre o modelo o processo de negócio. Essa coleta possui duas finalidades distintas:

   - acrescentar informações aos elementos estruturais já especificados no passo 1. Por exemplo, associar às tarefas o seu tempo médio de execução, associar a cada fluxo de controle alternativo uma probabilidade de ocorrência, determinar o número médio de instâncias do processo criadas por unidade de tempo, etc.;
   - incluir novos elementos que, por limitações das linguagens específicas de domínio, não puderam ser diretamente modelados no passo 1 (como, por exemplo, a descrição dos recursos necessários para a realização das tarefas).

   Esses dados quantitativos podem ser provenientes de observações e medições realizadas sobre o sistema real (quando esse já está implementado e correntemente em uso) ou do conhecimento prévio de especialistas sobre o domínio de negócio;

3. a criação de modelos estocásticos que combinam os dados estruturais (especificados no passo 1) e os dados quantitativos (coletados no passo 3) do processo de negócio;

4. a resolução numérica dos modelos estocásticos, para a obtenção das probabilidades dos estados do modelo no regime estacionário;

5. a extração de índices de desempenho a partir dos resultados obtidos no passo 4.

Ao longo deste texto nós discutimos os detalhes envolvidos nesses passos e propomos abordagens para o tratamento dos mesmos.

1.2 Definição do Problema e Objetivos

O problema fundamental tratado nesta tese é o mapeamento de modelos não formais de processos de negócio em modelos que possuam as propriedades matemáticas necessárias para que

realizemos análises numéricas de desempenho. Como indicado nos passos da Figura 1.1, esse mapeamento depende de informações que caracterizam quantitativamente um processo de negócio e que não estão representadas em seu modelo inicial.

Para tratar esse problema, nós combinamos informações de dois aspectos distintos de um processo de negócio (o controle do fluxo de tarefas e o gerenciamento de recursos) para gerar de forma automática modelos estocásticos capazes de predizer o desempenho do processo de forma precisa. Nessa abordagem, a criação do modelo estocástico de um processo de negócio possui duas etapas distintas:

1. a representação do controle de fluxo de tarefas (expresso no modelo do processo de negócio) em termos dos elementos existentes no formalismo estocástico;

2. a adição das informações quantitativas sobre o processo ao modelo no formalismo estocástico.

Neste trabalho, nós sustentamos que:

- um modelo bem definido em BPMN de um processo de negócio pode ser automaticamente convertido em um modelo equivalente em Redes de Autômatos Estocásticos (SAN, do inglês *Stochastic Automata Networks*).

- sabendo quais são os recursos requeridos por cada uma das tarefas de um processo de negócio e sabendo como o acesso a esses recursos é controlado, nós podemos obter de forma automática boa parte dos parâmetros necessários para caracterizar quantitativamente o comportamento de um processo de negócio em seu modelo em SAN.

SAN [Pla85, PA91] é uma técnica estocástica Markoviana usada para modelar de forma estruturada sistemas que possuem um grande espaço de estados. Diferentemente de outras técnicas de análise Markovianas que requerem a geração de uma matriz de transição de estados, a representação interna de um modelo em SAN permanece compacta mesmo quando o número de estados de sua cadeia de Markov subjacente começa a explodir. Para explorar essa representação de espaço de estados enxuta, vários métodos de solução numérica de modelos em SAN foram desenvolvidos visando à eficiência computacional em termos de consumo de memória e de tempo de processamento [SAP95, Fer98].

Neste trabalho, buscamos atender os seguintes objetivos específicos:

- determinar as condições necessárias para que a conversão de um modelo em BPMN para um modelo em SAN possa ser feita;

- criar um algoritmo que, dado como entrada um modelo em BPMN que respeite as condições levantadas no item anterior, devolva um (ou mais) modelo(s) em SAN em que o comportamento modelado seja equivalente ao do modelo de entrada;

- definir mecanismos para a especificação de informações referentes ao gerenciamento de recursos de um processo de negócio, de forma a enriquecer seu modelo em BPMN;

- desenvolver um método para adicionar as informações de gerenciamento de recursos a um modelo em SAN de processo de negócio;

- definir como se obter índices de desempenho para a análise de processos de negócio a partir dos modelos em SAN gerados pelo nosso método de conversão.

1.3 Contribuições

As principais contribuições deste trabalho de doutorado são:

- a avaliação da viabilidade da modelagem de processos de negócio em três diferentes formalismos Markovianos utilizados para a análise de desempenho: as Redes de Petri Estocásticas Generalizadas (GSPN, do termo inglês *Generalized Stochastic Petri Nets*), a Álgebra de Processos para Avaliação de Desempenho (PEPA, do termo inglês *Performance Evaluation Process Algebra*) e as Redes de Autômatos Estocásticos (SAN). Esses formalismos foram aplicados na modelagem de diferentes cenários de processos de negócio e avaliados de acordo com critérios bem definidos.

  Os resultados obtidos nessa avaliação foram primeiramente publicados como um relatório técnico [BFV09] e depois nos anais do *Symposium On Theory of Modeling and Simulation – DEVS Integrative M&S Symposium* (DEVS'10), que integrou a *Spring Simulation Multiconference* (SpringSim'10) [BFV10];

- a definição de uma notação para enriquecer modelos de processos de negócio com informações sobre o gerenciamento de seus recursos.

  No contexto do arcabouço de modelagem proposto por este trabalho, o gerenciamento de recursos define: (i) quais são os recursos requeridos no processo de negócio, (ii) quantos eles são, (iii) quais são suas capacidades de trabalho, e (iv) como e por quais tarefas eles são acessados;

- a criação de um algoritmo para a conversão automática de modelos em BPMN, anotados com informações sobre gerenciamento de recursos, para modelos estocásticos em SAN. Os modelos gerados nessa conversão levam em consideração os principais fatores que podem impactar o desempenho de um processo de negócio (e.g., a contenção por recursos e a carga de trabalho do sistema). Por essa razão, esses modelos são capazes de gerar índices de desempenho que se aproximam de forma satisfatória dos esperados para o processo no mundo real.

  A notação e o algoritmo criados foram objetos de três publicações: um relatório técnico do IME-USP [BFV11a], que foi posteriormente estendido e resultou em um artigo aceito para o *8th European Performance Engineering Workshop* – EPEW 2011 [BFV11b], e um artigo submetido ao *International Journal of Innovative Computing, Information and Control* (IJICIC), para a edição especial intitulada *Intelligent and Innovative Computing in Business Process Management* [BFV11c];

- a implementação do algoritmo de conversão, que resultou em uma ferramenta computacional chamada `BP2SAN` [BP2]. Com essa ferramenta, os especialistas de negócio podem obter modelos de análise de desempenho a partir de modelos de processos de negócio sem a necessidade de conhecer os conceitos estatísticos envolvidos em uma modelagem estocástica.

1.4 Trabalhos Relacionados

Dentre todas as linguagens, técnicas e ferramentas de software relacionadas à GPN, os modelos de processos são os elementos que mais recebem atenção da comunidade científica. Isso se deve

à sua habilidade para auxiliar um processo de negócio durante todo o seu ciclo de vida (projeto, implementação, gerenciamento e análise). Tendo em mente a importância dos modelos de processo na área de GPN, a decisão de estudá-los, estendê-los ou convertê-los com o objetivo de aprimorar sua analisabilidade é natural.

As técnicas geralmente utilizadas na modelagem de processos de negócio (e.g., BPMN, *Unified Modeling Language* (UML) [OMG10], *Event-driven Process Chain* (EPC) [STA05], etc.) não possibilitam diretamente a análise formal. Além disso, elas priorizam a perspectiva de controle de fluxo ou fornecem uma visão de gerenciamento de recursos que é insuficiente para a realização de análises de desempenho. Muitos trabalhos (como [DDO08, WG08, Esh02, vdA99]) já trataram da conversão de modelos de processos de negócio em formalismos como as redes de Petri e as álgebras de processos, com o objetivo da verificação e validação de modelos. Outras abordagens (como [Rei03, PQZ08, LGMC04, CGH+03, OLAR09]) dedicam-se à conversão de modelos de processos de negócio em modelos estocásticos (como as Redes de Petri Estocásticas e as Álgebras de Processos Estocásticas), com o objetivo da análise quantitativa dos modelos. Nesta seção, nós apresentamos brevemente alguns desses trabalhos relacionados à modelagem estocástica de processos de negócio.

O trabalho de Canevet et al. [CGH+03] propôs um mapeamento automático de *diagramas de estado* da UML, anotados com informações de desempenho, para modelos em PEPA. Essas informações de desempenho às quais os autores se referem são probabilidades associadas a estados e taxas associadas a transições do modelo UML. Uma vantagem importante da abordagem proposta pelos autores é que os resultados de desempenho obtidos pela solução do modelo em PEPA podem ser refletidos de volta para o modelo em UML. Entretanto, a abordagem não permite a definição de taxas funcionais, o que impede que alguns aspectos importantes relacionados ao desempenho do processo sejam representados no modelo.

A proposta de Prandi et al. [PQZ08] é um mapeamento de modelos em BPMN para modelos em *Calculus for Orchestration of Web Services* (COWS), um cálculo de processos inspirado na linguagem *Business Process Execution Language* (BPEL). Os autores fizeram uma breve discussão sobre o uso de uma extensão estocástica do formalismo, para possibilitar a análise quantitativa dos processos de negócio modelados em COWS. Apesar de ser baseada em um formalismo composicional, a COWS estocástica não explora a vantagem da composicionalidade no seu método de análise. Por essa razão, o formalismo sofre do mesmo problema da explosão do espaço de estados que limita o uso de outros formalismos Markovianos na análise de sistemas de grande escala.

Sauer and Chandy [SC81] afirmam que a contenção por recursos geralmente é um fator muito significativo no desempenho de um sistema e também é o mais difícil de ser quantificado. Por essa razão, é importante que a técnica de análise de desempenho possua facilitadores para a modelagem desse fator. Algumas abordagens obrigam o projetista a especificar todos os requisitos de recursos e as políticas de acesso a recursos no modelo do processo de negócio de forma explícita, para que o modelo estocástico seja capaz de prover uma análise de desempenho precisa.

Reijers [Rei03] criou um método para a análise de desempenho de *Workflow Nets* (WF-nets), uma subclasse das Redes de Petri especificamente criada para a modelagem de *workflows*. Esse método definiu uma extensão estocástica para as WF-nets, a *Resource-Extended SWF*, que inclui elementos para a representação de recursos e de múltiplas instâncias nas redes.

O trabalho de Oliveira et al. [OL09, OLAR09] também considera a contenção por recursos na

modelagem para análise de desempenho dos processos, mas utiliza as GSPN como formalismo. Na proposta dos autores, o modelo do processo de negócio deve ser construído por meio da composição de blocos de comportamento bem definidos, frequentemente empregados na descrição de *workflows*. Tarefas podem ser associadas a recursos finitos e um recurso pode tratar diferentes tipos de tarefas. O modelo considera múltiplas instâncias em atendimento no sistema.

Contudo, tanto na proposta de Reijers quanto na de Oliveira et al., os recursos são sempre considerados de acesso exclusivo e os índices de desempenho são extraídos do modelo por meio de simulação em vez de resolução numérica.

A abordagem proposta nesta tese para o tratamento dos recursos na modelagem dos processos de negócio objetiva ser mais flexível e abrangente que a encontrada nos trabalhos relacionados. Além de considerar recursos de acesso exclusivo, nós prevemos também a modelagem de recursos que funcionam sob uma política de acesso de tempo compartilhado. E com a notação que definimos para a especificação do gerenciamento de recursos em modelos de processos de negócio, é possível definir requisitos complexos de recursos.

1.5 Organização do Texto

No Capítulo 2, introduzimos os conceitos necessários para a compreensão do restante do texto. Nele, discutimos as noções envolvidas na modelagem de processos de negócio e na modelagem analítica para a avaliação de desempenho de sistemas. Também definimos a sintaxe e semântica das linguagens e formalismos de modelagem empregados neste trabalho.

O Capítulo 3 estuda a viabilidade da aplicação de três formalismos estocásticos (GSPN, PEPA e SAN) na modelagem de diferentes cenários de processos de negócio. O estudo realizado evidencia os prós e contras de cada formalismo no domínio de aplicação da GPN e fundamenta a escolha pelo uso de SAN para o desenvolvimento de nossa proposta.

A primeira parte do arcabouço de modelagem proposto neste trabalho de doutorado está definida no Capítulo 4, que trata da conversão automática de modelos em BPMN para modelos em SAN. Nesse capítulo, nós definimos formalmente as estruturas de dados empregadas na representação dos modelos e as operações associadas a elas, além dos mapeamentos elementares entre objetos de BPMN e submodelos em SAN. Após essas definições auxiliares, o algoritmo de conversão é especificado em pseudo-código.

O Capítulo 5 apresenta a segunda parte do arcabouço proposto neste trabalho: a modelagem do gerenciamento de recursos em processos de negócio. Nele, definimos uma notação para complementar modelos em BPMN com a descrição do seu gerenciamento de recursos. Na sequência, definimos um método que considera as informações anotadas nos modelos em BPMN para gerar modelos em SAN que expressam o gerenciamento de recursos associado ao processo de negócio e que, por essa razão, são capazes de fornecer medidas de desempenho que melhor se aproximam das esperadas para o processo no mundo real.

A extração de índices de desempenho a partir dos modelos em SAN gerados pelo nosso arcabouço é discutida no Capítulo 6. Exemplos são dados para ilustrar os resultados que podem ser obtidos por meio da análise numérica dos modelos estocásticos.

Finalmente, no Capítulo 7 discutimos os resultados obtidos neste trabalho e a proposta para trabalhos futuros.

O texto também contém dois anexos. No Anexo A, introduzimos as Cadeias de Markov em

Tempo Contínuo (CMTCs), que são a base dos formalismos empregados neste trabalho. No referido anexo, discutimos os principais conceitos relacionados à definição e à análise das CMTCs. No Anexo B encontra-se uma definição sucinta dos operadores da Álgebra Tensorial Clássica (ATC) e de sua extensão, a Álgebra Tensorial Generalizada (ATG), que é usada na representação interna dos modelos em SAN.

Capítulo 2

Conceitos Preliminares

Neste capítulo, nós apresentamos de forma concisa alguns conceitos relacionados à modelagem de processos de negócio e à modelagem analítica para a avaliação de desempenho de sistemas. Esses conceitos auxiliam a compreensão dos demais capítulos deste texto.

Aqui, também definimos a sintaxe e semântica das linguagens e formalismos de modelagem empregados neste trabalho (i.e., BPMN, GSPN, PEPA e SAN).

2.1 Modelagem de Processos de Negócio

Um processo de negócio é descrito por um ou mais procedimentos que, em conjunto, realizam um objetivo de negócio. A execução de um processo de negócio possui condições muito bem definidas de início e término, e pode combinar procedimentos automáticos e manuais [WfM99].

Um processo de negócio pode ser modelado sob diferentes perspectivas. De acordo com vários trabalhos sobre o assunto [Rus07, BRvU00, CKO92], as perspectivas mais relevantes são:

- *controle de fluxo* – descreve as tarefas pertencentes ao processo e sua ordem (parcial) de execução por meio de diferentes construtores de composição. As tarefas podem ser divididas em dois tipos : (i) *elementares*, representando unidades atômicas de trabalho, e (ii) *compostas*, modularizando a ordem de execução de um conjunto de tarefas;

- *dados* – entrelaça dados da lógica do negócio ao controle de fluxo do processo. Esses dados podem ser documentos ou outros objetos que são passados de uma tarefa para outra, ou variáveis locais do processo usadas para expressar pré ou pós condições para a execução de uma tarefa;

- *organizacional* (também chamada de *recursos*) – atrela ao processo uma estrutura organizacional, por meio da definição de papéis (desempenhados por pessoas ou equipamentos) responsáveis pela execução das tarefas;

- *tratamento de exceções* – lida com as causas das exceções e as ações que precisam ser tomadas nos seus tratamentos.

Grande parte dos trabalhos relacionados à modelagem de processos de negócio são direcionados à perspectiva de controle de fluxo. A prevalência dessa perspectiva é compreensível, dado que o controle de fluxo fornece uma visão sobre a especificação de um processo de negócio que é essencial para a avaliação de sua efetividade [vdAtHKB03]. As demais perspectivas assumem um papel secundário, uma vez que elas oferecem uma visão complementar da estrutura do processo.

Um outro importante fator associado à prevalência da perspectiva de controle de fluxo no desenvolvimento da área de GPN é um fator histórico, relacionado às tecnologias de *workflow*. Até o final dos anos 90, as tecnologias de *workflow* priorizavam a automação dos processos e, consequentemente, modelos que pudessem amparar essa automação.

Sob a perspectiva do controle de fluxo de tarefas, os processos de negócio são um tipo especial de sistemas concorrentes. Eles podem ser compostos por um grande número de tarefas e o relacionamento de precedência existente entre elas define uma ordem parcial que deve ser respeitada nas execuções das instâncias dos processos. A habilitação de uma tarefa para execução pode estar condicionada a alguma condição lógica. Uma mesma tarefa pode aparecer em diferentes lugares de um mesmo modelo. Tarefas e subprocessos podem ser repetitivos; um modelo de processo pode conter ciclos estruturados ou não-estruturados. Os processos de negócio podem possuir estruturas complexas de ramificação e junção de fluxos. Na maioria dos casos, eles podem ser modelados de forma estrutural, como a interação de subprocessos mais simples.

Neste trabalho, utilizamos a *Business Process Model and Notation* (BPMN) como linguagem para a especificação do controle de fluxo de tarefas dos processos de negócio.

2.1.1 *Business Process Model and Notation* (BPMN)

A *Business Process Model and Notation* (BPMN) [OMG11b] é um padrão mantido pelo *Object Management Group* (OMG) [OMG] para a modelagem de processos de negócio. O objetivo principal da BPMN é funcionar como uma "ponte padronizada para cobrir o vão existente entre o projeto dos processos de negócio e sua implementação" [OMG11b]. Ela provê uma notação comum para os usuários de negócio envolvidos nas diferentes fases do ciclo de vida de um processo de negócio: projeto, implementação, gerenciamento, monitoração e análise. A especificação da BPMN agrega as melhores práticas da comunidade de modelagem de processos de negócio e os melhores conceitos existentes em outras notações já consagradas, como as *Event-Process Chains* (EPC) e os diagramas de atividades da *Unified Modeling Language* (UML), entre outras.

Em BPMN, nós podemos construir três tipos de diagramas: *diagramas de colaboração*, *diagramas de processo* e *diagramas de coreografia*. Como neste trabalho estamos interessados em modelar e analisar o impacto dos requisitos de recursos no desempenho dos processos de negócio, nos restringiremos aos diagramas de processos de BPMN, pois eles concentram boa parte das informações de que necessitamos para criar nossos modelos de análise de desempenho.

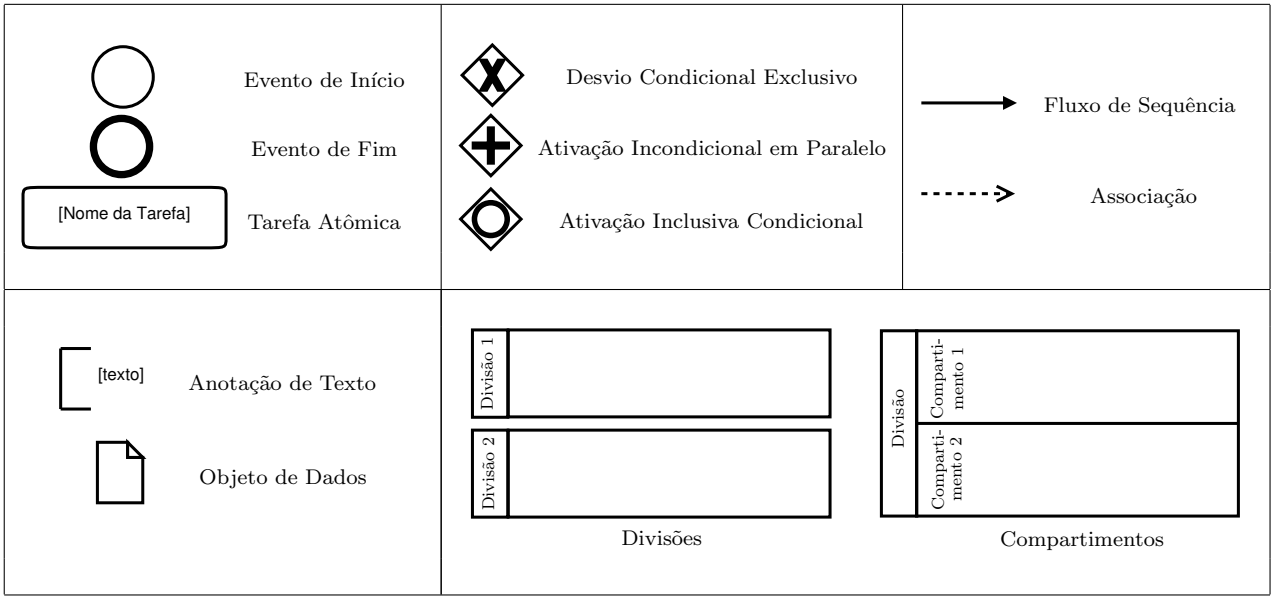
De acordo com o documento de especificação da BPMN, "um processo descreve uma sequência ou fluxo de tarefas em uma organização que tem como objetivo a realização de um trabalho". Um diagrama de processo é um grafo dirigido constituído por diferentes elementos (tarefas, eventos, desvios e fluxos de sequência) que definem uma semântica de execução finita.

Neste trabalho, nós utilizaremos somente uma subclasse dos modelos que podem ser representados como um diagrama de processo. A Tabela 2.1 mostra os objetos BPMN que podem constituir os modelos considerados neste trabalho [1]. Esses objetos são muito importantes na modelagem de processos. Com eles somos capazes de definir uma classe vasta de modelos de processos de negócio.

Na sequência, descrevemos esses objetos gráficos de acordo com as definições encontradas no documento de especificação da BPMN.

[1] Os nomes em português para os objetos de BPMN empregados neste texto foram extraídos da versão oficial em português do pôster da linguagem [Ofe].

Tabela 2.1: *Objetos básicos de fluxo e conexão em BPMN.*



Objetos da Notação

Um *evento* é algo que acontece durante o curso de execução de um processo e que afeta o fluxo do mesmo. O *evento de início* indica onde o processo começa, enquanto o *evento de fim* indica onde o processo termina.

Uma *tarefa* é um termo genérico para um trabalho realizado no processo. Uma tarefa pode ser atômica ou composta. Mas, neste trabalho, sempre que utilizarmos o termo *tarefa* estaremos nos referindo a uma tarefa atômica.

Um *fluxo de sequência* é usado para mostrar a ordem na qual as tarefas devem ser realizadas.

Um *desvio* é usado para controlar a divergência ou convergência de fluxos de sequência. Existem diversos tipos de desvios e cada tipo pode afetar tanto os fluxos de entrada (caso da convergência) quanto os fluxos de saída (caso da divergência). Neste trabalho, utilizamos os seguintes tipos de desvios:

- *Desvio Condicional Exclusivo.* Um *desvio condicional exclusivo divergente* é usado para criar caminhos alternativos dentro do fluxo do processo. Em uma dada instância do processo, apenas um dos caminhos alternativos pode ser seguido. Um *desvio condicional exclusivo convergente* é usado na junção de caminhos alternativos. Cada fluxo de sequência de entrada é roteado para o fluxo de sequência de saída sem nenhuma sincronização.

  Na terminologia dos modelos de *workflow*, o desvio condicional exclusivo equivale a uma *divisão/junção OU-Exclusivo*;

- *Ativação Incondicional em Paralelo.* Uma *ativação incondicional em paralelo divergente* cria caminhos paralelos sem verificar nenhuma condição. A *ativação incondicional em paralelo convergente* espera todos os fluxos de entrada antes de disparar a execução do(s) fluxo(s) de sequência de saída.

  Na terminologia dos modelos de *workflow*, a ativação incondicional em paralelo equivale a uma *divisão/junção E*;

- *Ativação Inclusiva Condicional.* Uma *ativação inclusiva condicional divergente* pode ser usada para criar caminhos alternativos e paralelos dentro de um fluxo de processo. Diferentemente do desvio condicional exclusivo, todas as expressões de condição dos fluxos de saída são avaliadas. A avaliação de uma condição verdadeira não elimina a avaliação das demais condições. Todos os fluxos de sequência cuja expressão de condição tiver sido avaliada como verdadeira serão habilitados para execução paralelamente. Sendo assim, todas as combinações de caminhos podem ser seguidas (incluindo a opção de não seguir nenhum caminho ou seguir todos eles ao mesmo tempo). Uma *ativação inclusiva condicional convergente* é usada para unir uma combinação de caminhos alternativos e paralelos. Um fluxo de sequência de entrada quando chega em uma convergência desse tipo deve ser sincronizado com outros fluxos de sequência que chegarão posteriormente.

  Na terminologia dos modelos de *workflow*, a ativação inclusiva condicional equivale a uma *divisão/junção OU*.

Uma *associação* é usada para ligar informações aos elementos gráficos. As *anotações de texto* fornecem informações adicionais aos leitores dos diagramas BPMN. Os *objetos de dados* proveem informações sobre os dados dos quais as tarefas precisam para serem executadas e/ou sobre os dados produzidos por elas.

As *divisões* e *compartimentos* representam os participantes de um processo. Eles são as entidades responsáveis pelas tarefas e podem ser uma organização, um papel, um ator humano ou um sistema automatizado. Os compartimentos podem subdividir uma divisão ou um outro compartimento.

As anotações, os objetos de dados, as divisões e os compartimentos são elementos para melhorar a legibilidade do modelo. Eles não determinam o controle de fluxo do processo modelado.

Outros objetos existentes em BPMN podem ser representados em termos dos objetos da Tabela 2.1. Por exemplo, *tarefas repetitivas* e *tarefas com múltiplas instâncias* podem ser modeladas por meio de tarefas atômicas, desvios condicionais exclusivos e ativações incondicionais em paralelo.

A Figura 2.1 mostra um exemplo de processo de negócio modelado em BPMN. Esse processo representa a venda de viagens de uma agência de turismo.

Instanciação e Término dos Processos

Um processo é instanciado quando um dos eventos de início ocorre. Mas, antes de introduzir a noção de término de uma instância, nós precisamos apresentar o conceito de *ficha* em BPMN.

Uma ficha é um conceito teórico usado no documento de especificação de BPMN [OMG11b] como um apoio para a definição informal da semântica operacional dos objetos existentes na notação. O comportamento dos objetos no processo pode ser definido pela descrição de como eles interagem com a ficha conforme ela "atravessa" a estrutura do processo. Essa ideia de usar fichas para determinar a dinâmica de um modelo vem do formalismo das Redes de Petri.

Cada ocorrência de um evento de início cria uma ficha no seu fluxo de sequência de saída, que é continuado conforme a semântica descrita para os outros objetos do processo. Por exemplo, o objeto de ativação incondicional em paralelo é habilitado se houver pelo menos uma ficha em cada fluxo de sequência de entrada. O elemento consome exatamente uma ficha de cada fluxo de sequência de entrada e produz exatamente uma ficha em cada sequência de fluxo de saída. Com

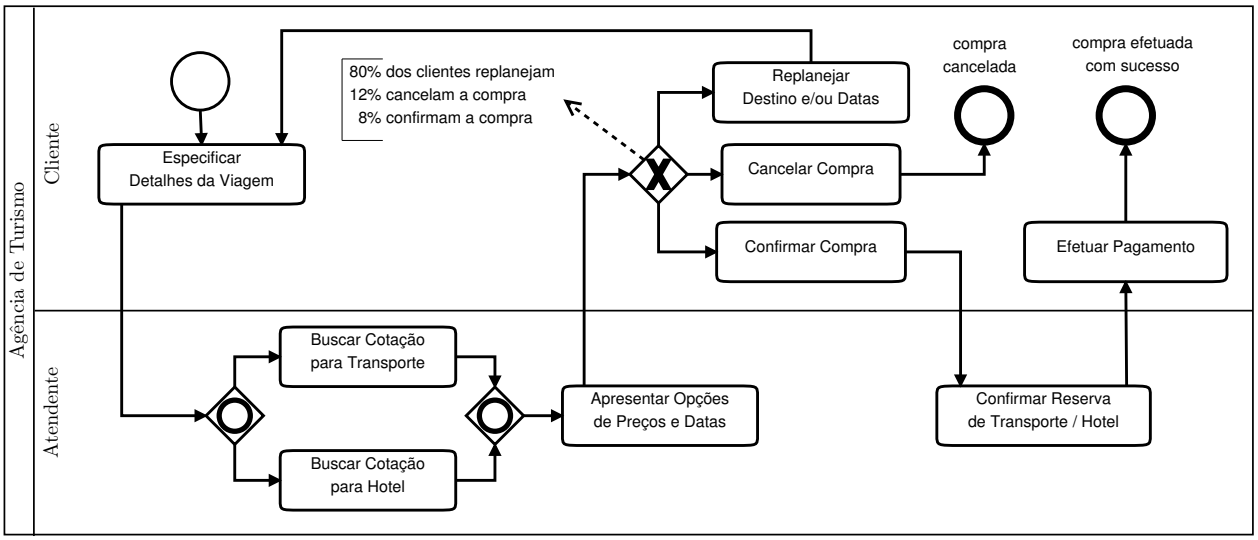


Figura 2.1: *Exemplo de processo de negócio modelado em BPMN.*

isso, podemos considerar que uma instância de processo foi terminada quando não existem mais fichas dentro da instância do processo.

2.1.2 Gerenciamento de Recursos

Nas técnicas e ferramentas atuais da área de GPN, o controle de fluxo de um processo de negócio é modelado abstraindo alguns aspectos que impactam de forma significativa o comportamento dinâmico do processo. Eis alguns aspectos frequentemente negligenciados:

- uma tarefa de um processo de negócio pode depender de um ou mais recursos para ser realizada;
- múltiplas instâncias de um mesmo processo de negócio podem estar simultaneamente ativas e, por essa razão, podem disputar o acesso aos recursos;
- um recurso pode realizar mais de um tipo de tarefa;
- os recursos podem colaborar entre si para realizar conjuntamente uma tarefa.

Em BPMN, nós podemos associar *papéis de recurso* às tarefas. Segundo o documento de especificação de BPMN [OMG11b], o recurso é quem vai desempenhar a tarefa ou ser responsável por ela. Além disso, o recurso "pode ser especificado na forma de um indivíduo específico, um grupo, uma função ou posição em uma organização, ou mesmo uma organização". Como podemos observar, essa definição é consideravelmente restritiva quando comparada com o conceito geral de recurso nas diferentes subáreas da Ciência da Computação.

A perspectiva de recursos (i.e., a modelagem de recursos e sua interação com os sistemas de *workflow*) foi o objeto de estudo de Russel et al. [RvdAtHE05]. Os autores descreveram um conjunto de *padrões de recursos em workflows* com o objetivo de catalogar as várias formas em que os recursos são representados e utilizados nos sistemas de *workflow*. Esses padrões se referem principalmente a recursos humanos e são usados como base para a comparação de linguagens de modelagem e sistemas de gerenciamento para processos de negócio.

Como neste trabalho estamos interessados na avaliação de desempenho, não podemos abstrair o gerenciamento de recursos dos modelos de processos de negócio. No nosso contexto de aplicação, um recurso é qualquer entidade requerida para a realização de uma tarefa do modelo. Ele pode ser tanto uma entidade física (processadores, memória, impressora, máquinas, pessoas, organizações, etc.) quanto uma entidade virtual (bibliotecas de software, serviços *web*, bancos de dados, etc.).

É importante também ressaltarmos que o tempo de execução de uma tarefa pode estar relacionado com os seus requisitos de recursos e, dessa forma, variar com a carga de trabalho do sistema. Por exemplo, considere um serviço *web* que depende de um servidor *web* para ser executado. Se considerarmos que o servidor pode processar somente uma requisição por vez, o tempo de execução do serviço *web* não será impactado pela carga de trabalho no servidor. Se uma nova requisição chegar enquanto o servidor está ocupado, a nova requisição será colocada em uma fila de espera, para ser tratada posteriormente. Mas, em uma suposição mais realística, o servidor pode tratar todas as requisições que recebe de forma paralela. Nesse caso, a capacidade de processamento do servidor será dividida entre as requisições em tratamento e o tempo de execução do serviço *web* vai variar em função da carga de trabalho atual no servidor.

2.1.3 Análise Qualitativa × Análise Quantitativa

A medida da qualidade de serviço pode estar associada a fatores que variam de acordo com o tipo de sistema e os requisitos dos usuários, podendo, assim, estar relacionada a aspectos qualitativos e quantitativos do sistema [FM03].

A análise qualitativa busca responder questões como as seguintes:

- o modelo possui estados de impasse (*deadlocks*)?
- o modelo possui estados inatingíveis?
- o sistema se comporta como o esperado?
- o sistema sempre termina de forma apropriada?
- um dado sistema A equivale a um outro dado sistema B?

Na análise quantitativa, estamos interessados em responder questões como as seguintes:

- com que probabilidade o comportamento de um sistema corresponde à sua especificação?
- quanto tempo leva para que o sistema chegue em um dado estado de interesse?
- com que probabilidade uma dada propriedade de interesse é satisfeita no sistema?

Segundo Dongen e Mendling [vDMvdA06], todo modelo de processo de negócio que é usado no desenvolvimento de sistemas corporativos deveria ser formalmente analisado, para a identificação de erros ou ineficiências em potencial. Problemas na fase da modelagem podem acarretar impactos severos nos lucros de uma empresa, especialmente quando o funcionamento de um processo de negócio chave é comprometido. Os autores atribuem duas razões principais à dificuldade de se detectar erros em modelos de processos de negócio:

1. a grande variedade de linguagens existentes para a modelagem de processos de negócio – cada linguagem possui uma sintaxe e uma semântica própria e, por essa razão, as técnicas de análise precisam ser definidas de forma individual;

2. a ausência de semântica formal – a maioria das linguagens para a modelagem de processos de negócio ou não possuem uma semântica formal definida, ou somente possuem essa semântica quando certas condições estruturais são respeitadas na construção do modelo.

Essa dificuldade motivou o desenvolvimento de diversos trabalhos de pesquisa como os citados na Seção 1.4, que se dedicam ao uso de formalismos para amparar a análise qualitativa ou quantitativa de processos de negócio. A maior vantagem do uso de modelagem formal é que ela constitui um arcabouço preciso e não-ambíguo para o raciocínio sobre os processos.

2.2 Avaliação de Desempenho de Sistemas

A *análise de desempenho* é o tipo mais empregado de avaliação quantitativa de sistemas. Ela se relaciona à descrição, à análise e à otimização do comportamento dinâmico e dependente de tempo dos sistemas [HHK02].

A análise de desempenho nos permite lidar com problemas frequentemente encontrados em sistemas computacionais. Jain [Jai91] agrupa esses problemas nos seguintes tópicos: a comparação de sistemas, a identificação de gargalos, a caracterização de cargas de trabalho, a configuração de sistemas e a previsão de desempenho.

Existem três abordagens para a avaliação de desempenho discutidas na literatura da área. Cada uma dessas abordagens possui vantagens e desvantagens. Para escolher a abordagem mais apropriada para um caso específico, é importante considerarmos os recursos disponíveis para a análise do problema e o nível desejado de precisão dos resultados. Essas abordagens são:

- *modelagem analítica* – utiliza modelos matemáticos para descrever e analisar numericamente determinados aspectos de interesse em um sistema. Prós: ela pode ser preditiva e relativamente rápida, além de prover um boa indicação do impacto dos parâmetros no sistema. Contras: modelar todos os aspectos relevantes para se determinar o desempenho de um sistema pode não ser uma tarefa trivial; muitas das suposições e simplificações feitas na modelagem podem impactar negativamente o nível de precisão dos resultados da análise;

- *simulação* – também se baseia em um modelo, mas a solução do modelo nesse caso é dada por um programa que simula o comportamento modelado. Prós: ela pode ser preditiva e pode levar em consideração mais detalhes que os modelos analíticos. Contras: simulações são difíceis de serem validadas e muito custosas em termos de tempo de execução;

- *medição* – é feita sobre o sistema real, por meio de códigos-fonte instrumentados, software de medição dedicado, hardware de medição dedicado, ou ainda uma mistura dos anteriores. Prós: ela é potencialmente precisa. Contras: ela possui alto custo de implementação; ela não pode ser usada em todas as fases do ciclo de vida de um sistema (já que depende de uma implementação); a seleção da carga de trabalho apropriada para avaliar o sistema pode ser uma tarefa complicada.

Neste trabalho de doutorado, empregamos técnicas de modelagem analítica para predizer o desempenho de processos de negócio. Nesse contexto, as principais ferramentas de modelagem são os *processos estocásticos*.

Um processo estocástico modela um sistema estocástico. Segundo Ross [Ros97], um processo estocástico "é uma família de variáveis aleatórias que descreve a evolução no tempo de algum processo (físico)". Mais formalmente, um processo estocástico $\{X_t, t \in T\}$ é uma coleção em que, para todo $t \in T$, X_t is uma variável aleatória. Se t é interpretado como o tempo, podemos considerar que X_t é o estado do processo no tempo t. Em um *processo em tempo discreto*, T é um conjunto enumerável, ao passo que em um *processo em tempo contínuo*, T é um intervalo de números reais.

As características de um processo estocástico dependem da classe de distribuições aleatórias incorporada na descrição do sistema. Quando variáveis aleatórias com distribuições arbitrárias são usadas no modelo, o processo estocástico subjacente frequentemente é intratável. Entretanto, vários trabalhos já estudaram e demonstraram a viabilidade do uso de modelos com distribuições exponenciais na avaliação de desempenho de sistemas. A partir desses modelos é possível se derivar uma Cadeia de Markov de Tempo Contínuo (CMTC) de maneira direta. Por essa razão, modelos com distribuições exponenciais são a base da maioria das metodologias de avaliação de desempenho existentes atualmente [Her98].

Um processo Markoviano (ou seja, um processo estocástico que possui uma CMTC subjacente) respeita a *propriedade de Markov*, também conhecida como *propriedade da falta de memória*. A propriedade de Markov determina que o estado futuro de um processo depende somente do seu estado atual e é independente dos estados anteriores.

Como exemplos de técnicas de modelagem Markovianas empregadas na avaliação de desempenho de sistemas computacionais, podemos citar as *Redes de Filas*, as *Redes de Petri Estocásticas* e as *Álgebras de Processos Estocásticas*. Essas técnicas serão discutidas na Seção 2.3. Uma descrição sucinta das propriedades das CMTCs encontra-se no Apêndice A.

A modelagem é apenas o primeiro passo envolvido na avaliação de desempenho via modelagem analítica. O passo seguinte, que resultará nos índices de desempenho de interesse, é a análise numérica do modelo. No caso dos formalismos Markovianos, existem dois tipos de análise que podem ser feitas sobre os modelos: *análise em regime estacionário* e *análise transiente*. A análise em regime estacionário nos dá a *distribuição de probabilidades estacionária* do sistema, ou seja, a fração do tempo que o sistema gasta em cada um dos seus estados quando ele atinge o equilíbrio. Contrariamente à análise em regime estacionário que está interessada no comportamento a longo prazo do sistema, a análise transiente investiga o comportamento transiente do processo, ou seja, ela é capaz de determinar o estado do sistema no final de um certo intervalo de tempo, o tempo entre a ocorrência de dois eventos, etc.

A análise de sistemas estocásticos pode ter em vista não somente propriedades quantitativas, mas também propriedades qualitativas [BK08]. Dado que esses sistemas possuem probabilidades associadas aos seus estados, é possível usar essas probabilidades para verificar propriedades qualitativas como, por exemplo, as que declaram que um certo evento ocorre quase certamente (ou seja, ocorre com probabilidade próxima de 1) ou que o evento quase nunca ocorre (ou seja, ocorre com probabilidade próxima de 0).

2.2.1 Métricas de Desempenho para Processos de Negócio

De acordo com Jain [Jai91], quando um sistema realiza um serviço corretamente, o seu desempenho pode ser medido pelo tempo necessário para a realização do serviço, pela taxa na qual o serviço é realizado e pelos recursos consumidos na realização do serviço. Essas três "dimensões" do desempenho de um sistema são chamadas de *responsividade*, *produtividade* e *utilização*, respectivamente. As métricas comumente empregadas para medir essas dimensões são, respectivamente, o *tempo de resposta*, o *rendimento* e a *utilização*. A definição clássica de cada uma dessas métricas é a seguinte:

- *tempo de resposta* – é o intervalo de tempo entre o recebimento de uma requisição de usuário e o envio da resposta correspondente do sistema;
- *rendimento* – é a taxa (em requisições por unidade de tempo) na qual as requisições são servidas pelo sistema;
- *utilização* (de um recurso) – é a fração do tempo na qual o recurso está ocupado tratando de requisições.

Se considerarmos os elementos de um modelo de processo de negócio (como discutido na Seção 2.1) e tomarmos como base essas três métricas comumente usadas, podemos definir três categorias diferentes de métricas que podem ser usadas na avaliação de desempenho de processos de negócio: métricas aplicadas sobre o processo como um todo, métricas aplicadas sobre as tarefas e métricas aplicadas sobre os recursos. A Tabela 2.2 define as métricas aplicadas a processos de negócio usadas neste trabalho.

Na avaliação de desempenho, frequentemente buscamos um único número que nos dê a característica chave de um conjunto de medidas. As três medidas mais popularmente empregadas para sintetizar uma amostra são a *média*, a *mediana* e a *moda*. Na Estatística, essas medidas são chamadas de *índices de tendências centrais* [Lil05].

Neste texto, sempre que nos referirmos a um *índice de desempenho*, estaremos nos referindo a um valor que sintetize uma das métricas definidas na Tabela 2.2 na análise de um processo de negócio.

2.3 Técnicas para a Modelagem Analítica de Desempenho

As *Redes de Filas* [LZGS84] são uma das técnicas de modelagem para análise de desempenho mais populares na área de Computação. Uma Rede de Filas modela um sistema que pode ser resumido pelos seguintes passos: (i) um cliente chega no sistema e entra em uma fila, (ii) o cliente espera pelo atendimento, (iii) o cliente recebe o serviço e (iv) ou o cliente entra em uma outra fila (para um novo atendimento) ou ele deixa a rede. Dessa forma, em um dado momento do tempo, vários clientes podem estar paralelamente em atendimento nos diferentes serviços que o sistema pode ter. Esse tipo de modelo é apropriado para expressar o comportamento assíncrono dos clientes. Mas, nas Redes de Filas, a única forma de interação entre os elementos do sistema que pode ser modelada de forma direta é a contenção no acesso aos serviços [FFN91].

Para podermos modelar estruturas mais sofisticadas de separação e junção de fluxos que dependem de sincronização, como as existentes nos modelos de processos de negócio, precisamos de outras técnicas para modelagem analítica que possuam maior expressividade.

Tabela 2.2: *Métricas para a avaliação de desempenho de processos de negócio.*

Alcance	Métrica	Significado
Processo	Tempo de Serviço	Tempo necessário para a execução completa de uma instância do processo.
	Rendimento	Número de instâncias de processo completadas por unidade de tempo.
Tarefa	Tempo de Execução	Tempo necessário para a execução da tarefa.
	Tempo de Espera	Tempo que a tarefa espera pela disponibilidade dos recursos necessários para a sua execução.
	Rendimento	Número de execuções da tarefa realizadas por unidade de tempo.
Recurso	Utilização	Fração do tempo da execução do processo durante a qual o recurso está em uso.
	Rendimento	Número de unidades de trabalho realizadas pelo recurso por unidade de tempo.
	Tamanho da Fila	Número de tarefas que aguardam a disponibilidade do recurso.

As cadeias de Markov nos permitem descrever arquiteturas e sistemas complexos, sem restrições nos esquemas de sincronização envolvidos. Contudo, para o uso de cadeias de Markov explicitamente na modelagem, precisamos enumerar todos os possíveis estados do sistema e todas as possíveis transições entre eles. Essa abordagem, além de ser muito susceptível a erros, se torna inviável para sistemas com mais de uma centena de estados.

Além das Redes de Filas, existem vários outros formalismos de alto-nível criados para amenizar a complexidade da modelagem e da análise de sistemas que podem ser representados por CMTCs.

Nesta seção, nós introduzimos três formalismos Markovianos que atendem aos requisitos necessários para a modelagem de processos de negócio: as Redes de Petri Estocásticas Generalizadas (GSPN), a Álgebra de Processos para Avaliação de Desempenho (PEPA) e as Redes de Autômatos Estocásticos (SAN).

2.3.1 Redes de Petri Estocásticas

A teoria das Redes de Petri (RdPs) é um dos exemplos mais conhecidos de teoria de ordem parcial para modelagem e análise de sistemas concorrentes. São muito utilizadas devido à sua representação gráfica de fácil compreensão e ao seu potencial matemático para a análise de modelos. Essas análises incluem verificações de propriedades inerentes aos sistemas concorrentes, como relações de precedência entre eventos, sincronização e existência ou não de impasses [Mur89].

As RdPs são formadas basicamente por dois tipos de componentes: (i) *transições* representando

ações do sistema; (ii) *lugares* representando variáveis de estado do sistema.

Uma RdP com marcação é formalmente definida pela tupla $RdP = \{L, T, A, P, m_0\}$, em que:

- $L = \{l_1, l_2, ..., l_L\}$ é um conjunto de lugares;
- $T = \{t_1, t_2, ..., t_T\}$ é um conjunto de transições;
- $A \subseteq (L \times T) \cup (T \times L)$ é o conjunto de arcos;
- $P : A \to \mathbb{N}$ é a função peso dos arcos;
- $m_0 = \{m_{01}, m_{02}, ..., m_{0L}\}$ é a marcação inicial da rede.

Os conceitos associados a cada um dos itens de uma RdP são:

- *lugar* (representado graficamente por um círculo) – modela uma condição que deve ser satisfeita para que o disparo da transição seja realizado;
- *transição* (representada graficamente por um retângulo ou barra) – pode ser compreendida como uma ação ou evento;
- *arco orientado* – liga um lugar a uma transição ou vice-versa, encadeando condições e eventos;
- *marca* ou *ficha* – representa um recurso disponível. O posicionamento dessas fichas nos lugares do grafo constitui a marcação da RdP. Cada lugar pode possuir 0 ou mais fichas. A evolução da marcação permite modelar o comportamento dinâmico do sistema;
- *peso* – cada arco possui um peso associado a ele; o peso indica quantas fichas uma transição consome de um lugar de entrada ou quantas fichas uma transição acrescenta em um lugar de saída. Quando um arco não possui um peso explicitamente indicado no grafo, considera-se que o seu peso é 1.

Uma marcação é uma atribuição de fichas a lugares e pode ser representada por um vetor cujo tamanho é o número de lugares na rede: o $i^{\text{ésimo}}$ componente do vetor representa o número de fichas contidas no lugar l_i. Denotamos por $m(l)$ o número de fichas em um lugar $l \in L$.

O *pré-conjunto* de um nó v em uma RdP é definido como $\bullet v = \{u \mid \langle u, v\rangle \in A\}$. De forma análoga, o *pós-conjunto* é definido como $v\bullet = \{u \mid \langle v, u\rangle \in A\}$.

Uma RdP pode ser representada por suas *matrizes de incidência* C^+ e C^-. Elas são matrizes $|L| \times |T|$ que combinam a informação proveniente das relações de fluxo e da função peso da rede. Os elementos $c^+_{l,t}$ da matriz C^+ são definidos como $c^+_{l,t} = P(\langle t, l\rangle)$, e os elementos $c^-_{l,t}$ da matriz C^- são definidos como $c^-_{l,t} = P(\langle l, t\rangle)$.

Um par formado por um lugar l e uma transição t é chamado de *auto laço* se p é tanto uma entrada quanto uma saída de t, ou seja, $(p \in \bullet t \ \wedge \ p \in t\bullet)$. Uma RdP é dita *pura* se ela não tem auto-laços. Uma RdP pura é completamente caracterizada por uma só matriz de incidência C, tal que $C = C^+ - C^-$.

A evolução dinâmica da marcação de uma RdP é governada pela ocorrência (disparos) de transições que consomem e produzem fichas. Regras de *habilitação* e *disparo* estão associadas às transições.

Definição 2.1. *Habilitação*

Um transição t está habilitada na marcação m se e somente se:

$$\forall l \in \bullet t, \qquad m(l) \geq P(\langle l, t \rangle)$$

ou, em notação matricial,

$$m \geq C^{-}(\cdot, t)^{T} \ .$$

O conjunto $H(m)$ indica o conjunto das transições habilitadas na marcação m, enquanto o *grau de habilitação* $h_i(m)$ é o número de habilitações simultâneas da transição t_i em m.

Definição 2.2. *Disparo*

O disparo de uma transição t, habilitada na marcação m, produz um marcação m' tal que

$$m' = m + O(t)I(t) \ ,$$

em que

$$I(t) = (P(\langle l_1, t \rangle), P(\langle l_2, t \rangle), \ldots, P(\langle l_L, t \rangle)) \quad e$$
$$O(t) = (P(\langle t, l_1 \rangle), P(\langle t, l_2 \rangle), \ldots, P(\langle t, l_L \rangle))$$

ou, em notação matricial,

$$m' = m - C^{-}(\cdot, t)^{T} + C^{+}(\cdot, t)^{T} \ .$$

Essa declaração é normalmente indicada de forma compacta como $m[t\rangle m'$ e podemos dizer que m' é diretamente alcançável a partir de m.

Exemplo 2.1. *Produtor/Consumidor*

A Figura 2.2 mostra um exemplo de rede de Petri extraído de [Bal07] que modela o problema clássico de um par de processos Produtor e Consumidor. O processo Produtor se caracteriza por duas fases: produção e entrega (representadas pelas transições t_a e t_b, respectivamente). O processo Consumidor também possui duas fases: recebimento e consumo (representadas pelas transições t_c e t_d, respectivamente). O lugar l_4 armazena a quantidade de espaços livres disponíveis e o lugar l_3 armazena a quantidade de itens produzidos e ainda não consumidos. O Produtor só produz um item por vez, ao passo que o consumidor só consome um item por vez. Um produtor só pode enviar um item produzido a um consumidor quando há espaço para armazená-lo, ou seja, quando há ao menos uma ficha em l_4. Um consumidor só pode consumir um item após a sua remoção do repositório (transição t_c). Essa remoção só pode ocorrer quando há itens produzidos e ainda não consumidos, ou seja, quando há ao menos uma ficha em l_3. Sendo assim, uma ficha que sai de l_4 com o disparo de t_b entra em l_3. Do mesmo modo, uma ficha que sai de l_3 com o disparo de t_c entra em l_4.

Definição 2.3. *Marcações Alcançáveis*

O conjunto de marcações alcançáveis $MA(m_0)$ de uma RdP com marcação inicial m_0 é definido como o menor conjunto de marcações que satisfaz as seguintes propriedades:

1. $m_0 \in MA(m_0)$,
2. $m_1 \in MA(m_0) \wedge \forall t \in T : m_1[t\rangle m_2 \Rightarrow m_2 \in MA(m_0)$.

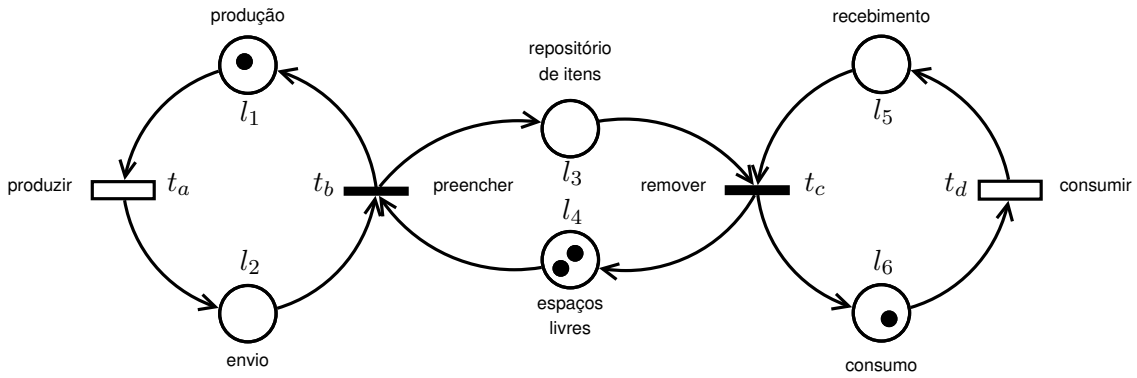


Figura 2.2: *Modelo em Rede de Petri do problema clássico do Produtor/Consumidor (extraído de [Bal07]).*

Também denotamos por $MA(m)$ o conjunto das marcações alcançáveis a partir de uma marcação qualquer m.

Sobre a base clássica das RdPs, outras redes de alto nível foram definidas com o objetivo de facilitar a modelagem de sistemas grandes ou complexos, ou até mesmo aumentar a expressividade do formalismo, como no caso das Redes de Petri Temporizadas (RdPTs).

O tempo foi introduzido nas RdPs para modelar a interação entre diferentes tarefas considerando seus tempos de início e término [Bal01]. Existem várias propostas diferentes para a incorporação do tempo nas RdPs, cada uma associa o tempo com um componente diferente do modelo. Essas propostas podem ser resumidas pelos seguintes itens:

- *lugares temporizados* – as fichas em um lugar temporizado se tornam disponíveis para a habilitação de uma transição apenas depois da decorrência de um intervalo de tempo (que é um atributo do lugar em questão);
- *fichas temporizadas* – uma ficha temporizada possui um *timestamp* indicando quando ela estará disponível para habilitar o disparo de um transição. Esse *timestamp* pode ser incrementado a cada vez que uma transição dispara;
- *arcos temporizados* – um atraso de "viagem" é associado a cada arco. As fichas ficam disponíveis para disparo somente quando elas chegam na transição;
- *transições temporizadas* – o início de uma tarefa corresponde à habilitação de uma transição e o término da tarefa corresponde ao disparo da transição. Diferentes políticas de disparo podem ser empregadas:
  - *disparo em três fases* – supõe que as fichas são consumidas do lugar de entrada quando a transição é habilitada e que as fichas são produzidas no lugar de saída após a decorrência do intervalo de tempo associado à transição;
  - *disparo atômico* – supõe que as fichas são consumidas do lugar de entrada e geradas no lugar de saída apenas após o disparo da transição.

Em RdPTs, as transições temporizadas são a abordagem mais utilizada, pois elas modelam de forma mais natural as mudanças de estado em um sistema real. As transições temporizadas podem ser vistas como tarefas de um sistema que consomem tempo para serem realizadas.

As Redes de Petri Estocásticas (RdPEs) são uma subclasse das RdPTs usadas para a descrição de sistemas dinâmicos de eventos discretos, cujo comportamento dinâmico pode ser representado por meio de CMTCs [FFN91].

Redes de Petri Estocásticas são RdPTs com transições temporizadas e com disparo atômico, nas quais o tempo de atraso de uma transição é uma variável aleatória exponencialmente distribuída: toda transição t_i está associada a um tempo de atraso aleatório cuja função de densidade de probabilidade é uma exponencial negativa com taxa r_i.

Nas RdPEs, quando múltiplas transições estão habilitadas em uma mesma marcação m, a transição que disparará primeiro será a transição que possuir o menor tempo de atraso associado a ela [Bal01].

Redes de Petri Estocásticas Generalizadas

As *Redes de Petri Estocásticas Generalizadas*, do inglês *Generalized Stochastic Petri Nets* (GSPN), são RdPEs estendidas com um outro tipo de transição possível além das transições temporizadas: as *transições imediatas*, ou seja, transições cujo tempo de atraso é zero [BCD+95, MCB84].

Se por um lado um transição temporizada é uma representação natural para uma tarefa que consome tempo, por outro lado uma transição imediata pode representar a verificação de uma condição lógica [Bal07].

Transições imediatas são disparadas com prioridade sobre transições temporizadas. As marcações alcançáveis em uma GSPN podem ser particionadas em duas: *marcações tangíveis* – aquelas em que somente transições temporizadas estão habilitadas – e *marcações sumidiças* (*vanishing*) – aquelas em que ao menos uma transição imediata está habilitada. O tempo gasto por uma GSPN em uma marcação sumidiça é deterministicamente zero, enquanto o tempo gasto em uma marcação tangível é positivo com probabilidade 1.

Na dinâmica temporal de uma GSPN no que se refere às marcações tangíveis, podemos supor que, quando uma transição temporizada é habilitada, um cronômetro interno é iniciado com um valor aleatoriamente selecionado segundo a distribuição exponencial associada à transição. Esse valor é decrementado em uma velocidade constante e a transição dispara quando o cronômetro atinge o valor zero.

No caso das marcações sumidiças, a dinâmica não consome tempo algum. Se somente uma transição imediata está habilitada na marcação, então ela será disparada e a marcação seguinte será produzida. Quando mais de uma uma transição imediata está habilitada, é necessário utilizar alguma métrica para definir a transição que será disparada. Se as transições que estão habilitadas são concorrentes, então elas podem ser disparadas em qualquer ordem. Mas se elas estão em conflito, então algum mecanismo deve ser empregado para selecionar uma transição. Por essa razão, as GSPN permitem a associação de prioridades às transições imediatas pertencentes a um mesmo conjunto de conflito.

Os modelos em GSPN utilizados neste trabalho podem ser formalmente definidos pela tupla $GSPN = \{L, T, A, P, W, m_0\}$, em que $\{L, T, A, P, m_0\}$ é uma RdP clássica e W é uma função que define o componente estocástico do modelo [2].

[2] A definição tradicional de uma GSPN inclui outros dois componentes adicionais: arcos inibidores e uma função que associa um nível de prioridade à transições imediatas. Porém, nos modelos em GSPN deste trabalho, não usamos esses componentes. Mais detalhes sobre eles podem ser encontrados em [BCD+95].

$$
\begin{array}{lll}
\text{Conjunto de Lugares} & L & = \{l_1, l_2, l_3, l_4, l_5, l_6\} \\
\text{Conjunto de Transições} & T & = \{t_a, t_b, t_c, t_d\} \\
\text{Peso das Transições} & W & = \{\alpha, 1, 1, \beta\} \\
\text{Marcação Inicial} & m_0 & = \{1, 0, 0, 2, 0, 1\}
\end{array}
$$

$$
\text{Matriz de Incidência} \quad C = \begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{array}
\begin{array}{c}
\begin{array}{cccc} t_a & t_b & t_c & t_d \end{array} \\
\begin{pmatrix}
-1 & +1 & & \\
+1 & -1 & & \\
 & +1 & -1 & \\
 & -1 & +1 & \\
 & & -1 & +1 \\
 & & +1 & -1
\end{pmatrix}
\end{array}
$$

Figura 2.3: *Especificação formal da GSPN da Figura 2.2.*

Mais precisamente, a função W mapeia transições em funções reais positivas da marcação do modelo. O valor $W(t_i, m)$ é chamado de *taxa* da transição t_i na marcação m se t_i é temporizada, ou *peso* da transição t_i na marcação m se t_i é imediata. A taxa ou peso das transições também pode ser independente da marcação; nesse caso, podemos denotar $W(t_i)$ somente por w_i.

Em marcações sumidiças, os pesos das transições imediatas podem ser usados para determinar qual transição imediata será disparada em uma situação de conflito. Grosso modo, uma situação de conflito ocorre quando um conjunto de transições imediatas está habilitado em uma dada marcação e o disparo de uma dessas transições desabilita o disparo de alguma outra transição do conjunto na marcação subsequente.

O exemplo dado na Figura 2.2 é uma GSPN, em que as transições t_b e t_c (que aparecem como retângulos preenchidos na cor preta) são transições imediatas. A especificação formal dessa rede é mostrada na Figura 2.3. O conjunto de marcações alcançáveis da rede é mostrado na Figura 2.4. As marcações delimitadas por elipses tracejadas são marcações sumidiças.

As RdPEs são isomórficas a CMTCs. A CMTC subjacente a um modelo em GSPN pode ser obtida por meio de duas regras simples:

1. o espaço de estados S da CMTC associada corresponde ao conjunto de marcações alcançáveis $MA(m_0)$ do modelo em GSPN. Cada marcação $m_i \in MA(m_0)$ corresponde a um estado $s_i \in S$ ($m_i \leftrightarrow s_i$);

2. a taxa de transição de um estado s_i (correspondente à marcação m_i) para um estado s_j (marcação s_j) é obtida pela soma das taxas de disparo das transições que estão habilitadas em m_i e cujo disparo produz a marcação m_j.

Considere que w_k é a taxa de disparo de t_k e $H_j(m_i) = \{t_h \mid t_h \in H(m_i) \wedge m_i[t_h\rangle m_j\}$, ou seja, $H_j(m_i)$ é o conjunto de transições cujo disparo leva a rede da marcação m_i para a marcação m_j. Com isso, o gerador infinitesimal (denotado pela matriz Q) da CMTC associada a um modelo em GSPN é dado por

$$
Q_{ij} = \begin{cases} \sum_{t_k \in H_j(m_i)} w_k & \text{para } i \neq j \\ -Q_i & \text{para } i = j \end{cases} \tag{2.1}
$$

em que

$$
Q_i = \sum_{t_k \in H(m_i)} w_k \tag{2.2}
$$

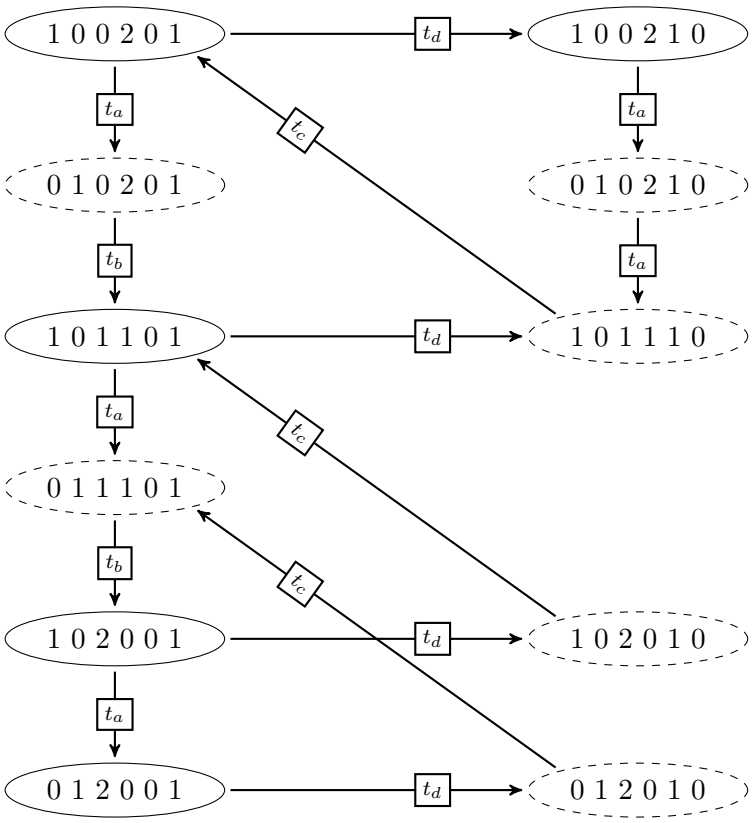


Figura 2.4: *Grafo de marcações alcançáveis da rede da Figura 2.2.*

É importante ressaltar que nem toda GSPN gera uma CMTC ergódica. Para que a CMTC associada a uma GSPN seja ergódica, é preciso que a rede seja *limitada* (ou seja, o espaço de estados é finito) e *reversível*.

Um lugar l em uma rede de Petri é *k-limitado* se e somente se, para toda marcação alcançável da rede, o número de fichas em l é inferior ou igual a k. Uma rede de Petri, por sua vez, é *limitada* se e somente se existe um número k finito tal que todos os lugares da rede sejam *k-limitados*.

Uma rede de Petri é *reversível* se e somente se a partir de qualquer marcação alcançável da marcação inicial m_0 da rede podemos retornar à m_0 novamente. Ou seja, em uma RdP reversível vale que $\forall m \in MA(m_0), m_0 \in MA(m)$.

Se a CMTC associada a uma GSPN é ergódica, então a distribuição estacionária π de probabilidades dos seus estados (marcações alcançáveis) existe. A distribuição π é a base para a obtenção de índices de desempenho sobre o modelo em GSPN. Esses índices podem ser calculados por meio de *funções de recompensa* definidas sobre marcações da rede. O valor médio de uma recompensa é computado com o auxílio de π.

Seja $r(m)$ uma função de recompensa; a recompensa média $\mathbb{E}[r]$ pode ser computada por meio da seguinte soma ponderada:

$$\mathbb{E}[r] = \sum_{m_i \in MA(m_0)} m_i \pi_i \tag{2.3}$$

Diferentes interpretações podem ser atribuídas à função de recompensa para o cálculo de diversos índices de desempenho. Os mais comuns são:

- *probabilidade de uma determinada condição no modelo*

Supondo que uma condição $\Upsilon(m)$ é verdadeira somente em algumas marcações da rede, podemos definir a seguinte função de recompensa:

$$r(m) = \begin{cases} 1 & \text{se } \Upsilon(m) = \text{verdadeiro} \\ 0 & \text{no caso contrário} \end{cases} \tag{2.4}$$

A probabilidade $P(\Upsilon)$ é dada pela Equação 2.3 sobre a função de recompensa definida na Equação 2.4.

- *número médio de fichas em um lugar l_i do modelo*

  A função de recompensa $r(m)$, nesse caso, simplesmente devolve o número de fichas do lugar l_i na marcação m:

  $$r(m) = n \qquad \text{se e somente se } m(l_i) = n \tag{2.5}$$

  O número médio de fichas em l_i, denotado por $\mathbb{E}[m(l_i)]$, é dado pela aplicação da Equação 2.3 sobre a função de recompensa definida na Equação 2.5.

- *número médio de disparos por unidade de tempo (ou rendimento) de uma transição t_j do modelo*

  Sabendo que uma transição dispara somente quando ela está habilitada, podemos definir uma função de recompensa que assume o valor w_j em toda marcação que habilita t_j:

  $$r(m) = \begin{cases} w_j & \text{se } t_j \in H(m) \\ 0 & \text{no caso contrário} \end{cases} \tag{2.6}$$

  O rendimento de t_j é dado pela aplicação da Equação 2.3 sobre a função de recompensa definida na Equação 2.6.

Existem várias ferramentas de software para a solução de modelos em RdP. Neste trabalho, nós empregamos uma ferramenta chamada *Stochastic Model checking Analyzer for Reliability and Timing* (SMART) [CJIMS06, SMA]. A SMART permite a análise de RdPEs por meio de variadas técnicas de solução, como, por exemplo, verificação simbólica, análise numérica ou simulação.

2.3.2 Álgebras de Processos Estocásticas

Nas *álgebras de processos*, como o próprio nome sugere, os processos são representados algebricamente, na forma de termos. Uma álgebra de processos é composta por um conjunto de símbolos de ações, um conjunto de operadores e um conjunto de axiomas descrevendo as propriedades dos operadores. O conjunto de axiomas (ou leis equacionais) pode também especificar quando dois processos são considerados equivalentes. Esse arcabouço algébrico possibilita o raciocínio formal sobre propriedades estruturais e comportamentais de um modelo de sistema concorrente.

As bases das álgebras de processos foram desenvolvidas, independentemente, por Milner e Hoare. Elas são parcialmente enraizadas nas Redes de Petri, na Teoria de Autômatos e nas linguagens formais. Milner desenvolveu a álgebra de processos *Calculus of Communicating Systems* (CCS) [Mil89], enquanto Hoare definiu a *Calculus of Sequential Processes* (CSP) [Hoa78]. Outras álgebras de processos bem difundidas são a *Language of Temporal Ordering Specification* (LOTOS) [BB87], a *Algebra of Communicating Processes* (ACP) [BK84] e a π-Calculus [Mil99].

As álgebras de processos clássicas se concentram nos aspectos funcionais do sistema: comportamento observável, controle de fluxo e sincronização. O tempo associado às ações é implícito e os modelos são não-determinísticos. O não-determinismo é originado pela necessidade de se modelar sistemas nos quais há incerteza sobre o comportamento de um processo, ou seja, sistemas em que escolhas devem ser feitas. Entretanto, a não quantificação do tempo e da incerteza impossibilita que medidas de desempenho sejam extraídas a partir do modelo.

Uma variação das álgebras de processos empregada na modelagem para a análise de desempenho de sistemas são as Álgebras de Processos Estocásticas (APEs). Os primeiros trabalhos formalizando a ideia de se estender as álgebras de processos clássicas para associar às ações atrasos com distribuição exponencial apareceram no início dos anos 90, com a *TImed Processes and Performance* (TIPP) [HHM98] e a *Performance Evaluation Process Algebra* (PEPA) [Hil96, HR98, CGHT07], ambas baseadas na CSP, e a *Extended Markovian Process Algebra* (EMPA) [BG96]. Depois dessas primeiras extensões, outras foram criadas tendo como base comum um modelo semântico subjacente relacionado às CMTCs.

Os fundamentos das APEs são os mesmos que os das álgebras de processos clássicas. Mas, além de considerar o comportamento funcional, as APEs adicionam aos modelos informações quantitativas, como tempo e probabilidades, permitindo a avaliação de diferentes propriedades do sistema relativas ao comportamento funcional (e.g., impasses e vivacidade), ao comportamento temporizado (e.g., vazões das tarefas, tempos de espera, tempo de serviço do sistema, etc.), ou à combinação desses dois últimos (e.g., probabilidade da ocorrência de um dado evento, duração de determinadas sequências de eventos, etc.) [Her98].

As APEs trazem para a modelagem de desempenho algumas características inerentes às álgebras de processos: a *composicionalidade* (a habilidade de se modelar um sistema em termos da interação de seus subsistemas), a *formalidade* (que dá um significado preciso para todos os termos da linguagem) e a *abstração* (a habilidade de se construir modelos complexos a partir de subprocessos detalhados, mas desconsiderando comportamentos internos quando for apropriado) [HR98].

As APEs estendem as álgebras de processos clássicas associando a cada ação do modelo uma variável aleatória, representando a sua duração. O objetivo dessa extensão é criar uma linguagem apropriada à especificação de processos estocásticos, mas mantendo o máximo possível das características de uma álgebra de processos, de modo que o modelo de desempenho possa ser definido como uma anotação no modelo básico do processo. Assume-se que as variáveis aleatórias associadas às ações são exponencialmente distribuídas, o que conduz à propriedade de que todo modelo de processo descrito em uma APE possui uma CMTC subjacente. Medidas de desempenho podem ser extraídas a partir dessa cadeia.

Álgebra de Processos para a Avaliação de Desempenho

Neste trabalho, empregamos a *Álgebra de Processos para Avaliação de Desempenho*, do inglês *Performance Evaluation Process Algebra* (PEPA). Uma descrição geral de modelos definidos em PEPA pode ser sumarizada pelos seguintes itens:

- um sistema é descrito por meio de interações entre *componentes*;
- componentes são constituídos por *ações*, que são as unidades atômicas do sistema. Toda ação é uma instância de um *tipo de ação*;

- A é o conjunto de *tipos de ações* presentes no sistema;
- Act é o multiconjunto de todas as ações do sistema;
- uma ação $a \in \mathsf{Act}$ é descrita por um par (α, r), em que $\alpha \in \mathsf{A}$ e r é a *taxa* (o parâmetro da distribuição exponencial negativa que governa a duração da ação), que pode ser um número pertencente a $\mathbb{R}^+$ ou o símbolo especial $\top$, que pode ser entendido como *taxa não especificada*;
- um conjunto de operadores é utilizado para construir definições de comportamentos complexos a partir de comportamentos mais simples. Esses operadores estão presentes na maioria das álgebras de processos clássicas; eles são: *composição sequencial*, *escolha*, *paralelismo* (ou *entrelaçamento*), *sincronização* e *encapsulamento*.

A sintaxe de PEPA (extraída de [Hil96]) é definida por:

$$P ::= (\alpha, r) \mid P.P \mid P \underset{L}{\bowtie} P \mid P + P \mid P/L \mid A$$

sendo L um conjunto de tipos de ações e A uma constante.

O nome dessas construções e a semântica associada a elas serão definidas a seguir. A título de convenção, componentes serão denotados por nomes iniciados por letras maiúsculas, enquanto nomes de tipos de ações serão iniciados por letras minúsculas.

Constante: $A \stackrel{def}{=} P$

Termo cujo significado é dado por uma equação de definição como $A \stackrel{def}{=} P$, que atribui à constante A o comportamento do componente P.

Ação: (α, r)

É o termo que habilita a execução de uma instância da ação de tipo α, com taxa igual a r.

Composição sequencial: $P_1 \,.\, P_2$

Termo que representa o componente que realiza o comportamento de P_1 e, na sequência, assume o comportamento de P_2.

O Exemplo 2.2 ilustra a aplicação da composição sequencial.

Exemplo 2.2. *Módulo de Memória em um Sistema Multiprocessado*

$$\mathtt{Mem} \stackrel{def}{=} (\mathtt{obter}, \top).(\mathtt{usar}, \mu).(\mathtt{liberar}, \top).\ \mathtt{Mem}$$

O componente `Mem` *representa um módulo de memória em um sistema multiprocessado que permite somente uma transferência de dados por vez. Cada processador requisitante de acesso à memória precisará "adquirir" a memória e, depois de ter completado a transferência dos dados, liberar a memória, disponibilizando-a para uma nova aquisição. As taxas das ações* `obter` *e* `liberar` *não foram definidas porque estão fora do controle da memória e, portanto, assumem um papel passivo nesse componente.*

Escolha: $P_1 + P_2$

Termo que representa o componente que possui dois comportamentos possíveis: P_1 ou P_2. O Exemplo 2.3 ilustra o uso do termo de escolha.

Exemplo 2.3. *Processador em um Sistema Multiprocessado*

$$\begin{aligned} \texttt{Proc} \overset{def}{=} \; & (\texttt{processar}, p_1\lambda).(\texttt{local}, m).\texttt{Proc} + \\ & (\texttt{processar}, p_2\lambda).(\texttt{obter}, o).(\texttt{usar}, \mu).(\texttt{liberar}, r).\texttt{Proc} \end{aligned}$$

`Proc` representa um processador no sistema multiprocessado. Após executar a ação `processar`, o componente pode assumir dois comportamentos distintos: o processador precisará acessar a memória local (com probabilidade p_1) ou precisará acessar o módulo de memória compartilhada (com probabilidade $p_2 = 1 - p_1$).

Sincronização: $P_1 \underset{L}{\bowtie} P_2$

Termo que representa uma *cooperação* entre dois componentes.

O Exemplo 2.4 ilustra o uso do termo de sincronização.

Exemplo 2.4. *Sincronização entre a Memória e o Processador em um Sistema Multiprocessado*

Os componentes `Mem` e `Proc`, definidos anteriormente, precisam trabalhar conjuntamente quando o processador precisa de dados não disponíveis localmente. Entretanto, as tarefas locais do processador podem ser realizadas de forma independente ao módulo de memória compartilhada. Sendo assim, a sincronização entre esses dois componentes ocorre somente sobre um subconjunto de suas ações.

$$\texttt{Sistema}_1 = (\texttt{Proc} \| \texttt{Proc}) \underset{L}{\bowtie} \texttt{Mem} \qquad\qquad L = \{\texttt{obter}, \texttt{usar}, \texttt{liberar}\}$$

O conjunto L define as ações sobre as quais a cooperação $\underset{L}{\bowtie}$ atuará. Em outras palavras, L define os pontos de sincronismo entre os componentes. As ações em L requerem o envolvimento dos dois componentes; a ação resultante terá o mesmo tipo das duas ações participantes e a sua taxa refletirá a taxa do componente participante mais lento (o que implica que a taxa $\top$ de uma ação passiva se tornará a taxa da ação que coopera com ela).

O operador $\|$ equivale a uma cooperação da forma $\underset{\emptyset}{\bowtie}$, ou seja, uma cooperação sem pontos de sincronismo, em que os processos podem ser executados de forma completamente independente. Sendo assim, `Proc`$\|$`Proc` representa dois processadores iguais que atuam de forma independente e que disputam o acesso à memória compartilhada.

Encapsulamento: P/L

O encapsulamento de um conjunto de ações L em um componente P torna-as privadas ao processo em questão, tornando-as não visíveis aos componentes que atuam em cooperação. Dessa forma, o encapsulamento pode impedir que sincronizações ocorram com essas ações. O único resultado observável de uma ação encapsulada é o atraso associado a ela e uma ação de tipo desconhecido, denotada por τ.

No Exemplo 2.5, será definida uma nova versão do componente `Proc` escondendo o acesso do processo à sua memória local e a uma nova versão do componente do sistema.

Exemplo 2.5. *Acesso Encapsulado a Memória Local em um Sistema Multiprocessado*

$$\begin{aligned} \texttt{Proc}' &\stackrel{def}{=} \texttt{Proc}/\{\texttt{local}\} \\ \texttt{Sistema}_2 &\stackrel{def}{=} (\texttt{Proc}' \| \texttt{Proc}') \underset{L}{\bowtie} \texttt{Mem} \qquad\qquad L = \{\texttt{obter}, \texttt{usar}, \texttt{liberar}\} \end{aligned}$$

O uso do encapsulamento tem duas consequências: (i) garante que componentes adicionados ao modelo no futuro não interajam de modo indesejado com as ações atualmente existentes no modelo; (ii) auxilia uma possível simplificação do modelo na fase de análise, uma vez que ações privadas não interferem no cálculo das medidas de desempenho.

É importante ressaltar (como pôde ser visto nos exemplos dados) que as expressões em álgebra de processos podem ser recursivas, o que torna possível a especificação de comportamentos infinitos sem a necessidade de um operador de recursão explícito.

Sempre que mais de uma ação estiver habilitada no processo em um dado momento (como ocorre no caso da escolha), uma *condição de disputa* governará o comportamento dinâmico do modelo, o que quer dizer que, entre todas as ações elegíveis à execução, a que for mais "rápida" será escolhida.

A condição de disputa substitui as ramificações não-determinísticas (definidas pelos operadores das álgebras de processos clássicas) por ramificações probabilísticas. A probabilidade de uma ação habilitada ser executada é dada pela razão entre a taxa da ação e a soma das taxas de todas as ações habilitadas no momento. O Exemplo 2.6 ilustra a modelagem de uma ramificação probabilística.

Exemplo 2.6. *Ramificações Probabilísticas*

$$(\alpha, \frac{r}{3}).P_1 + (\alpha, \frac{2r}{3}).P_2$$

Essa expressão especifica um componente que executa a ação α com duração média de $1/r$ e que tem, na sequência, dois possíveis comportamentos: P_1 (com probabilidade $1/3$) ou P_2 (com probabilidade $2/3$). Para representar esse comportamento, a ação α foi dividida em duas ações na expressão, de modo que sua taxa pudesse ser ajustada para capturar as probabilidades das suas duas possíveis continuações.

A semântica operacional dos termos de PEPA é dada pelas regras mostradas na Tabela 2.3. Essas regras podem ser lidas da seguinte maneira: se a(s) transição(ões) acima da linha de inferência pode(m) ser inferida(s), então podemos inferir a transição abaixo da linha.

As regras consideram as seguinte suposições:

- uma ação leva algum tempo para ser completada. Consequentemente, cada transição representa um avanço no tempo;
- todas as ações são homogêneas no tempo, ou seja, a taxa e o tipo de uma ação são independentes do tempo em que ela ela ocorre;
- o conjunto de ações de um componente também independe do tempo.

Na regra do operador de sincronização no caso em que $\alpha \in L$ (onde há sincronização dos componentes), a taxa resultante R reflete a capacidade de cada componente para executar ações do

Tabela 2.3: *Semântica operacional dos termos de PEPA.*

Termo	Regra de Transição
Prefixo	$\dfrac{}{(\alpha, r).P_1 \xrightarrow{(\alpha,r)} P_1}$
Escolha	$\dfrac{P_1 \xrightarrow{(\alpha,r)} P_1'}{P_1 + P_2 \xrightarrow{(\alpha,r)} P_1'}$ $\quad$ $\dfrac{P_2 \xrightarrow{(\alpha,r)} P_2'}{P_1 + P_2 \xrightarrow{(\alpha,r)} P_2'}$
Sincronização ($\alpha \notin L$)	$\dfrac{P_1 \xrightarrow{(\alpha,r)} P_1'}{P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(\alpha,r)} P_1' \underset{L}{\bowtie} P_2}$ $\quad$ $\dfrac{P_2 \xrightarrow{(\alpha,r)} P_2'}{P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(\alpha,r)} P_1 \underset{L}{\bowtie} P_2'}$
Sincronização ($\alpha \in L$)	$\dfrac{P_1 \xrightarrow{(\alpha,r_1)} P_1' \quad P_2 \xrightarrow{(\alpha,r_2)} P_2'}{P_1 \underset{L}{\bowtie} P_2 \xrightarrow{(\alpha,R)} P_1' \underset{L}{\bowtie} P_2'}$ $\quad$ $R = \dfrac{r_1}{r_\alpha(P_1)} \dfrac{r_2}{r_\alpha(P_2)} r_m$ $\quad$ $r_m = min(r_\alpha(P_1), r_\alpha(P_2))$
Encapsulamento	$\dfrac{P_1 \xrightarrow{(\alpha,r)} P_1'}{P_1/L \xrightarrow{(\alpha,r)} P_1'/L} (\alpha \notin L)$ $\quad$ $\dfrac{P_1 \xrightarrow{(\alpha,r)} P_1'}{P_1/L \xrightarrow{(\tau,r)} P_1'/L} (\alpha \in L)$
Constante	$\dfrac{P_1 \xrightarrow{(\alpha,r)} P_1'}{A \xrightarrow{(\alpha,r)} P_1'} (A \stackrel{def}{=} P_1)$

tipo α. Para um componente P_1 e o tipo de ação α, isso é dado pela *taxa aparente* de α em P_1 – $r_\alpha(P_1)$ – que é calculada pela soma das taxas das ações de tipo α habilitadas em P_1.

A semântica de cada termo em PEPA é dada por um sistema de transições rotuladas. Um estado corresponde a um *derivado* (ou seja, um termo sintático da linguagem) e um arco representa a ação que faz com que um derivado evolua para outro. O conjunto de derivados de um modelo é o conjunto de todos estados alcançáveis. Pela aplicação exaustiva das regras da Tabela 2.3 sobre esses estados, obtemos o Grafo de Derivação (GD) do modelo. Como exemplo, a Figura 2.5 mostra o GD do processo $\texttt{Proc} \underset{L}{\bowtie} \texttt{Mem}$, em que $L = \{\texttt{obter}, \texttt{usar}, \texttt{liberar}\}$ e `Mem` e `Proc` são os componentes definidos nos exemplos 2.2 e 2.3.

Os aspectos temporais do comportamento dos componentes são representados nos arcos do GD por meio do parâmetro da distribuição exponencial negativa que governa a duração da ação correspondente ao arco. O GD é a base da CMTC subjacente, que é utilizada para derivar as medidas de desempenho de um modelo em PEPA.

Para que a CMTC subjacente a um modelo em PEPA seja ergódica, o GD do modelo precisa ser fortemente conexo. A gramática de PEPA impõe as condições sintáticas necessárias para garantir a ergodicidade. O componente inicial de um modelo em PEPA é sempre cíclico. Em modelos cíclicos, a ergodicidade garante que os componentes de um modelo são estáticos, ou seja, que eles não são criados ou destruídos enquanto o modelo evolui. O termo inicial de um modelo em PEPA mostra todos os componentes paralelos que existirão durante a vida do modelo. Portanto, um modelo cíclico terá um número fixo desses termos ao longo de sua evolução.

Escolhas podem ocorrer dentro dos componentes e elas representam modos de comportamento alternativos, mas não estruturas alternativas. Sendo assim, em toda escolha, independentemente

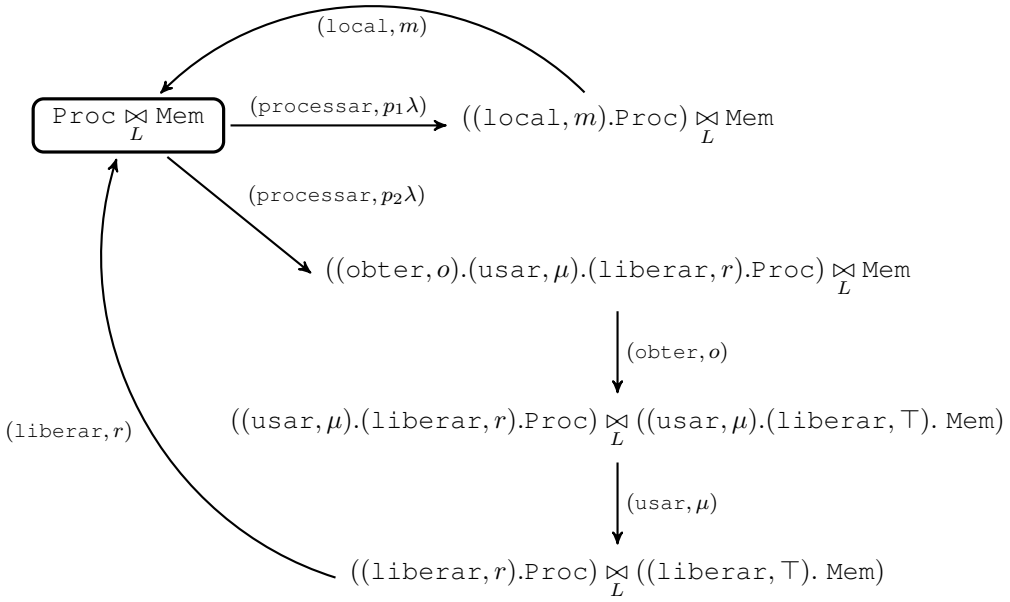


Figura 2.5: *Grafo de derivação do processo* $\texttt{Proc} \bowtie_L \texttt{Mem}$*, em que* $L = \{\texttt{obter}, \texttt{usar}, \texttt{liberar}\}$.

do componente escolhido, o modelo deve poder voltar para a ponto em que a mesma escolha pode ser feita novamente. Por essa razão, um componente que envolve o operador de escolha pode ser usado em uma cooperação, mas um componente envolvendo cooperação não pode ser usado em uma escolha. Essa condição sintática também é imposta na gramática de PEPA.

Assim como ocorre nas RdPEs, para derivar medidas de desempenho a partir de modelos em PEPA podemos associar *recompensas* a determinadas ações de interesse no sistema. A recompensa associada a um componente C_i (ou estado do sistema) pode ser calculada por meio da soma das recompensas associadas às ações que ele habilita. As medidas de desempenho são então derivadas a partir da recompensa total na distribuição de probabilidades do sistema no regime estacionário. Dessa forma, recompensas podem ser definidas no nível do modelo em PEPA, ao invés de serem definidas sobre a cadeia de Markov subjacente.

Seja r_i a recompensa associada ao componente C_i e π a probabilidade do regime estacionário. A recompensa total R é dada por:

$$R = \sum_i r_i \pi_i \tag{2.7}$$

Neste trabalho, para solucionar os modelos em PEPA, empregamos a ferramenta PEPA *Plug-in Project* [Smi10, PEPa]. A ferramenta permite a realização de análises de desempenho sobre modelos PEPA tanto por meio de métodos numéricos de solução como também por simulação. A PEPA *Plug-in* é um complemento para o ambiente integrado de desenvolvimento *Eclipse*. Por ser uma ferramenta gráfica, ela contém um editor para PEPA e também fornece alguns gráficos para a visualização dos índices de desempenho obtidos sobre os modelos.

2.3.3 Redes de Autômatos Estocásticos

As *Redes de Autômatos Estocásticos*, do inglês *Stochastic Automata Networks* (SAN), são uma técnica usada para modelar sistemas com espaços de estados grandes, introduzida por Plateau em 1985 [Pla85, PA91] e estendida em uma série de trabalhos posteriores [Fer98, Sal03, Bre04, Sal09, Bre09]. SAN tem sido aplicado com sucesso na modelagem de sistemas concorrentes que podem ser vistos como coleções de componentes que operam mais ou menos independentemente, requirindo somente interações infrequentes (como, por exemplo, sincronizações de ações) ou funcionando sob taxas de execução que podem variar em função do estado de um componente ou do estado do sistema como um todo.

Um sistema é descrito em SAN como um conjunto de N subsistemas modelados como um conjunto de autômatos estocásticos A_i, $1 \leq i \leq N$, cada um contendo n_i estados locais e transições entre eles. O estado global de um modelo em SAN é definido pela combinação do estado interno de cada autômato. Uma mudança no estado global é causada pela mudança do estado local de qualquer um dos autômatos do modelo.

Eventos

Uma mudança de estado em um modelo SAN é causada pela ocorrência de um *evento*. *Eventos locais* causam uma transição de estado em somente um autômato (*transição local*), enquanto *eventos sincronizantes* causam transições de estado simultâneas em mais de um autômato (*transições sincronizantes*). Uma transição é rotulada pela lista de eventos que podem dispará-la. Um evento é classificado como local quando o seu identificador aparece em somente um dos autômatos do modelo. De modo contrário, quando o seu identificador aparece em mais de um autômato do modelo, o evento é considerado sincronizante.

SAN pode ser aplicado tanto à escala de tempo contínua quanto à escala de tempo discreta. Na escala de tempo discreta, associa-se aos eventos *probabilidades de ocorrência*, enquanto na escala de tempo contínua, associa-se aos eventos *taxas de ocorrência*. Neste trabalho, sempre que usarmos modelos em SAN estaremos nos referindo a modelos em escala de tempo contínua.

Em modelos em SAN com escala de tempo contínua, todo evento deve estar associado a uma taxa de ocorrência e a probabilidades de roteamento. Uma taxa é um número real não-negativo, enquanto uma probabilidade é um valor real no intervalo $[0, 1]$.

A taxa de ocorrência é o parâmetro da distribuição exponencial que rege o comportamento temporizado do evento. Ela indica a frequência média de ocorrência do evento, ou em outras palavras, o inverso do intervalo de tempo médio entre duas ocorrências do evento. As probabilidades são usadas para atribuir pesos às transições rotuladas pelo evento, principalmente quando um evento está associado a mais de uma transição que parte de um mesmo estado do autômato.

A taxa e a probabilidade de um evento podem ser constantes ou *funcionais*, ou seja, podem depender do estado no qual o evento ocorre.

Taxas e Probabilidades Funcionais

Uma taxa ou probabilidade funcional é uma função do espaço de estados global. O conceito de taxa funcional em SAN (em que a taxa depende do estado global do modelo SAN) é mais abrangente que o conceito de *taxa de serviço dependente de estado* existente nas Redes de Filas (em que a taxa depende somente do estado da fila por si só). Por exemplo, podemos usar taxas funcionais para

modelar como o tempo de execução de uma tarefa do processo é afetado pela variação da carga do sistema, ou para modelar a dependência que pode existir entre a probabilidade de execução de uma tarefa e o estado corrente do processo.

Exemplo 2.7. *Modelo em SAN e sua Respectiva CMTC*

A Figura 2.6 mostra exemplo extraído do trabalho de Fernandes [Fer98] e exemplifica o uso de eventos sincronizantes e taxas funcionais em modelos SAN. Na Figura 2.6a temos dois autômatos estocásticos que se sincronizam por meio do evento e_4. Sendo assim, o autômato $\mathcal{A}_2$ só passa do estado w para o estado u quando o autômato $\mathcal{A}_1$ está no estado z e o evento e_4 ocorre. Nesse caso, ao mesmo tempo em que $\mathcal{A}_2$ passará para u, $\mathcal{A}_1$ passará para o estado x com probabilidade π_2, ou para o estado y com probabilidade π_1. É importante observarmos que a transição de z para x em $\mathcal{A}_1$ também pode ocorrer por meio do evento local e_3, sem a necessidade de sincronização com $\mathcal{A}_2$.

No modelo em SAN da Figura 2.6a também temos um evento com taxa funcional, o evento e_5. A Figura 2.6b define a função f associada ao evento.

Espaço de Estados Produto × Espaço de Estados Alcançáveis

A CMTC subjacente a um modelo em SAN é o produto (combinação) de todos os autômatos do modelo. Por essa razão, o tamanho do espaço de estados na CMTC subjacente é dado pelo produto do número de estados de cada autômato que compõe o modelo em SAN.

Por exemplo, a cadeia de Markov mostrada na Figura 2.6c (correspondente ao modelo em SAN da Figura 2.6a) possui $3 \times 2 = 6$ estados. Na CMTC do exemplo, todos os estados são alcançáveis. Apesar disso, é muito frequente em modelos em SAN termos estados inalcançáveis no espaço de estados resultante do produto dos autômatos. O Exemplo 2.8 mostra um exemplo de modelo com estados inalcançáveis.

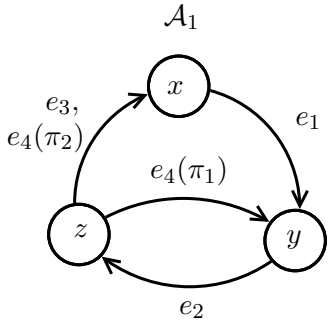
Exemplo 2.8. *Múltiplos Processos e um Recurso Compartilhado*

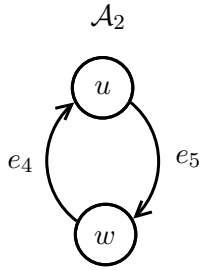
Considere o simples modelo em SAN da Figura 2.7, em que n processos compartilham o acesso a 2 unidades de um recurso. O autômato $\mathcal{A}_{recurso}$ representa as duas unidades do recursos e os autômatos $\mathcal{A}_{processo_1}, \ldots, \mathcal{A}_{processo_n}$ representam os n processos. Os eventos sincronizantes e_{a_i} e e_{l_i}, com $1 \leq i \leq n$, representam a alocação e a liberação (respectivamente) de 1 unidade do recurso pelo processo i. Dado que temos somente 2 unidades do recurso, em um dado estado do sistema não poderemos ter mais do que 2 autômatos em $\{\mathcal{A}_{processo_1}, \ldots, \mathcal{A}_{processo_n}\}$ cujo estado corrente seja usando. *Sendo assim, todos os estados do sistema em que o número de processos no estado* usando *for superior a 2 são* estados inalcançáveis *no sistema, já que a probabilidade do modelo se encontrar em alguns desses estados é nula.*

Funções de Integração

A obtenção de índices de desempenho e de confiabilidade sobre modelos SAN pode ser feita por meio da definição de *funções de integração*. Essas funções se baseiam na probabilidade do sistema estar em algum(uns) determinado(s) estado(s) de interesse. A avaliação dessas funções sobre a distribuição de probabilidades do modelo no regime estacionário nos dá os resultados desejados.

Duas funções importantes existentes em SAN nos auxiliam na definição de taxas funcionais, probabilidades funcionais e funções de integração: a função *st* (Definição 2.4) e a função *nb* (Definição 2.5).



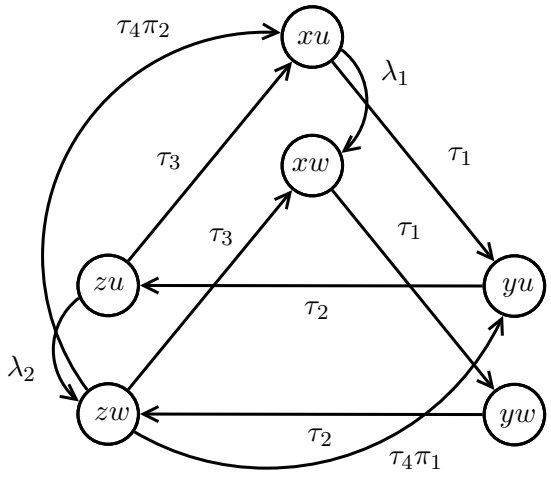


Evento	Taxa
e_1	τ_1
e_2	τ_2
e_3	τ_3
e_4	τ_4
e_5	f

(a) *Modelo em SAN com evento sincronizante e taxa funcional.*

$$f = \begin{cases} \lambda_1, & \text{se autômato } \mathcal{A}_1 \text{ está no estado } x \\ 0, & \text{se autômato } \mathcal{A}_1 \text{ está no estado } y \\ \lambda_2, & \text{se autômato } \mathcal{A}_1 \text{ está no estado } z \end{cases}$$

(b) *Definição da taxa funcional f.*



(c) *CMTC equivalente ao modelo em SAN.*

Figura 2.6: *Exemplo de modelo em SAN e sua respectiva cadeia de Markov.*

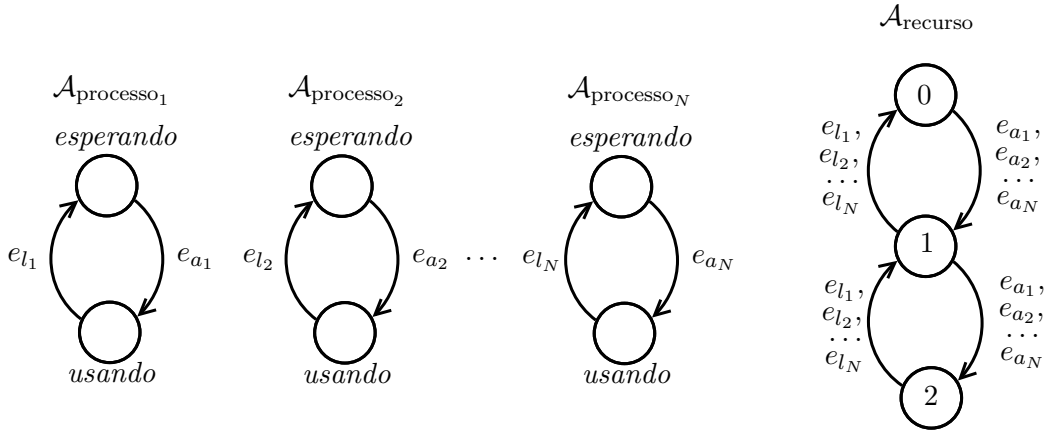


Figura 2.7: *Modelo em SAN de n processos compartilhando 2 unidades de um recurso.*

Definição 2.4. *Estado Corrente de um Autômato (st)*

A função $st(\mathcal{A})$ devolve o estado corrente no autômato $\mathcal{A}$.

Definição 2.5. *Número de Autômatos em um Estado (nb)*

A função $nb(\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}, \boldsymbol{e})$ devolve o número de autômatos em $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ cujo estado corrente é $\boldsymbol{e}$.

Exemplo 2.9. *Probabilidade de todas as Unidades do Recurso Estarem em Uso*

Considere o modelo em SAN do Exemplo 2.8 e suponha que estamos interessados na probabilidade das 2 unidades do recurso estarem em uso. A integração da função $f_{max\_ocupação}$ nos dá a probabilidade de interesse:

$$f_{max\_ocupação} = (st(\mathcal{A}_{recurso}) == 2)$$

No Capítulo 6, mostraremos como definir funções de integração para extrair os índices mais utilizados na avaliação de desempenho de sistemas.

Gerador Infinitesimal

A expressão algébrica do *gerador infinitesimal* (matriz de taxas de transição de estados) da cadeia de Markov associada a um modelo SAN bem definido é dada pelos geradores dos autômatos individuais e por operadores da Álgebra Tensorial Generalizada (ATG) [BFS05], uma extensão da Álgebra Tensorial Clássica (ATC). A fórmula tensorial para o gerador infinitesimal de um modelo SAN é chamada *Descritor Markoviano*.

Cada autômato A_i de um modelo SAN é descrito por $n_i \times n_i$ matrizes quadradas. No caso de modelos SAN com eventos sincronizantes, o descritor é expresso em duas partes: uma parte *local* (para agrupar os eventos locais) e uma parte *de sincronização* (para agrupar os eventos sincronizantes). A parte local é definida pela soma tensorial de $Q_l^{(i)}$, as matrizes dos geradores infinitesimais das transições locais de cada A_i. Na parte de sincronização, cada evento corresponde a dois produtos tensoriais: um para as matrizes de ocorrência $Q_{s^+}^{(i)}$ (que contêm as taxas positivas) e outro para as matrizes de ajuste $Q_{s^-}^{(i)}$ (que contêm as taxas negativas). O descritor é a soma da parte local e da parte de sincronização, expressa como:

$$Q = \bigoplus_{i=1}^{N}{}_g \, Q_l^{(i)} + \sum_{s \in \varepsilon} \left(\bigotimes_{i=1}^{N}{}_g \, Q_{s^+}^{(i)} + \bigotimes_{i=1}^{N}{}_g \, Q_{s^-}^{(i)} \right) \tag{2.8}$$

em que $\begin{cases} N & \text{é o número de autômatos do modelo em SAN} \\ \varepsilon & \text{é o conjunto de identificadores dos eventos sincronizantes} \end{cases}$.

O Apêndice B faz uma definição sucinta dos operadores da Álgebra Tensorial Clássica e da Álgebra Tensorial Generalizada

Para exemplificar a obtenção do descritor Markoviano de um modelo em SAN, considere o exemplo dado na Figura 2.6a. A parte local Q_l do descritor é expressa por:

$$Q_l = Q_l^{(1)} \oplus_g Q_l^{(2)} = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} -\tau_1 & \tau_1 & 0 \\ 0 & -\tau_2 & \tau_2 \\ \tau_3 & 0 & -\tau_3 \end{pmatrix} \end{array} \bigoplus_g \begin{array}{c} \\ u \\ w \end{array} \begin{array}{c} \begin{array}{cc} u & w \end{array} \\ \begin{pmatrix} -f & f \\ 0 & 0 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ xu \\ xw \\ yu \\ yw \\ zu \\ zw \end{array} \begin{array}{c} \begin{array}{cccccc} xu & xw & yu & yw & zu & zw \end{array} \\ \left(\begin{array}{cc|cc|cc} -(\tau_1+\lambda_1) & \lambda_1 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -\tau_2 & 0 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\lambda_2+\tau_3) & \lambda_2 \\ 0 & \tau_3 & 0 & 0 & 0 & -\tau_3 \end{array}\right) \end{array}$$

A parte de sincronização positiva Q_{s^+} do descritor é dada por:

$$Q_{s^+} = Q_{s^+}^{(1)} \otimes_g Q_{s^+}^{(2)} = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \tau_4\pi_2 & \tau_4\pi_1 & 0 \end{pmatrix} \end{array} \bigotimes_g \begin{array}{c} \\ u \\ w \end{array} \begin{array}{c} \begin{array}{cc} u & w \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ xu \\ xw \\ yu \\ yw \\ zu \\ zw \end{array} \begin{array}{c} \begin{array}{cccccc} xu & xw & yu & yw & zu & zw \end{array} \\ \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \tau_4\pi_2 & 0 & \tau_4\pi_1 & 0 & 0 & 0 \end{array}\right) \end{array}$$

Na definição de Q_{s^+}, nós consideramos o autômato $\mathcal{A}^{(1)}$ como o autômato *mestre* da sincronização, ou seja, é ele quem contém a taxa de disparo do evento sincronizante. Os demais autômatos que participam da sincronização em questão são considerados *escravos*; atribui-se sempre o valor 1 para a taxa do evento sincronizante em um autômato escravo.

A parte de sincronização negativa Q_{s^-} do descritor (considerando $\mathcal{A}^{(1)}$ como autômato mestre) é definida como:

$$
Q_{s^-} = Q^{(1)}_{s^-} \otimes_g Q^{(2)}_{s^-} = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\tau_4 \end{pmatrix} \end{array} \bigotimes_g \begin{array}{c} \\ u \\ w \end{array} \begin{array}{c} \begin{array}{cc} u & w \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{array}
$$

$$
= \begin{array}{c} \\ xu \\ xw \\ yu \\ yw \\ zu \\ zw \end{array} \begin{array}{c} \begin{array}{cccccc} xu & xw & yu & yw & zu & zw \end{array} \\ \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\tau_4 \end{array}\right) \end{array}
$$

Finalmente, o gerador infinitesimal Q do autômato global do modelo é:

$$
Q = Q_l + Q_{s^+} + Q_{s^-} = \begin{array}{c} \\ xu \\ xw \\ yu \\ yw \\ zu \\ zw \end{array} \begin{array}{c} \begin{array}{cccccc} xu & xw & yu & yw & zu & zw \end{array} \\ \left(\begin{array}{cc|cc|cc} -(\tau_1+\lambda_1) & \lambda_1 & \tau_1 & 0 & 0 & 0 \\ 0 & -\tau_1 & 0 & \tau_1 & 0 & 0 \\ \hline 0 & 0 & -\tau_2 & 0 & \tau_2 & 0 \\ 0 & 0 & 0 & -\tau_2 & 0 & \tau_2 \\ \hline \tau_3 & 0 & 0 & 0 & -(\lambda_2+\tau_3) & \lambda_2 \\ \tau_4\pi_2 & \tau_3 & \tau_4\pi_1 & 0 & 0 & -(\tau_3+\tau_4) \end{array}\right) \end{array}
$$

É possível observar que o gerador infinitesimal obtido por meio do descritor do modelo SAN corresponde exatamente à CMTC mostrada na Figura 2.6c.

Em SAN, o problema da explosão de estados associado aos modelos Markovianos é atenuado pelo fato de que o gerador infinitesimal do modelo é armazenado apenas em sua forma compacta, dado que ele é representado pelas matrizes menores de cada autômato do modelo. Toda informação relevante para a solução e análise do modelo pode ser recuperada dessas matrizes por meio da fórmula do descritor Markoviano, sem a necessidade de que o gerador infinitesimal seja explicitamente construído.

SAN foi o primeiro formalismo de modelagem a utilizar a álgebra tensorial como forma de representação. Depois dele, outros trabalhos propuseram a aplicação da álgebra tensorial na representação de formalismos clássicos, como nas RdPEs [Don94, CT96] e nas APEs [HK01].

Um modelo em SAN pode ser numericamente resolvido pela ferramenta *Performance Evaluation of Parallel Programs* (PEPS) [BFPS07, PEPb]. A PEPS implementa vários métodos iterativos para a solução de modelos em SAN e estratégias para melhorar a relação entre tempo de proces-

samento e consumo de memória no resolução dos modelos. A ferramenta possibilita a realização tanto de análises em regime estacionário quanto de análises transientes.

2.4 Conclusão do Capítulo

Neste capítulo introduzimos os conceitos e as técnicas de modelagem que empregamos neste trabalho de doutorado. Para a modelagem do controle de fluxo dos processos de negócios, usamos a BPMN. Para os modelos para a análise de desempenho, usamos os formalismos estocásticos Markovianos GSPN, PEPA e SAN. No Capítulo 3, comparamos o uso desses formalismos na modelagem de diferentes cenários de processos de negócio.

Capítulo 3

Modelagem Estocástica Aplicada a Processos de Negócio

Neste capítulo, nós avaliamos a viabilidade da aplicação de diferentes modelos de análise de desempenho no domínio da GPN. Para isso, definimos diferentes cenários que agrupam requisitos comumente encontrados na modelagem de processos de negócio. Exemplos dos cenários (descritos em BPMN) foram representados em três formalismos Markovianos utilizados na modelagem e análise de sistemas concorrentes: as Redes de Petri Estocásticas Generalizadas (GSPN), a Álgebra de Processos para Avaliação de Desempenho (PEPA) e as Redes de Autômatos Estocásticos (SAN).

Esses formalismos podem ser avaliados sob duas perspectivas diferentes: *modelagem* e *solução*. Neste trabalho, nós nos concentramos na perspectiva de modelagem. Os seguintes critérios guiaram nossa avaliação: expressividade, poder de abstração, extensibilidade e legibilidade do modelo.

O restante deste capítulo está organizado da seguinte forma: a Seção 3.1 apresenta os cenários de processos de negócio escolhidos para a avaliação dos formalismos e os seus respectivos modelos para análise de desempenho. Na Seção 3.2 definimos os critérios de avaliação e discutimos os resultados observados na modelagem dos cenário. Essa seção sintetiza os prós e contras do uso de cada um dos formalismos na modelagem de processos de negócio visando a análise de desempenho.

3.1 Cenários de Processos de Negócio

Para avaliar apropriadamente os formalismos estudados neste trabalho sob o domínio da GPN, nós vamos considerar diferentes cenários que ilustram características importantes e frequentemente encontradas nos modelos de processos de negócio.

O primeiro cenário (Seção 3.1.1) compreende quatro estruturas de controle de fluxo consideradas pela *Workflow Management Coalition* [WFM] essenciais na modelagem de processos de negócio: *sequências*, *divisão/junção OU*, *divisão/junção E* e *iterações*.

O segundo cenário (Seção 3.1.2) aborda a modelagem de estruturas mais complexas de ramificação e junção de fluxos, existentes em diversas linguagens e ferramentas da área de GPN.

O terceiro cenário (Seção 3.1.3) explora características dos formalismos para a modelagem de desempenho que não são comumente encontradas nas linguagens de modelagem de processos de negócio. Cada cenário é ilustrado por um exemplo simples de processo encontrado no mundo real, descrito em BPMN.

3.1.1 Cenário 1 – Estruturas Básicas

A Figura 3.1 apresenta um modelo em BPMN de um processamento de pedidos típico, comumente encontrado em aplicações de comércio eletrônico. Após a recepção de um novo pedido de

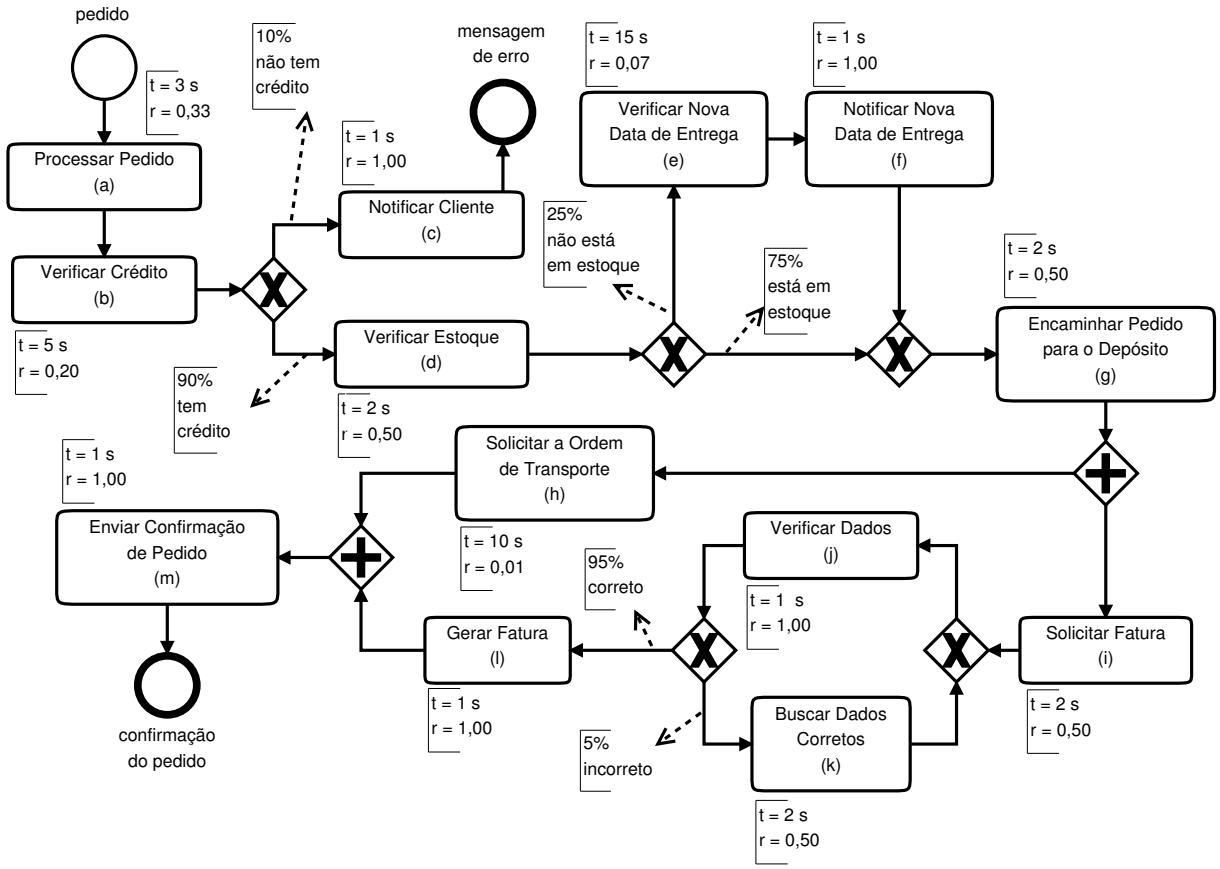


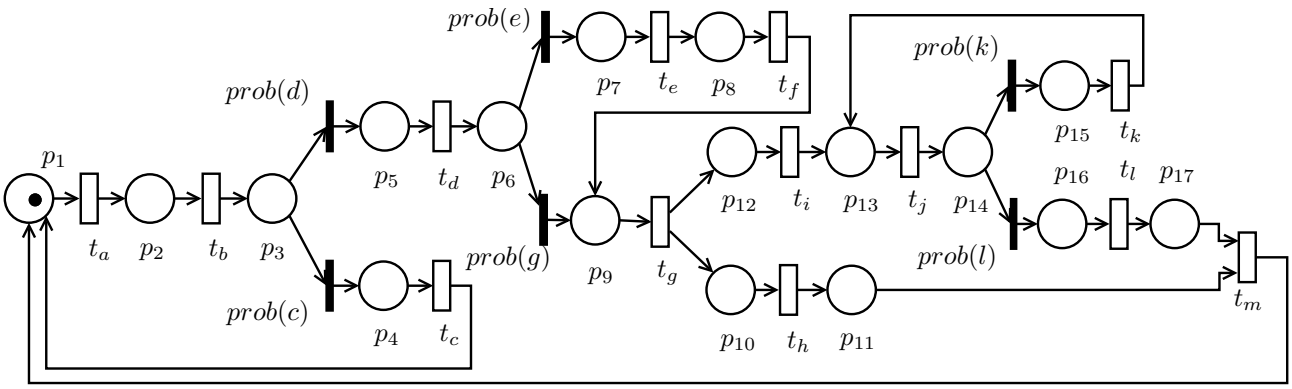
Figura 3.1: *Um processamento de pedidos típico modelado em BPMN.*

cliente, a aplicação verifica o crédito do cliente; se algum problema é encontrado, o processamento é interrompido. Caso contrário, a aplicação verifica a disponibilidade dos itens no estoque. Depois de encaminhar o pedido para o depósito, a expedição do pedido é solicitada ao mesmo tempo em que a fatura é gerada. A geração da fatura é composta por quatro passos: (i) solicitação , (ii) verificação dos dados, (iii) correção dos dados e (iv) emissão da fatura.

Após a conclusão da expedição do pedido e da geração da fatura, o processamento do pedido termina com o envio de uma confirmação para o cliente.

A Figura 3.1 possui, para cada tarefa x no modelo, uma anotação que indica o seu *tempo médio de execução* $t(x)$ e a sua *taxa de execução* $r(x)$ (equivalente a $1/t(x)$). Nós supomos que o tempo de execução de um tarefa x é uma variável exponencialmente distribuída com média $r(x)$. Na Figura 3.1, nós também temos anotações com as probabilidades associadas aos ramos dos desvios que envolvem escolhas.

A tarefa `Encaminhar Pedido para o Depósito` marca o início de uma *divisão E* (i.e., a divisão de um fluxo de sequência em mais fluxos de sequência a serem executados de forma paralela) no modelo de "processamento de pedidos". Nós denotamos o início de um paralelismo em um modelo em GSPN por uma transição com tantos lugares de saída quanto for o número de ramificações da ativação incondicional paralela. Na Figura 3.2, isso corresponde à transição t_g e os seus lugares de saída p_{10} e p_{12}). Uma *junção E* (i.e., um ponto de sincronização), como a que precede a tarefa `Enviar Confirmação de Pedido`, pode ser modelada como uma transição com tantos lugares de entrada quantos forem os fluxos de sequência a serem sincronizados. Na Figura 3.2, a *junção E* é modelada pelo transição t_m e os seus lugares de entrada p_{11} e p_{17}).



Transição Temporizada	t_a	t_b	t_c	t_d	t_e	t_f	t_g	t_h	t_i	t_j	t_k	t_l	t_m
Taxa	0,33	0,2	1	0,5	0,07	1	0,5	0,01	0,5	1	0,5	1	1

Transição Imediata	$prob(c)$	$prob(d)$	$prob(e)$	$prob(g)$	$prob(k)$	$prob(l)$
Probabilidade	0,1	0,9	0,25	0,75	0,05	0,95

Figura 3.2: *Modelo em GSPN do exemplo de "processamento de pedidos".*

A representação de uma ativação incondicional em paralelo em PEPA requer mais de um componente. Como podemos ver na Figura 3.3, `PPedido` e `PFatura` são compostos por meio do operador de cooperação, usando `g` e `m` como tipos de ações sincronizantes. O tipo de ação `g` marca o início do paralelismo, enquanto `m` delimita a sincronização dos fluxos de sequência paralelos [1].

Nos modelos em SAN, o paralelismo é representado por diferentes autômatos. Como pode ser visto na Figura 3.4, dois autômatos são necessários no exemplo (um para cada fluxo de sequência que parte da ativação incondicional em paralelo) e a sincronização é feita pelos eventos sincronizantes e_g e e_m.

Um desvio condicional exclusivo pode ser modelado em GSPN por transições que compartilham um mesmo lugar de entrada. Se existe uma probabilidade associada à cada fluxo de sequência que parte do desvio, nós podemos usar transições imediatas para representá-las logicamente. No exemplo, as tarefas `Notificar Cliente` e `Verificar Estoque` estão em um desvio condicional exclusivo. Na Figura 3.2, esse desvio é representado pelo lugar p_3 e as duas transições imediatas rotuladas pelas suas probabilidades: $prob(c)$ e $prob(d)$.

Em PEPA, nós usamos o operador "+" para modelar uma escolha. A taxa da tarefa que precede a decisão pode ser ajustada para capturar a probabilidade de cada fluxo de sequência possível a ser seguido a partir do desvio, como fizemos na tarefa `b` no componente `PPedido` na Figura 3.3.

No modelo em SAN, um desvio condicional exclusivo é representado por um estado com uma ou mais transições de saída (como o estado 1_1 no autômato $\mathcal{A}_1$ da Figura 3.4). Assim como acontece nos modelos em PEPA, a condição de disputa que governa o comportamento dinâmico de um modelo quando mais de uma tarefa está habilitada nos permite especificar as ramificações probabilísticas. Para representar as probabilidades associadas aos ramos, nós podemos associar a cada transição um evento cuja taxa é dada pela taxa do evento que precede o ponto de decisão, multiplicada pela probabilidade de transição (como nos eventos e_{b_1} e e_{b_2} do autômato $\mathcal{A}_1$).

[1] O operador $\bowtie_L$ é representado textualmente no PEPA *Plug-in Project* como `<L>`.

```
// Taxas de execução associadas a cada tarefa
r_a = 0.33;   r_b = 0.20;   r_c = 1.00;   r_d = 0.50;
r_e = 0.07;   r_f = 1.00;   r_g = 0.50;   r_h = 0.01;
r_i = 0.50;   r_j = 1.00;   r_k = 0.50;   r_l = 1.00;
r_m = 1.00;

// Probabilidades de roteamento associadas às escolhas
prob_c = 0.10;   prob_d = 1 - prob_c;
prob_e = 0.25;   prob_g = 1 - prob_e;
prob_k = 0.05;   prob_l = 1 - prob_k;

// Componentes do processamento de pedidos
PPedido = (a,r_a).((b,prob_c * r_b).(c,r_c).PPedido +
                   (b,prob_d * r_b).PEstoque);

PEstoque = (d,prob_g * r_d).PTermino +
           (d,prob_e * r_d).(e,r_e).(f,r_f).PTermino;

PTermino = (g,r_g).(h, r_h).(m,r_m).PPedido;

PFatura  = (g,⊤).(i,r_i).PVerificacao;

PVerificacao    = (j,prob_l * r_j).(l,r_l).(m,⊤).PFatura +
                  (j,prob_k * r_j).(k,r_k).PVerificacao;

PPedido <g,m> PFatura
```

Figura 3.3: *Modelo em PEPA do exemplo de "processamento de pedidos".*

3.1.2 Cenário 2 – Estruturas Avançadas de Ramificação e Junção

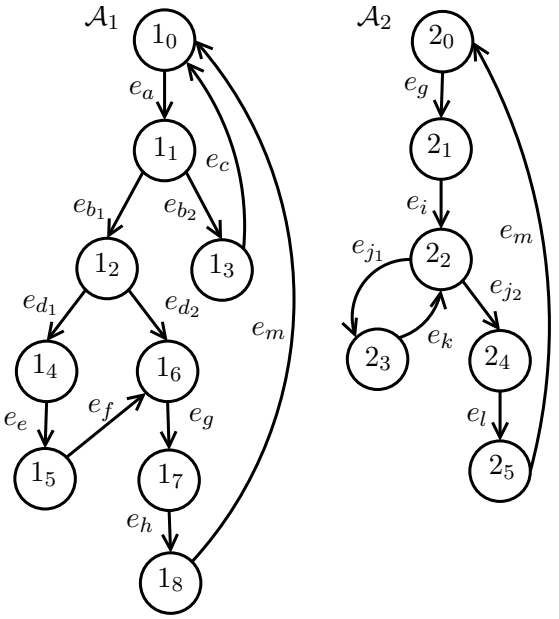
Este cenário considera modelos de processos de negócio com estruturas de ramificação e junção mais sofisticadas. Exemplos conhecidos desse tipo de estruturas são os descritos no trabalho de Aalst et al [vdAtHKB03]: *escolha múltipla*, *junção sincronizada*, *junção múltipla*, e *junção discriminatória*. Para ilustrar este cenário, usaremos um processo existente no sistema de saúde francês.

O sistema de saúde na França é composto por uma mistura de financiamento público com financiamento privado. O financiamento público oferece a cobertura dos serviços médicos básicos. Mas os franceses podem também contratar seguros-saúde que podem cobrir ou reduzir o valor das despesas com serviços suplementares. Sendo assim, para definir o custo de um serviço médico para um paciente, é necessário: (i) verificar se o paciente é coberto pelo seguro-saúde público e/ou particular; (ii) para cada seguro aplicável, avaliar a quantia coberta, de acordo com o serviço médico realizado; finalmente, (iii) combinar os valores cobertos para calcular os custos finais para o paciente. A Figura 3.5 mostra o modelo em BPMN desse processo.

A tarefa `b` na Figura 3.5 marca o início de uma *escolha múltipla*: após a sua execução, uma das seguintes situações pode ocorrer: (i) nenhum dos ramos será habilitado para execução; (ii) somente um dos ramos será habilitado pra execução; (iii) ambos os ramos serão habilitados para execução. Nesse caso, eles podem ser executados em paralelo.

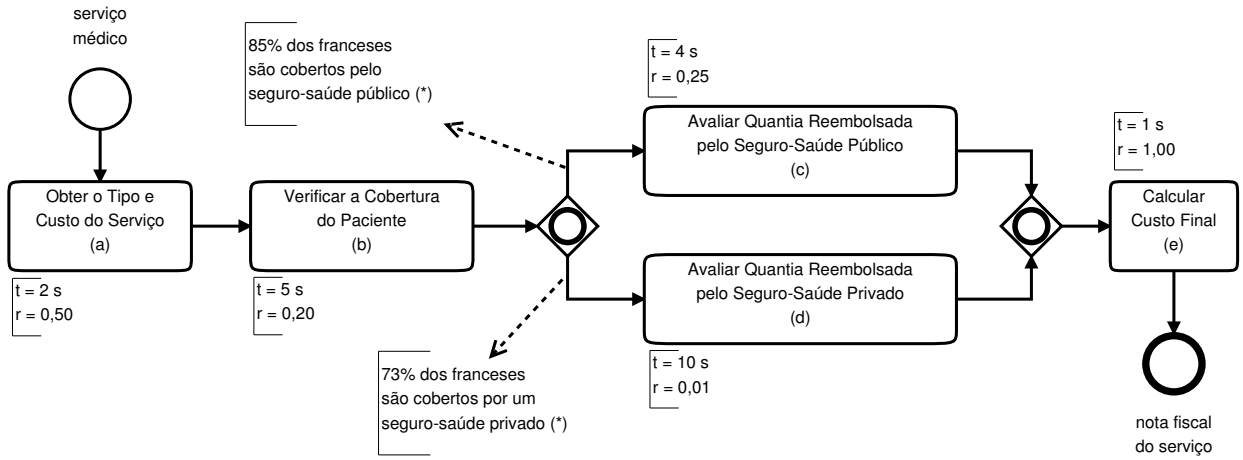
Como há a possibilidade dos ramos serem executados paralelamente, precisamos de uma estrutura especial para fazer a junção dos ramos de uma escolha múltipla. Os três tipos de junção descritos nos padrões de controle de fluxo [vdAtHKB03] são:

- a *junção sincronizada*, que sincroniza o fim da execução de todos os ramos habilitados na



Evento	Taxa
e_a	$0,33$
e_{b_1}	$prob(d) * 0,20$
e_{b_2}	$prob(c) * 0,20$
e_c	$1,00$
e_{d_1}	$prob(e) * 0,50$
e_{d_2}	$prob(g) * 0,50$
e_e	$0,07$
e_f	$1,00$
$\boldsymbol{e_g}$	$0,50$
e_h	$0,01$
e_i	$0,50$
e_{j_1}	$prob(k) * 1,00$
e_{j_2}	$prob(l) * 1,00$
e_k	$0,50$
e_l	$1,00$
$\boldsymbol{e_m}$	$1,00$

Figura 3.4: *Modelo em SAN do exemplo de "processamento de pedidos".*



(*) Essas porcentagens podem não representar a situação atual na França.

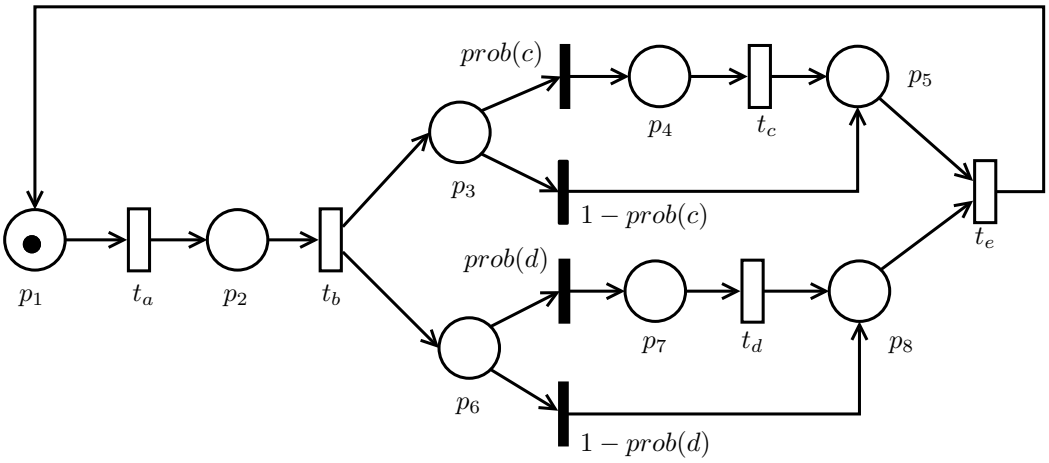
Figura 3.5: *Um processo (simplificado) para determinar o custo de um serviço médico, modelado em BPMN.*

escolha múltipla antes de habilitar as tarefas seguintes;

- a *junção múltipla*, que habilita a execução do fluxo de sequência seguinte toda vez em que a execução de um fluxo de sequência habilitado na escolha múltipla termina; e
- a *junção discriminatória*, que habilita o fluxo de sequência seguinte somente uma vez, no primeiro término de execução dos fluxos de sequência habilitados pela escolha múltipla.

Pela descrição dada do processo do sistema de saúde francês, podemos observar que o padrão de junção associado à escolha múltipla da Figura 3.5 é a junção sincronizada.

No modelo em GSPN, para expressar a escolha múltipla, nós usamos duas transições imediatas para cada ramo da escolha: uma modela a probabilidade do ramo ser executado, enquanto a outra modela a probabilidade do ramo não ser executado. Como as transições imediatas na Figura 3.6 alimentam os lugares p_5 e p_8 mesmo se os ramos não são selecionados para serem executados, a



Transição Temporizada	t_a	t_b	t_c	t_d	t_e
Taxa	$0,5$	$0,2$	$0,25$	$0,01$	1

Transição Imediata	$prob(c)$	$prob(d)$
Probabilidade	$0,85$	$0,73$

Figura 3.6: *Modelo em GSPN do processo de "cálculo de custo de procedimento médico".*

junção é modelada pela transição t_e e os seus dois lugares de entrada p_5 e p_8. Dado que não possuem uma taxa de execução, transições imediatas nos ajudam a expressar ramificações e junções mais sofisticadas sem impactar os resultados da análise.

Para mapear o padrão de escolha múltipla em PEPA e em SAN, nós precisamos introduzir no modelo ações e eventos "artificiais". Esse artefato auxilia na modelagem, mas precisa ser usado de forma consciente, já que ele pode ter efeitos colaterais. O tempo gasto em uma ação ou evento artificial impacta as medidas de desempenho. Nós podemos minimizar esse impacto nos resultados da análise atribuindo altas taxas de execução (relativas às outras taxas do modelo) a essas ações e eventos artificiais. Além disso, em sistemas reais, é razoável supor que, associado a cada decisão de roteamento ou sincronização, há um custo que também deve ser considerado na modelagem.

Os tipos de ação artificiais criados para modelar o exemplo em PEPA foram `c1`, `c2`, `d1` e `d2` da Figura 3.7. Os processos `P1` e `P2` representam os ramos da escolha múltipla; eles cooperam com o processo principal `PCalc`, usando `<b,e>` como o conjunto de tipos de ação sincronizantes.

De modo análogo, os evento locais e_{c_1}, e_{c_2}, e_{d_1} and e_{d_2} foram introduzidos no modelo em SAN da Figura 3.8 para expressar as probabilidades de execução de cada um dos ramos da escolha múltipla; os eventos sincronizantes e_b e e_e são responsáveis por delimitar, respectivamente, o início da escolha e a junção sincronizante.

É importante esclarecermos que ações imediatas não existem em PEPA, mas existem em outras álgebras de processos estocásticas, como, por exemplo, na *Extended Markovian Process Algebra* (EMPA).

3.1.3 Cenário 3 – Dependências Funcionais

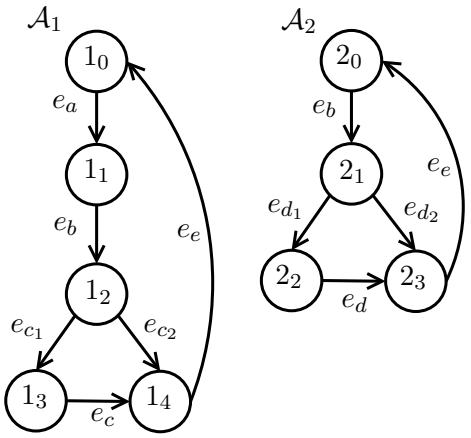
O cenário discutido nesta seção representa processos de negócio com tarefas cujas taxas de execução dependem do estado do sistema. Para ilustrá-lo, usaremos o modelo da Figura 3.9, que

```
// Taxas de execução associadas a cada tarefa
r_a = 0.50;   r_b = 0.20;   r_c = 0.25;   r_d = 0.01;
r_e = 1.00;   r_instantanea = 50.00;

// Probabilidades de roteamento associadas à escolha múltipla
prob_c = 0.85;   prob_d = 0.73;

// Componentes para o cálculo do custo do serviço médico
PCalc = (a,r_a).(b,r_b).(e,r_e).PCalc;
P1 = (b,⊤).((c1,prob_c * r_instantanea).(c,r_c).(e, ⊤).P1 +
             (c2,(1-prob_c) * r_instantanea).(e,⊤).P1);
P2 = (b,⊤).((d1,prob_d * r_instantanea).(d,r_d).(e,⊤).P2 +
             (d2,(1-prob_d) * r_instantanea).(e,⊤).P2);
PCalc <b,e> P1 <b,e> P2   // Sistema completo
```

Figura 3.7: *Modelo em PEPA do processo de "cálculo de custo de procedimento médico".*



Evento	Taxa
e_a	$0,50$
$\boldsymbol{e_b}$	$0,20$
e_{c_1}	$prob(c) * 50$
e_{c_2}	$(1 - prob(c)) * 50$
e_c	$0,25$
e_{d_1}	$prob(d) * 50$
e_{d_2}	$(1 - prob(d)) * 50$
e_d	$0,01$
$\boldsymbol{e_e}$	$1,00$

Figura 3.8: *Modelo em SAN do processo de "cálculo de custo do procedimento médico".*

define um processo simples de produção e empacotamento. O processo é composto de três entidades: a primeira representa um produtor de itens (que opera ininterruptamente enquanto o estoque não está completo), a segunda representa um empacotador (que precisa agrupar itens para criar pacotes e enviá-los do estoque para o serviço de transporte) e a última entidade representa o estoque propriamente dito.

Nesse exemplo, supomos que um pacote sempre contém três itens e que o estoque só pode manter nove itens por vez. Como o estoque é uma entidade com características passivas, ele foi modelado em BPMN como um *objeto de dados*, que não tem nenhum efeito no controle de fluxo do processo, mas que fornece informações adicionais sobre o que as tarefas precisam para serem executadas ou sobre o que elas produzem.

No modelo em GSPN da Figura 3.10, o estoque é modelado pelos lugares p_3 e p_6; p_3 mantém o número corrente de itens no estoque, enquanto p_6 indica o número de lugares disponíveis no estoque. A transição t_a é somente habilitada quando existe alguma ficha em p_6, ou seja, quando o estoque não está completo. O número inicial de fichas no lugar p_6 é dado por N – uma variável que representa o número máximo de itens que o estoque pode manter. Com essa abordagem, podemos facilmente alterar a capacidade do estoque sem mudar a estrutura do modelo em GSPN. Os itens produzidos são representados por fichas. Arcos ponderados são usados para modelar o requisito de

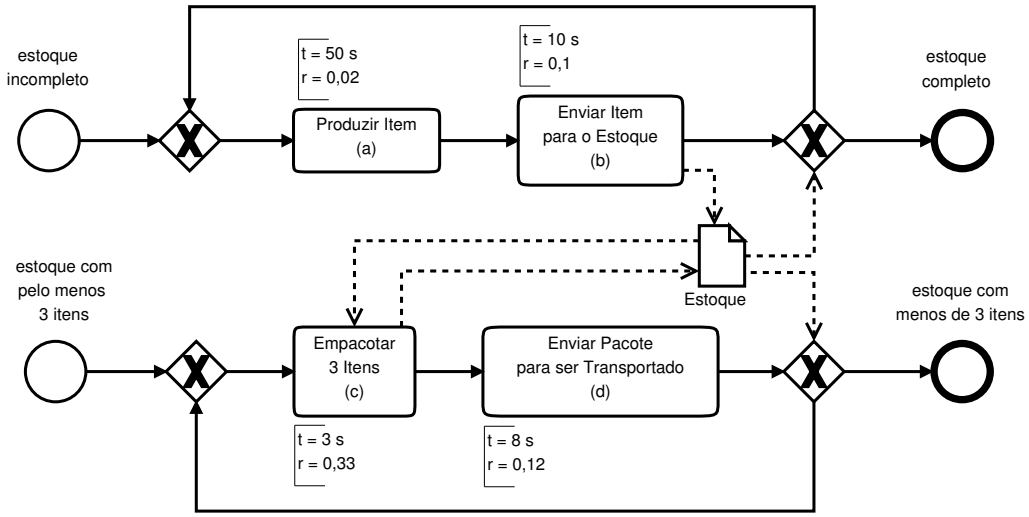
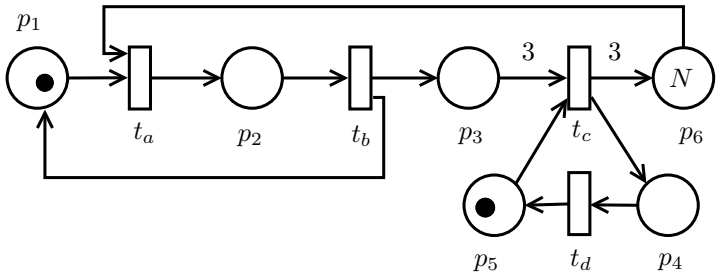


Figura 3.9: *Um processo simples de produção e empacotamento em BPMN.*



Transição Temporizada	t_a	t_b	t_c	t_d
Taxa	$0,02$	$0,1$	$0,33$	$0,12$

Figura 3.10: *Modelo em GSPN do processo de "produção/empacotamento".*

que todo pacote gerado deve possuir três itens.

Para modelar o exemplo em PEPA, nós temos que usar ações sincronizantes adicionais, como mostrado na Figura 3.11. O estoque é representado pelo conjunto de componentes `PEstoque0`, ... , `PEstoque9`. De forma geral, para representar um conjunto de recursos idênticos em PEPA, nós precisamos de pelo menos tantos componentes quantos forem os números possíveis de recursos em uso nos diferentes momentos da execução do processo. As execuções das ações `(a,r_a)`, em `PProdutor`, e `(c,r_c)`, em `PEmpacotador`, ocorrerão somente em cooperação com `PEstoque`.

No modelo em SAN da Figura 3.12, nós temos um autômato para representar cada entidade do sistema – $\mathcal{A}_1$ é o produtor, $\mathcal{A}_2$ é o empacotador e $\mathcal{A}_3$ representa o estoque. Em $\mathcal{A}_3$ nós temos um estado para cada número possível de itens no estoque. Para impedir a ocorrência do evento e_a (a produção de um item) quando o estoque já está cheio, nós podemos usar o poderoso conceito das transições funcionais de SAN. A taxa de e_a é definida em função do estado corrente do autômato $\mathcal{A}_3$. Se o estado de $\mathcal{A}_3$ é 3_9 (i.e., o estoque está cheio), então a taxa de e_a será 0 (ou seja, o evento não ocorrerá); de modo contrário, a taxa de e_a será $0,02$.

```
// Taxas de execução associadas às tarefas
r_a = 0.02; r_b = 0.1; r_c = 0.33; r_d = 0.12;

// Processo de Produção e Empacotamento
PProdutor = (a, r_a).(b, r_b).PProdutor;
PEmpacotador   = (c, r_c).(d, r_d).PEmpacotador;

// Estoque com número máximo de itens igual a 9
PEstoque  = (a,⊤).(b,⊤).PEstoque1;
PEstoque1 = (a,⊤).(b,⊤).PEstoque2;
PEstoque2 = (a,⊤).(b,⊤).PEstoque3;
PEstoque3 = (a,⊤).(b,⊤).PEstoque4 + (c,⊤).PEstoque;
PEstoque4 = (a,⊤).(b,⊤).PEstoque5 + (c,⊤).PEstoque1;
PEstoque5 = (a,⊤).(b,⊤).PEstoque6 + (c,⊤).PEstoque2;
PEstoque6 = (a,⊤).(b,⊤).PEstoque7 + (c,⊤).PEstoque3;
PEstoque7 = (a,⊤).(b,⊤).PEstoque8 + (c,⊤).PEstoque4;
PEstoque8 = (a,⊤).(b,⊤).PEstoque9 + (c,⊤).PEstoque5;
PEstoque9 = (c,⊤).PEstoque6;

PProdutor <a,b> PEstoque <c> PEmpacotador  // Sistema completo
```

Figura 3.11: *Modelo em PEPA do processo de "produção/empacotamento".*

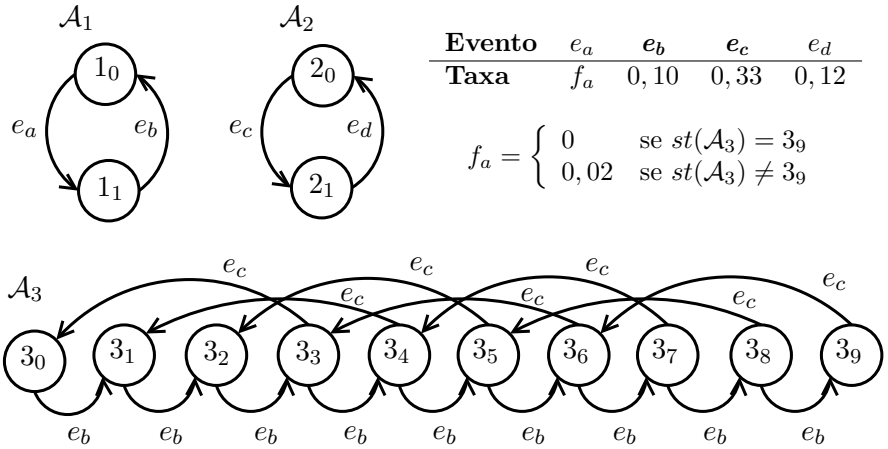


Figura 3.12: *Modelo em SAN do processo de "produção/empacotamento".*

Tabela 3.1: *Síntese da comparação dos formalismos e suas respectivas ferramentas.*

Critério de Avaliação	GSPN	PEPA	SAN
Expressividade	+	-	+
Poder de abstração	+	+/-	+
Expansibilidade	-	+	+/-
Legibilidade	-	+	+

3.2 Avaliação dos Formalismos

A Tabela 3.1 resume as vantagens e desvantagens dos formalismos, observadas nos mapeamentos dos cenários apresentados na Seção 3.1. Nós adotamos quatro critérios de avaliação:

- *expressividade* – a habilidade do formalismo para representar diretamente os cenários de processo de negócio;
- *poder de abstração* – a facilidade fornecida pelo formalismo para modelar os processos de negócio em um nível apropriado de generalidade;
- *extensibilidade* – a facilidade para estender um modelo no formalismo sem impactar o comportamento previamente modelado;
- *legibilidade* – a facilidade de compreender a lógica do negócio a partir do modelo.

Na Tabela 3.1, o símbolo "+" indica que o formalismo satisfatoriamente atende ao critério analisado; "+/-" indica que o formalismo atende ao critério com algumas restrições; "-" indica que o formalismo não atende suficientemente ao critério.

Como visto na Seção 3.1.1, as estruturas de controle de fluxo básicas usadas na modelagem de processos de negócio podem ser representadas nos três formalismos com facilidades equivalentes. Para modelar ramificações e junções mais avançadas, as transições imediatas das GSPN demostraram-se uma ferramenta importante. Elas facilitam a modelagem sem impactar os resultados da análise (como ilustrado no cenário da Seção 3.1.2). Dependências funcionais entre as tarefas de um processo podem ser diretamente expressas usando taxas funcionais (existentes tanto em GSPN quanto em SAN). A Seção 3.1.2 discutiu um caso particular de uso, mas as taxas funcionais são também interessantes para modelar casos em que as taxas das tarefas variam de acordo com a carga do sistema ou a quantidade de recursos disponíveis.

Cada um dos formalismos estudados fornecem diferentes tipos de abstração. A noção de lugares e marcações das GSPN possibilita a *abstração de parametrização* (que nos permitiu modelar o terceiro cenário – Seção 3.1.2 – de maneira mais flexível). As taxas funcionais também possibilitam a abstração de parametrização. O operador de encapsulamento de PEPA possibilita a *abstração comportamental*, enquanto as transições imediatas das GSPN possibilitam a *abstração temporal*. Tanto PEPA quanto SAN possibilitam a *abstração de composição*, que nos permite construir modelos complexos de forma incremental. Esse tipo de modelo nos fornece uma boa noção sobre como o sistema modelado pode ser implementado. No maior cenário usado neste capítulo (apresentado na Seção 3.1.1), é possível observar a importância do uso de composição na modelagem de processos de negócio.

No que se refere a legibilidade, em uma primeira observação, os modelos em GSPN parecem mais simples de serem compreendidos quando comparados aos modelos dos outros dois formalismos. Entretanto, eles requerem a compreensão de todos os detalhes envolvidos no modelo antes de serem capazes de fornecer uma visão geral da lógica do processo de negócio.

Processos de negócio são sistemas dinâmicos e frequentemente são construídos de forma modular. É desejável que os modelos de processos de negócio possam ser facilmente estendidos, para acomodar novos módulos sem impactar incorretamente o comportamento dos módulos já existentes. Nesse contexto, a composicionalidade de PEPA e SAN e o mecanismo de encapsulamento de PEPA são características valiosas.

A seguir, sintetizamos os prós e contras observados a partir do estudo comparativo e de outras características inerentes aos formalismos.

3.2.1 GSPN

Aspectos Positivos:

- Notação gráfica que fornece uma imagem clara do comportamento dinâmico do modelo;
- Lugares, marcações e transições funcionais, que possibilitam a abstração de parametrização;
- Transições imediatas, que possibilitam a abstração do tempo e facilitam a modelagem de decisões de roteamento ou sincronizações;
- Noção explícita de estados, que facilita a definição de índices de desempenho sobre os modelos.

Aspectos Negativos:

- Notação gráfica que transmite pouca noção sobre a estrutura do sistema e que dificulta a percepção geral da lógica de negócio modelada;
- Modelos em GSPN não são intrinsecamente composicionais. Essa característica dificulta a construção de modelos de grande escala ou a expansão de modelos já existentes.

3.2.2 PEPA

Aspectos Positivos:

- Construtores composicionais que permitem a modelagem do sistema a partir de componentes separados;
- Facilidade para o raciocínio sobre os modelos, já que as noções de equivalência são definidas em termos da semântica operacional.

Aspectos Negativos:

- O foco nas ações e a ausência da noção explícita de estado, que dificulta a abstração de parametrização;
- Ausência de mecanismos para a abstração do tempo;
- A ferramenta mais recomendada para PEPA (a PEPA *Plug-in Project*) não implementa ainda o conceito de taxas funcionais.

3.2.3 SAN

Aspectos Positivos:

- Modelos são construídos de forma composicional;
- Noção explícita de estados e transições funcionais, que possibilitam a abstração de parametrização;
- A matriz global da cadeia de Markov subjacente não é explicitamente gerada. As matrizes dos componentes individuais e informações referentes às interações entre os componentes são combinadas no descritor SAN (que é representado por operações da Álgebra Tensorial);
- Aptidão para a análise estruturada, já que o espaço de estados do sistema é representado como um produto de espaço de estados menores.

Aspectos Negativos:

- Ausência de mecanismos para abstração do tempo;
- O tempo de computação da solução do modelo pode ser excessivamente longo devido ao custo computacional para efetuar o produto descritor - vetor.

3.3 Conclusão do Capítulo

Neste capítulo, nós comparamos a aplicação de três formalismos (GSPN, PEPA e SAN) na modelagem de diferentes cenários de processos de negócio. As características desses cenários nos permitiram avaliar os formalismos sob diferentes critérios. Diante dos prós e contras levantados, optamos pelo uso de SAN como base do método de análise de desempenho de processos de negócio proposto nesta tese. SAN demonstrou-se um formalismo expressivo, que permite a modelagem e análise estruturada dos processos de negócio e que possui uma representação eficiente do espaço de estados.

Os resultados obtidos no estudo comparativo apresentado neste capítulo foram publicados em um relatório técnico [BFV09] e, posteriormente, nos anais do *Symposium On Theory of Modeling and Simulation – DEVS Integrative M&S Symposium* (DEVS'10), que integrou a *Spring Simulation Multiconference* (SpringSim'10) [BFV10].

Capítulo 4

Conversão de Modelos de Processos de Negócio em Modelos em SAN

Neste capítulo, nós mostramos que um diagrama de processo em BPMN pode dar origem a uma Rede de Autômatos Estocásticos (SAN). Nós definimos um algoritmo que automaticamente converte diagramas de processo BPMN, por meio de um conjunto de mapeamentos e operações que definimos sobre os modelos. Essa conversão é o primeiro passo para a construção de um modelo para avaliação de desempenho de um processo de negócio. Após a conversão, o modelo em SAN automaticamente gerado pode ser enriquecido com dados referentes ao tempo de execução médio das tarefas ou com os requisitos de recursos do processo de negócio modelado. Usando um solucionador para modelos em SAN, nós podemos extrair desse modelo variados índices de desempenho do processo de negócio.

A Seção 4.1 introduzirá as propriedades de um modelo em BPMN que consideramos válido como entrada para o nosso algoritmo de conversão. Na Seção 4.2 nós definimos formalmente a estrutura do grafo de um modelo em SAN resultante da nossa conversão, bem como as operações sobre esse grafo que sustentam o algoritmo descrito na Seção 4.3. Um exemplo de conversão de processo de negócio é dado na Seção 4.4. A conclusão do capítulo é feita na Seção 4.5.

4.1 Definição da Estrutura dos Modelos em BPMN

Como brevemente discutido na Seção 2.1.1, um diagrama de processo em BPMN é um grafo direcionado constituído de vértices de eventos, tarefas e desvios. Este trabalho se restringe a uma subclasse de todos os diagramas de processo que podem ser formados a partir dos objetos de BPMN. As definições de 4.1 a 4.5 descrevem formalmente essa subclasse.

Definição 4.1. *Grafo de Processo em BPMN*

Um grafo de processo em BPMN *(GP) é um grafo dirigido representado pela tupla* $GP = (V, E, L, \ell, \mathrm{p})$*, com* $V = S \cup A \cup G \cup F$*, em que:*

- S *é um conjunto de vértices representando os* eventos de início
- A *é um conjunto de vértices representando* tarefas atômicas
- G *é um conjunto de vértices representando* desvios
- F *é um conjunto de vértices representando* eventos de fim
- $E \subseteq (V \times V)$ *é o conjunto de* arcos

- L *é um conjunto de* rótulos de vértices
- $\ell : V \longrightarrow L$ *é uma* função de rotulação *de vértices*
- $\mathrm{p} : E' \longrightarrow [0,1]$*, em que* $E' \subseteq E$*, é uma* função parcial de probabilidade *que associa arcos a valores de probabilidade*

Um arco *em GP é um par* $\langle v, w \rangle$*, onde* $v, w \in V$*, indicando que existe um fluxo de sequência de* v *para* w*. Um* rótulo *é usado para denotar o nome de um evento ou tarefa modelada; no caso de um vértice de desvio, o rótulo indica o tipo do desvio.*

Definição 4.2. *Caminho em um Grafo de Processo em BPMN*

Seja $GP = (V, E, L, \ell, \mathrm{p})$ *um grafo de processo em BPMN e* $v_1, v_n \in V$*.*

Um caminho *de* v_1 *para* v_n *(representado por* $v_1 \rightsquigarrow v_n$*) é uma sequência de vértices* $v_1, v_2, \cdots, v_n$ *(com* $v_i \in V$*), tal que* $\langle v_i, v_{i+1} \rangle \in E$*, para* $0 < i < n$*.*

Definição 4.3. *Vértices de Entrada e Vértices de Saída*

Seja $GP = (V, E, L, \ell, \mathrm{p})$ *um grafo de processo em BPMN.*

As funções entradas $: V \to 2^V$ *e* saídas $: V \to 2^V$ *dão os* vértices de entrada *e os* vértices de saída*, respectivamente, de um vértice em GP e são definidas como*

$$\begin{array}{lrcl} \forall v \in V, & \text{entradas}(v) & = & \{u \in V |\ \langle u, v \rangle \in E\} \quad e \\ \forall v \in V, & \text{saídas}(v) & = & \{w \in V |\ \langle v, w \rangle \in E\} \quad . \end{array}$$

A notação BPMN não possui uma semântica operacional formalmente definida. Por essa razão, podemos construir modelos em BPMN que podem possuir mais de uma interpretação, sendo que algumas interpretações podem levar a situações de impasse na execução do processo.

Para minimizar as ambiguidades semânticas que as entidades existentes em BPMN podem introduzir no modelo e para garantir as propriedades que um modelo de processo de negócio precisa respeitar, nosso método de conversão faz algumas suposições sobre o modelo em BPMN fornecido como entrada. Essas suposições possibilitam a automatização da conversão.

Consideramos que um grafo de processo em BPMN bem formado é um grafo no qual:

- um vértice de evento de início pode possuir apenas um vértice de saída e nenhum vértice de entrada;
- um vértice de evento de fim pode possuir apenas um vértice de entrada e nenhum vértice de saída;
- um vértice de tarefa pode possuir apenas um vértice de entrada e apenas um vértice de saída;
- todo vértice de desvio deve possuir um dos seguintes rótulos "+", para a ativação incondicional em paralelo; "◯", para a ativação inclusiva condicional; e "×", para o desvio condicional exclusivo;
- um vértice de desvio pode desempenhar o papel de uma divergência ou de uma convergência (mas não os dois papéis ao mesmo tempo). Como consequência, um vértice de desvio pode ter ou um único vértice de entrada e mais de um de saída (caso da divergência), ou um único vértice de saída e mais de um de entrada (caso da convergência);

- para todo vértice v de tarefa ou desvio deve valer que:
  1. existe um caminho de um vértice de evento de início para v ;
  2. existe um caminho de v para um vértice de evento de fim.
- todo arco que parte de um vértice de ativação inclusiva condicional ou desvio condicional exclusivo deve ter um valor de probabilidade associado a ele;
- a soma dos valores de probabilidade dos arcos que saem de um vértice de desvio condicional exclusivo deve ser 1.

A Definição 4.4 formaliza o conceito de grafo bem formado.

Definição 4.4. *Grafo de Processo em BPMN Bem Formado*
Seja $GP = (V, E, L, \ell, \mathrm{p})$ *um grafo de processo em BPMN, com* $V = S \cup A \cup G \cup F$.
GP *é um* grafo de processo em BPMN bem formado *se e somente se:*

- $\forall v \in S,\ (|\,\text{entradas}(v)| = 0) \wedge (|\,\text{saídas}(v)| = 1)$
- $\forall v \in F,\ (|\,\text{entradas}(v)| = 1) \wedge (|\,\text{saídas}(v)| = 0)$
- $\forall v \in A,\ (|\,\text{entradas}(v)| = 1) \wedge (|\,\text{saídas}(v)| = 1)$
- $\forall v \in G,\ (\ell(v) = \text{“}+\text{”}) \vee (\ell(v) = \text{“}\bigcirc\text{”}) \vee (\ell(v) = \text{“}\times\text{”})$
- $\forall v \in G,\ ((|\,\text{entradas}(v)| > 1) \wedge (|\,\text{saídas}(v)| = 1)) \vee$
  $((|\,\text{entradas}(v)| = 1) \wedge (|\,\text{saídas}(v)| > 1))$
- $\forall v \in A \cup G,\ \exists s \in S$ *tal que existe um caminho* $s \rightsquigarrow v$
- $\forall v \in A \cup G,\ \exists f \in F$ *tal que existe um caminho* $v \rightsquigarrow f$
- $\forall v \in G$ *tal que* $(\ell(v) = \text{“}\times\text{”}) \vee (\ell(v) = \text{“}\bigcirc\text{”}),\ \forall w \in \text{saídas}(v),\ \mathrm{p}(\langle v, w\rangle)$ *precisa estar definido*
- $\forall v \in G$ *tal que* $\ell(v) = \text{“}\times\text{”},\ \sum_{w\ \in\ \text{saídas}(v)} \mathrm{p}(\langle v, w\rangle) = 1$

É importante ressaltarmos que a associação de arcos com valores de probabilidade não existe na especificação de BPMN. Nós introduzimos essa característica na nossa definição porque a dinâmica de um processo de negócio é probabilística por essência. Essas probabilidades são quantificações do comportamento do processo de negócio.

A Definição 4.4 se refere a propriedades sintáticas do modelo em BPMN, ou seja, ela impõe restrições estruturais sobre o grafo do processo. Mas existem também algumas propriedades semânticas importantes que precisamos garantir para termos um modelo de processo em BPMN bem definido. Essas propriedades são descritas pela Definição 4.5. Em um modelo de processo em BPMN bem definido, duas propriedades importantes para um processo de negócio são garantidas:

1. o modelo de processo não contém tarefas inatingíveis;
2. o processo pode sempre terminar.

Definição 4.5. *Modelo de Processo em BPMN Bem Definido*

Um modelo de processo em BPMN bem definido *é um grafo de processo em BPMN bem formado no qual:*

- *um desvio condicional exclusivo não converge (unifica) fluxos de sequência paralelos;*
- *uma ativação incondicional em paralelo não converge (sincroniza) fluxos de sequência exclusivos;*
- *uma ativação inclusiva condicional somente converge (entrelaça) fluxos de sequência originados por alguma outra ativação inclusiva condicional. Além disso, há uma correspondência um-para-um entre as ativações inclusivas condicionais divergentes e as convergentes.*

4.2 Definição da Estrutura dos Modelos em SAN e suas Operações

As definições de 4.6 a 4.11 especificam formalmente a estrutura de um modelo em SAN e as operações aplicáveis sobre ele na forma em que são utilizados em nosso algoritmo de conversão.

Definição 4.6. *Modelo em SAN e Autômato em SAN*

Um modelo em SAN $\mathcal{S}$ *é um conjunto* $\mathcal{S} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ *de* N *autômatos em SAN.*

Um autômato em SAN $\mathcal{A}$ *é dado pela tupla* $\mathcal{A} = (Q, E, T, L, \ell, \mathrm{p})$*, em que:*

- Q *é um conjunto de* estados
- E *é um conjunto de* eventos
- $T \subseteq (Q \times Q \times E)$ *é um conjunto de* transições de estados rotuladas por eventos
- L *é um conjunto de* rótulos de estados
- $\ell : Q \longrightarrow L$ *é uma* função de rotulação *de estados*
- $\mathrm{p} : T' \longrightarrow [0, 1]$*, em que* $T' \subseteq T$*, é uma* função parcial de probabilidade *que associa uma transição a um valor de probabilidade*

Definição 4.7. *Transições de Saída e Transições de Entrada*

Seja $\mathcal{A} = (Q, E, T, L, \ell, \mathrm{p})$ *um autômato em SAN.*

As funções entradas $: Q \to 2^T$ *e* saídas $: Q \to 2^T$ *que devolvem as* transições de entrada *e as* transições de saída*, respectivamente, de um estado de* $\mathcal{A}$ *são definidas como*

$$\begin{aligned} \forall q \in Q, \quad \text{entradas}(q) &= \{\langle p, q, e\rangle \mid \langle p, q, e\rangle \in T\} \quad e \\ \forall q \in Q, \quad \text{saídas}(q) &= \{\langle q, r, e\rangle \mid \langle q, r, e\rangle \in T\} \quad . \end{aligned}$$

Definição 4.8. *Estado Fonte e Estado Absorvente*

Seja $\mathcal{A} = (Q, E, T, L, \ell, p)$ *um autômato em SAN e* $q \in Q$ *um estado de* $\mathcal{A}$*.*

Se $\text{entradas}(q) = \emptyset$*, então* q *é um* estado fonte*.*

Se $\text{saídas}(q) = \emptyset$*, então* q *é um* estado absorvente*.*

Neste trabalho, nós propomos um algoritmo para converter automaticamente um modelo de processo em BPMN bem definido (Definição 4.4) em um modelo em SAN no formato da Definição 4.6. Essa conversão começa com o mapeamento individual dos objetos do grafo em BPMN para objetos em SAN, de acordo com os mapeamentos básicos definidos na Tabela 4.1. De forma geral, podemos dizer que o mapeamento de cada vértice do grafo em BPMN gera no modelo em SAN ao menos um novo autômato, com pelo menos dois estados e uma transição entre eles (associada a um novo evento rotulado com o identificador do vértice BPMN). Vértices de eventos, tarefas e desvios condicionais exclusivos geram somente eventos locais. Vértices de ativações incondicionais em paralelo geram somente eventos sincronizantes. Vértices de ativações inclusivas condicionais (que precisam aparecer aos pares, delimitando blocos fechados) geram tanto eventos locais quanto eventos sincronizantes.

Fluxos de sequência paralelos são mapeados como conjuntos de autômatos sincronizados. Uma escolha é expressa por meio de um estado que possua mais de uma transição de saída (lembrando que, nesse caso, cada transição de saída deve ser ponderada por um valor de probabilidade). Uma tarefa atômica a é mapeada em um autômato com três estados sequenciais. O primeiro estado indica que a tarefa está à espera da disponibilidade dos recursos dos quais depende para ser executada; o segundo estado indica que a tarefa já obteve acesso aos recursos e está em curso de execução (pelo evento r_a); o terceiro estado indica que a execução da tarefa está finalizada (com a ocorrência do evento a).

Para obter o modelo em SAN correspondente a um modelo de processo em BPMN bem definido, nós precisamos compor os autômatos gerados a partir dos mapeamentos da Tabela 4.1. Essa composição é feita por meio de um conjunto de *operações de simplificação* aplicáveis sobre modelos em SAN. Essas operações são definidas a seguir.

Definição 4.9. *Operação de Fusão de Estados* ($\triangleright$)

Seja $\mathcal{A} = (Q, E, T, L, \ell, \mathrm{p})$ *um autômato em SAN e sejam* $q_1, q_2 \in Q$ *dois estados em* $\mathcal{A}$.

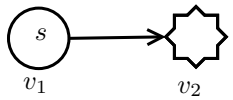
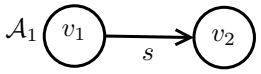
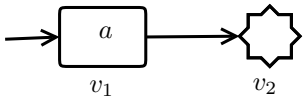
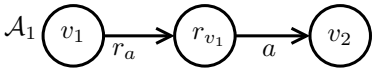
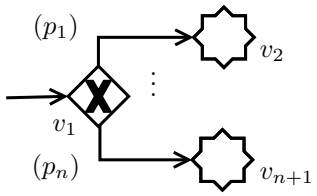
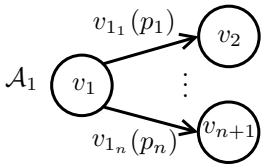
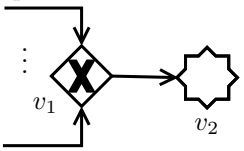
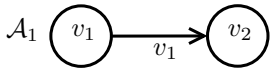
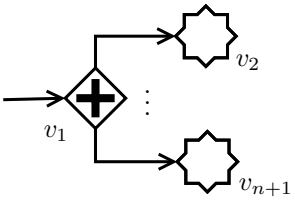
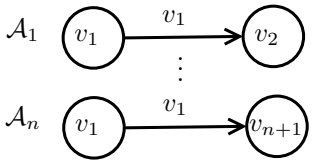
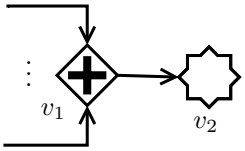
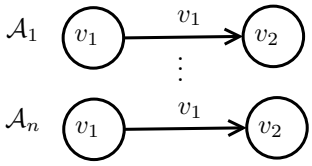
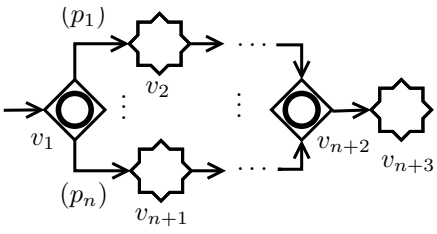
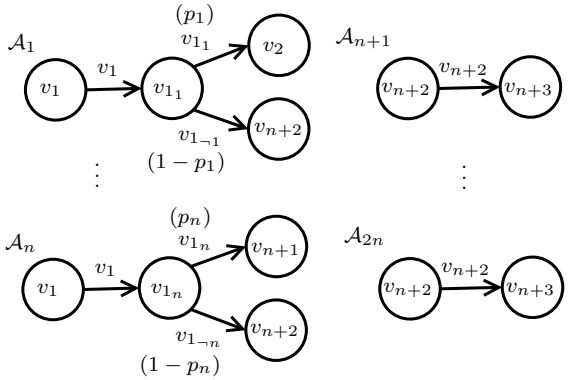
A fusão dos estados q_1 *e* q_2 *em* $\mathcal{A}$ *(representada por* $\mathcal{A}[q_1 \triangleright q_2]$*) resulta em um autômato em SAN* $\mathcal{A}_F = (Q_F, E_F, T_F, L_F, \ell_F, \mathrm{p}_F)$ *tal que:*

- $Q_F = Q \setminus \{q_2\}$
- $E_F = E$
- $T_F = \{\langle p, q, e\rangle \in T \mid (p \neq q_2) \wedge (q \neq q_2)\} \cup \{\langle p, q_1, e\rangle \mid \langle p, q_2, e\rangle \in T\} \cup \{\langle q_1, q, e\rangle \mid \langle q_2, q, e\rangle \in T\}$
- $L_F = L$
- $\forall q \in Q_F, \ell_F(q) = \ell(q)$
- $\forall \langle p, q, e\rangle \in T_F,$

$$\mathrm{p}_F(\langle p, q, e\rangle) = \begin{cases} \mathrm{p}(\langle p, q, e\rangle) & \text{se } \langle p, q, e\rangle \in T \\ \mathrm{p}(\langle p, q_2, e\rangle) & \text{se } (q = q_1) \wedge (\langle p, q_2, e\rangle \in T) \\ \mathrm{p}(\langle q_2, q, e\rangle) & \text{se } (p = q_1) \wedge (\langle q_2, q, e\rangle \in T) \end{cases}$$

Essa operação elimina q_2 *de* $\mathcal{A}$*, transformando todas as transições de entrada ou saída de* q_2 *em transições de entrada ou saída de* q_1 *(mantendo o rótulo de* q_1 *inalterado).*

Tabela 4.1: *Mapeamento dos objetos de BPMN em SAN.*

Objeto BPMN[1,2]	**Mapeamento para SAN**
Evento de início rotulado por s 	
Tarefa atômica rotulada por a 	
Desvio condicional exclusivo divergindo 1 fluxo de sequência em n 	
Desvio condicional exclusivo convergindo n fluxos de sequência em 1 	
Ativação incondicional em paralelo divergindo 1 fluxo de sequência em n 	
Ativação incondicional em paralelo convergindo n fluxos de sequência em 1 	
Bloco de ativação inclusiva condicional divergindo 1 fluxo de sequência em n e convergindo-os em 1 	

[1] Estamos usando o símbolo ⬠ para expressar qualquer vértice válido em um modelo em BPMN bem definido (i.e., uma *tarefa atômica*, um *evento de fim* ou um *desvio*).

[2] Nessa representação gráfica dos grafos em BPMN, há duas anotações textuais nos vértices: um rótulo (que aparece dentro do vértice) e um identificador v_i (que aparece embaixo do vértice).

Definição 4.10. *Operação de Supressão de Estado* ($\blacktriangleright$)

Seja $\mathcal{A} = (Q, E, T, L, \ell, \mathrm{p})$ *um autômato em SAN e* $q \in Q$ *um estado de* $\mathcal{A}$ *tal que* $|\,\text{saídas}(q)| = 1$. *Seja* t_s *a transição de saída de* q *e* $\boldsymbol{s}$ *o estado de saída de* q.

A supressão do estado q *em* $\mathcal{A}$ *(representado por* $\mathcal{A}[q \blacktriangleright]$*) resulta em um autômato em SAN* $\mathcal{A}_S = (Q_S, E_S, T_S, L_S, \ell_S, \mathrm{p}_S)$ *tal que:*

- $Q_S = Q \setminus \{q\}$
- $E_S = \{e \in E \mid \langle p, r, e\rangle \in (T \setminus \{t_s\})\}$
- $T_S = \{\langle p, r, e\rangle \in T \mid (p \neq q) \wedge (r \neq q)\} \cup \{\langle p, s, e\rangle \mid \langle p, q, e\rangle \in T\}$;
- $L_S = \{l \in L \mid \exists p \in Q, ((\ell(p) = l) \wedge (p \neq q))\}$
- $\forall q \in Q_S, \ell_S(q) = \ell(q)$
- $\forall \langle p, r, e\rangle \in T_S$,

$$\mathrm{p}_S(\langle p, r, e\rangle) = \begin{cases} \mathrm{p}(\langle p, r, e\rangle) & \textit{se} \quad \langle p, r, e\rangle \in T \\ \mathrm{p}(\langle p, q, e\rangle) & \textit{nos demais casos} \end{cases}$$

Essa operação elimina q *e sua transição de saída de* $\mathcal{A}$*, transformando todas as transições de entrada de* q *em transições de entrada de seu único estado de saída* $\boldsymbol{s}$.

Definição 4.11. *Operação de Concatenação de Autômatos* ($\boxplus$)

Sejam $\mathcal{A}_1 = (Q_1, E_1, T_1, L_1, \ell_1, \mathrm{p}_1)$ *e* $\mathcal{A}_2 = (Q_2, E_2, T_2, L_2, \ell_2, \mathrm{p}_2)$ *dois autômatos em SAN. Sejam* $q_1 \in Q_1$ *um estado absorvente em* $\mathcal{A}_1$ *e* $q_2 \in Q_2$ *um estado fonte em* $\mathcal{A}_2$.

A concatenação de $\mathcal{A}_1$ e $\mathcal{A}_2$ via os estados q_1 e q_2 *(representado por* $\mathcal{A}_1 \underset{q_2}{\overset{q_1}{\boxplus}} \mathcal{A}_2$ *) é o autômato em SAN* $\mathcal{A}_C = (Q_C, E_C, T_C, L_C, \ell_C, p_C)$*, em que:*

- $Q_C = (Q_1 \cup Q_2) \setminus \{q_2\}$
- $E_C = E_1 \cup E_2$
- $T_C = T_1 \cup \{\langle q_1, p, e\rangle \mid \langle q_2, p, e\rangle \in T_2\} \cup (T_2 \setminus \{\langle q_2, p, e\rangle \mid \langle q_2, p, e\rangle \in T_2\})$
- $L_C = L_1 \cup L_2$
- $\forall q \in Q_C, \ell_C(q) = \begin{cases} \ell_1(q) & \textit{se} \quad q \in Q_1 \\ \ell_2(q) & \textit{nos demais casos} \end{cases}$
- $\forall t = \langle p, q, e\rangle \in T_C$,

$$\mathrm{p}_C(t) = \begin{cases} \mathrm{p}_1(t) & \textit{se} \quad t \in T_1 \\ \mathrm{p}_2(\langle q2, q, e\rangle) & \textit{se} \quad (p = q1) \wedge (\langle q_2, q, e\rangle \in T_2) \\ \mathrm{p}_2(t) & \textit{nos outros casos} \end{cases}$$

Esta operação cria um autômato que é o resultado da união de $\mathcal{A}_1$ *e* $\mathcal{A}_2$ *por meio da fusão dos estados* q_1 *e* q_2*. No autômato resultante dessa operação,* q_2 *é eliminado e todas as suas transições de saída se transformam em transições de saída de* q_1.

4.3 Algoritmo de Conversão

O Algoritmo 1 mostra os principais passos envolvidos na conversão de um modelo de processo em BPMN bem definido para um modelo em SAN.

Algoritmo 1 ConverteBPMNparaSAN(GP)

entrada: GP – um grafo de processo em BPMN bem formado que seja também um modelo de processo em BPMN bem definido

saída: $\mathcal{S}$ – um modelo em SAN

1: $\mathcal{S} \leftarrow \emptyset$
2: $V \leftarrow S_{GP} \cup A_{GP} \cup G_{GP} \cup F_{GP}$ {Todos os vértices do grafo GP}
3: **para cada** $v \in V$ **faça**
4: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup$ ConverteVerticeEmAutomatos(GP, vertex, $\mathcal{S}$)
5: **fim para**
6: ReduzModeloSAN(GP, $\mathcal{S}$)
7: **devolva** $\mathcal{S}$

Procedimento 1 ReduzModeloSAN(GP, $\mathcal{S}$)

entrada: GP – um grafo em BPMN bem formado

entrada/saída: $\mathcal{S}$ – um modelo em SAN

1: { Concatena os autômatos "sequenciais" }
2: **enquanto** existir um estado absorvente $q_1 \in \mathcal{A}_1$ e um estado fonte $q_2 \in \mathcal{A}_2$ (com $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{S}$) tal que $\ell_{\mathcal{A}_1}(q_1) = \ell_{\mathcal{A}_2}(q_2)$ **faça**
3: $\quad \mathcal{A}_C \leftarrow \mathcal{A}_1 \overset{q_1}{\underset{q_2}{\boxplus}} \mathcal{A}_2$ { Concatena os dois autômatos }
4: $\quad$ { Funde os estados equivalentes, correspondentes aos desvios incondicionais exclusivos }
5: $\quad$ **enquanto** $\exists\ q_1, q_2 \in Q_{\mathcal{A}_C}$ tal que $\ell_{\mathcal{A}_C}(q_1) = \ell_{\mathcal{A}_C}(q_2)$ **faça**
6: $\quad\quad \mathcal{A}_C \leftarrow \mathcal{A}_C[q_1 \rhd q_2]$
7: $\quad$ **fim enquanto**
8: $\quad \mathcal{S} \leftarrow (\mathcal{S} \setminus \{\mathcal{A}_1, \mathcal{A}_2\}) \cup \mathcal{A}_C$
9: **fim enquanto**
10: **para cada** $\mathcal{A} \in \mathcal{S}$ **faça**
11: $\quad$ { Remove os estados criados para ligar fluxos de sequência alternativos }
12: $\quad$ **enquanto** $\exists\ q \in Q_{\mathcal{A}}$ e $\exists v \in G_{GP}$ tal que: $(\ell_{\mathcal{A}}(q) = v \ \wedge\ \ell_{GP}(v) = \text{“}\times\text{”} \ \wedge\ |\text{saídas}(v)| = 1)$ **faça**
13: $\quad\quad \mathcal{A} \leftarrow \mathcal{A}[q \blacktriangleright]$ { Suprime o estado q}
14: $\quad$ **fim enquanto**
15: $\quad$ { Funde o estado fonte com os estados absorventes, para criar autômatos cíclicos }
16: $\quad$ **enquanto** $\exists\ q_2 \in Q_{\mathcal{A}}$ tal que q_2 é um estado absorvente **faça**
17: $\quad\quad q_1 \leftarrow$ o único estado fonte de $\mathcal{A}$
18: $\quad\quad \mathcal{A} \leftarrow \mathcal{A}[q_1 \rhd q_2]$ { Funde os estados q_1 e q_2 }
19: $\quad$ **fim enquanto**
20: **fim para**

Primeiramente, o algoritmo cria um modelo em SAN composto por todos os autômatos gerados a partir da conversão individual dos vértices do grafo em BPMN fornecido como entrada. Essa conversão individual, feita pela função "ConverteVerticeEmAutomatos", é a implementação dos mapeamentos descritos na Tabela 4.1. Em seguida, o modelo em SAN é reduzido por meio do Procedimento 1, que aplica as operações definidas na Seção 4.2 para criar o modelo em SAN final.

O primeiro bloco de reduções do Procedimento 1 (linhas de 2 a 9) consiste na concatenação exaustiva de autômatos, para a definição dos fluxos de sequência do processo. As concatenações são feitas por meio de estados que possuam o mesmo rótulo, ou seja, de estados que representam elementos que estão diretamente conectados entre si no modelo em BPMN. Cada autômato resultante de uma concatenação passa pela eliminação de estados redundantes, que é feita por meio da aplicação exaustiva da operação de fusão de estados sobre os estados do autômato que possuam rótulos iguais. Ao término desse bloco de reduções, teremos tantos autômatos no modelo em SAN quantos forem os fluxos de sequência paralelos no processo.

O objetivo do segundo bloco de reduções do Procedimento 1 (linhas de 12 a 14) é eliminar dos autômatos os estados desnecessários no modelo, criados para a convergência de fluxos de sequência alternativos (como os criados por um desvio incondicional exclusivo).

O terceiro e último bloco de reduções do Procedimento 1 (linhas de 16 a 19) faz a fusão entre os estados absorventes e o estado fonte em cada autômato do modelo em SAN. Isso é feito para que tenhamos autômatos cíclicos, uma condição necessária para que o modelo em SAN seja bem formado. É importante observar que, antes desse último bloco de reduções, todo autômato do modelo tem apenas um estado fonte. Todos os autômatos criados no mapeamento dos vértices do modelo em BPMN sempre possuem uma só fonte (ver Tabela 4.1) e as operações aplicadas sobre eles nas reduções dos blocos 1 e 2 não alteram essa característica.

O número final de autômatos em um modelo em SAN gerado pelo nosso método de conversão a partir de um modelo em BPMN bem definido e denotado por GP é dado pela fórmula

$$|S_{GP}| + \sum_{g \in G'} (|\,\text{saídas}(g)| - 1)$$

em que

$$G' = \{g \in G_{GP} \mid (|\,\text{saídas}(g)| > 1) \wedge (\ell_{GP}(g) \in \{\text{“}\times\text{”}, \text{“}\bigcirc\text{”}\})\}.$$

Cada autômato do modelo em SAN gerado para o processo de negócio pode ser visto como um fluxo de sequência independente. Por essa razão, nós temos ao menos tantos autômatos no modelo quanto for o número de eventos de início em GP ($|S_{GP}|$). Além disso, para cada vértice de desvio divergente do tipo ativação incondicional em paralelo ou ativação inclusiva condicional, um novo conjunto de autômatos é necessário. O tamanho desse conjunto é dado pelo número de ramos (saídas) do vértice de desvio divergente menos um (porque um dos ramos é tratado como a continuação do fluxo de sequência que deu origem à divergência).

É importante ressaltarmos duas características desse algoritmo de conversão. A primeira delas é que os modelos em SAN gerados pelo ele não são os únicos possíveis para representar os modelos em BPMN fornecidos como entrada. A segunda, é que a ordem na qual os pares de autômatos são selecionados para a concatenação (linhas 2 e 3 do Procedimento 1) impacta o modelo final. Diferentes ordens podem gerar modelos estruturalmente diferentes, mas equivalentes em termos do comportamento modelado. Isso também pode implicar diferentes níveis de dificuldade na solução

numérica do modelo (alguns modelos podem consumir mais recursos computacionais que outros).

A dificuldade na resolução de um modelo está diretamente relacionada ao seu número de estados atingíveis. Como o número total de estados de um modelo em SAN é dado pelo produto do número de estados de todos os seus autômatos, é difícil afirmar para todos os casos que menos autômatos com mais estados são preferíveis a mais autômatos com menos estados (ou vice-versa). Por essa razão, nós decidimos não favorecer explicitamente nenhuma dessas duas situações no nosso algoritmo.

O modelo em SAN gerado por nosso algoritmo reflete o comportamento de uma única instância de processo de negócio. Para analisar o comportamento do sistema quando várias instâncias são executadas paralelamente, nós precisamos replicar os autômatos do modelo em SAN. Cada réplica do modelo representa uma instância do processo. SAN possui o conceito de replicação e técnicas para agregar componentes similares, que permite reduzir o espaço de estados do modelo [Bre04, BBFP04]).

4.3.1 Complexidade Computacional

O tempo necessário para a conversão de um modelo de processo em BPMN bem definido para um modelo em SAN é $O(|V|^2)$, sendo V o conjunto de todos os vértices do modelo fornecido como entrada para o algoritmo.

O tempo das operações nas linhas de 1 a 5 no Algoritmo 1 é $O(|V|)$, uma vez que a soma do número de estados dos autômatos gerados na conversão dos vértices é uma função linear em $|V|$.

O tempo do Procedimento 1 é dado pela soma dos tempos dos seus três blocos de operações. O primeiro bloco, entre as linhas 2 e 9, trata das concatenações de autômatos e das fusões de estados nos autômatos resultantes. Mas antes de realizar uma concatenação, existe a busca pelos pares de autômatos $\mathcal{A}_1$ e $\mathcal{A}_2$ que satisfazem a condição necessária para serem concatenados. Essa busca consome um tempo $O(|V|^2)$, já que o número total de estados no conjunto de autômatos é $O(|V|)$.

O tempo de uma concatenação $\mathcal{A}_1 \overset{q_1}{\underset{q_2}{\boxplus}} \mathcal{A}_2$ é $O(|V|)$, já que $|\text{entradas}(q_1)|$ e $|\text{saídas}(q_2)|$ são sempre menores que a soma total do número de estados dos autômatos. O número máximo de concatenações que podemos ter é determinado pelo número de autômatos gerados na conversão dos vértices do modelo em BPMN, que também é $O(|V|)$.

Toda concatenação de autômatos no Procedimento 1 é sucedida por operações de fusão de estados redundantes, cujo custo é limitado pelo número de estados que um autômato do modelo pode ter, ou seja, esse custo é $O(|V|)$. Sendo assim, o custo das operações do bloco entre as linhas 2 e 9 é $O(|V|^2)$.

O segundo bloco, compreendido entre as linhas 12 e 14, faz a supressão de estados nos autômatos restantes no modelo. Como a operação de supressão leva um tempo constante, o tempo desse bloco é $O(|V|)$.

No terceiro bloco, entre as linhas 16 e 19, para todo autômato restante no modelo em SAN, temos uma operação de fusão de seus estados absorventes com o seu único estado fonte. Como o custo de uma operação de fusão é $O(|V|)$ e o número de estados absorventes também é limitado pela soma do número de estados dos autômatos no modelo, temos que o custo desse bloco é $O(|V|^2)$.

Portanto, o tempo total da execução do Algoritmo 1 é $O(|V|^2)$.

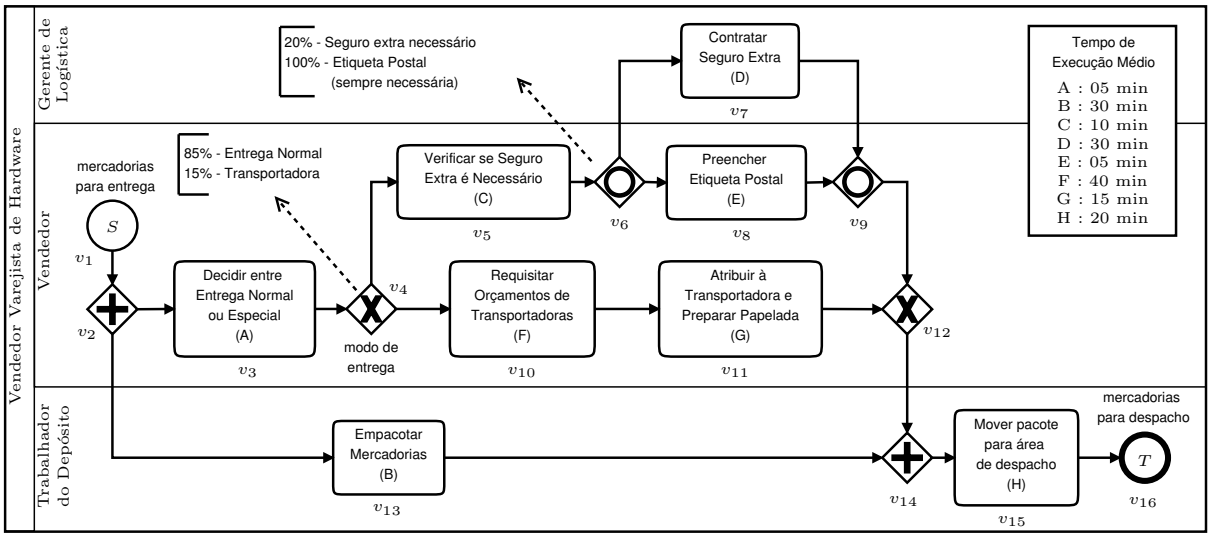


Figura 4.1: *Modelo em BPMN do processo de despacho de produtos de um varejo de hardware [OMG11a].*

4.3.2 Implementação

O método de conversão foi implementado como núcleo de uma ferramenta computacional, a `BP2SAN` [BP2]. Essa ferramenta recebe como parâmetro de entrada um modelo em BPMN (textualmente descrito usando a linguagem DOT [GN00]) e gera modelos em SAN comportamentalmente equivalentes ao processo de negócio dado como entrada. Os modelos em SAN resultantes são textualmente expressos na linguagem aceita pela ferramenta PEPS [BFPS07].

Os modelos em SAN automaticamente gerados podem ser posteriormente enriquecidos, por exemplo, com informações sobre as taxas de execução associadas aos eventos, ou com autômatos adicionais e taxas funcionais para expressar restrições de recursos.

Como se pode imaginar, definir as taxas e probabilidades a serem associadas com cada evento do modelo estocástico de um processo de negócio pode não ser uma tarefa trivial. Em alguns casos, as taxas e probabilidades podem ser determinadas por um especialista de negócio, com base em seu conhecimento do domínio. Em outros casos, quando o sistema real já está implementado e em uso, esses valores podem se basear nos tempos e frequências observados no sistema real.

4.4 Exemplo de Conversão: Processo de Entrega de um Varejo de Hardware

A Figura 4.1 mostra um modelo em BPMN extraído dos exemplos indicados no documento de especificação da BPMN 2.0 [OMG11a]. Esse modelo exemplifica os conceitos mais importantes da modelagem de processos de negócio em BPMN. Ele representa os passos que um varejo de hardware deve cumprir antes que a mercadoria possa ser enviada ao cliente. O modelo possui três compartimentos, cada um rotulado com o empregado encarregado pela execução das tarefas contidas no compartimento: *Gerente de Logística*, *Vendedor* e *Trabalhador do Depósito*. O modelo em SAN gerado pela conversão dos vértices na Figura 4.1 é mostrado na Figura 4.2.

Depois da conversão dos vértices, o modelo em SAN é reduzido de acordo com os passos definidos no Procedimento 1. A Figura 4.3 mostra uma das possíveis sequências de operações de redução que pode ser aplicada sobre o modelo em SAN da Figura 4.2. O modelo resultante dessa sequência de operações é o mostrado na Figura 4.4.

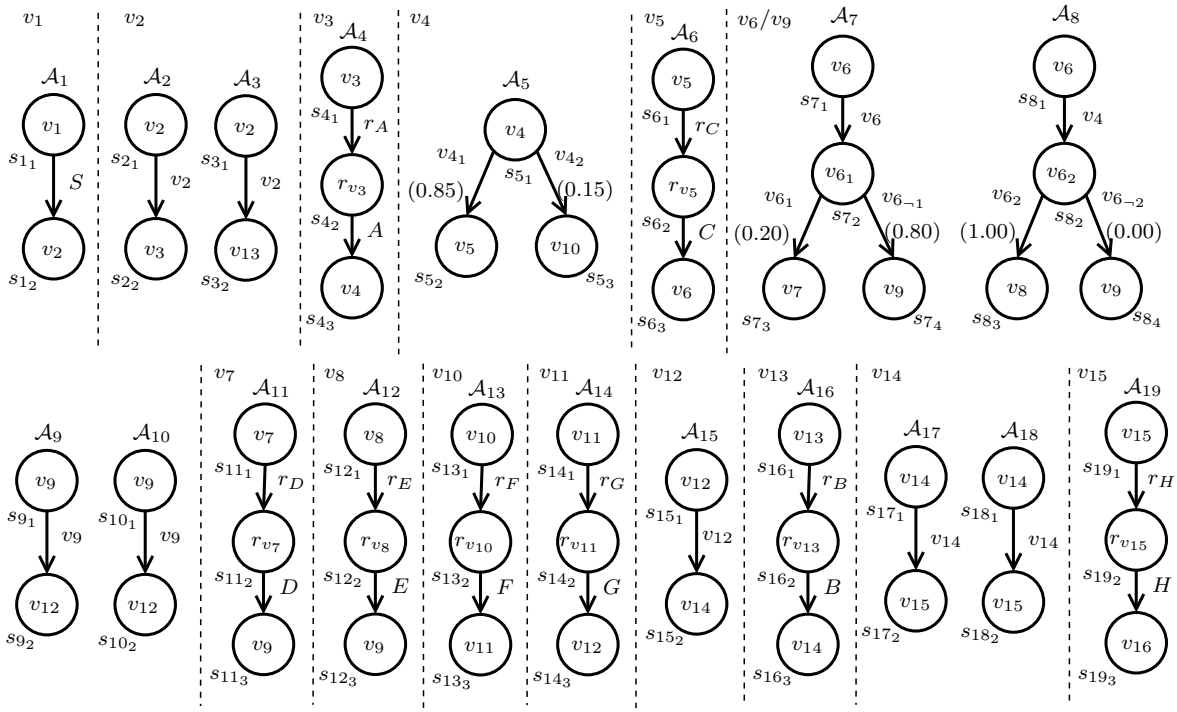


Figura 4.2: *Modelo em SAN obtido após a conversão dos vértices do grafo BPMN da Figura 4.1.*

$$\mathcal{A}' = (((\mathcal{A}_1 \underset{s_{2_1}}{\overset{s_{1_2}}{\boxplus}} \mathcal{A}_2 \underset{s_{4_1}}{\overset{s_{2_2}}{\boxplus}} \mathcal{A}_4 \underset{s_{5_1}}{\overset{s_{4_3}}{\boxplus}} \mathcal{A}_5 \underset{s_{6_1}}{\overset{s_{5_2}}{\boxplus}} \mathcal{A}_6 \underset{s_{7_1}}{\overset{s_{6_3}}{\boxplus}} \mathcal{A}_7 \underset{s_{9_1}}{\overset{s_{7_4}}{\boxplus}} \mathcal{A}_9 \underset{s_{11_1}}{\overset{s_{7_3}}{\boxplus}} \mathcal{A}_{11})[s_{7_4} \rhd s_{11_3}] \underset{s_{13_1}}{\overset{s_{5_3}}{\boxplus}} \mathcal{A}_{13} \underset{s_{14_1}}{\overset{s_{13_3}}{\boxplus}} \mathcal{A}_{14})[s_{9_2} \rhd s_{14_3}] \underset{s_{15_1}}{\overset{s_{9_2}}{\boxplus}} \mathcal{A}_{15} \underset{s_{17_1}}{\overset{s_{15_2}}{\boxplus}} \mathcal{A}_{17} \underset{s_{19_1}}{\overset{s_{17_2}}{\boxplus}} \mathcal{A}_{19})[s_{9_2} \blacktriangleright][s_{1_1} \rhd s_{19_3}]$$

$$\mathcal{A}'' = (\mathcal{A}_3 \underset{s_{16_1}}{\overset{s_{3_2}}{\boxplus}} \mathcal{A}_{16} \underset{s_{18_1}}{\overset{s_{16_3}}{\boxplus}} \mathcal{A}_{18})[s_{3_1} \rhd s_{18_2}] \quad \mathcal{A}''' = (\mathcal{A}_8 \underset{s_{10_1}}{\overset{s_{8_4}}{\boxplus}} \mathcal{A}_{10} \underset{s_{12_1}}{\overset{s_{8_3}}{\boxplus}} \mathcal{A}_{12})[s_{10_1} \rhd s_{12_3}][s_{8_1} \rhd s_{10_2}]$$

Figura 4.3: *Sequência de operações de redução que podem ser aplicadas sobre o modelo em SAN da Figura 4.2.*

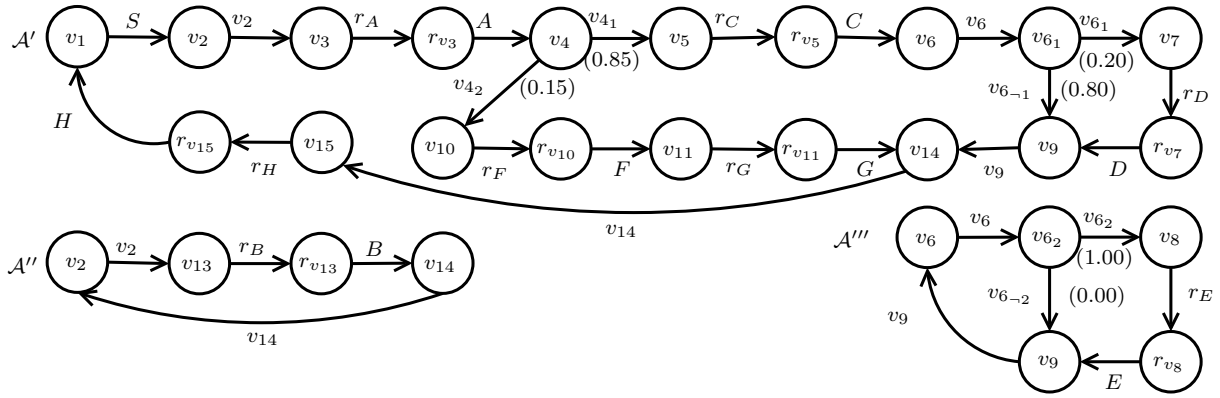


Figura 4.4: *Modelo em SAN resultante da sequência de reduções da Figura 4.3.*

Tabela 4.2: *Resultados obtidos a partir da solução do modelo em SAN da Figura 4.4.*

Instâncias Paralelas	Estados Produto	Estados Alcançáveis	Tempo de Serviço (h)	Utilização Gerente	Utilização Vendedor	Utilização Trabalhador
1	380	85	1,106	0,052	0,269	0,518
2	144.400	5.809	1,599	0,080	0,412	0,793
3	54.872.000	349.013	2,228	0,093	0,476	0,916

O modelo da Figura 4.4 foi solucionado na ferramenta PEPS, considerando de 1 a 3 instâncias paralelas do processo. Para eventos SAN do modelo que representam tarefas, nós usamos taxas definidas sobre os tempos de execução médios indicados na Figura 4.1. Para os demais eventos (usados na sincronização e nas decisões de roteamento), usamos uma taxa constante alta.

Cada empregado do varejo de hardware pode apenas tratar de uma tarefa por vez. Sendo assim, quando uma instância do processo precisa de um empregado para executar uma tarefa e ele está ocupado, então a instância deve esperar por sua disponibilidade. Taxas funcionais foram usadas para evitar a execução de uma tarefa no modelo quando o recurso humano do qual ela depende não está disponível. O Capítulo 5 apresenta mais detalhes sobre a modelagem em SAN do impacto da contenção por recursos sobre um processo de negócio.

A Tabela 4.2 mostra os resultados obtidos na análise do modelo. Nela, temos o tamanho do espaço de estados produto e o tamanho do espaço de estados alcançáveis para cada número de instâncias paralelas que analisamos. Além disso, a tabela mostra o tempo de serviço do processo e a taxa de utilizações dos 3 recursos responsáveis pela execução das tarefas. Com apenas 3 instâncias paralelas do processo, o tempo necessário para o atendimento completo de uma instância duplicou em relação ao tempo de atendimento no caso sequencial. A utilização do trabalhador do depósito quase alcançou o nível de saturação.

Os detalhes sobre como extrair índices de desempenho a partir de modelos em SAN são discutidos no Capítulo 6.

4.5 Conclusão do Capítulo

Neste capítulo, nós apresentamos um algoritmo para conversão automática de uma subclasse de diagramas de processo em BPMN para modelos em SAN.

Primeiro, nós especificamos formalmente as características de um modelo de processo em BPMN bem definido. Depois, nós definimos mapeamentos de objetos de BPMN para modelos em SAN elementares. Finalmente, um conjunto de operações sobre modelos em SAN foi definido para transformar modelos elementares no modelo em SAN final de um processo de negócio.

Esse método de conversão é o primeiro passo de um arcabouço que automatiza a construção de modelos completos de avaliação de desempenho a partir de modelos de processos de negócio. O segundo passo dessa automatização (discutido no Capítulo 5) prevê a inclusão no modelo das informações relacionadas ao gerenciamento de recursos do processo de negócio.

O algoritmo de conversão descrito neste capítulo foi publicado como um relatório técnico do IME-USP [BFV11a], que foi posteriormente estendido e resultou em um artigo aceito para o *8th European Performance Engineering Workshop* – EPEW 2011 [BFV11b].

Capítulo 5

Gerenciamento de Recursos em Modelos de Processos de Negócio

As tarefas de um processo de negócio geralmente dependem de diferentes recursos para serem executadas. O desempenho esperado para um processo de negócio depende de como os recursos são provisionados e usados. O *gerenciamento de recursos* define: (i) quais são os recursos requeridos no processo de negócio, (ii) quantos eles são, (iii) quais são suas capacidades de trabalho, e (iv) como e por quais tarefas eles são acessados.

Neste capítulo, nós definimos uma nova proposta para enriquecer modelos de processos de negócio com informações sobre o seu gerenciamento de recursos, com o objetivo de viabilizar a avaliação de desempenho via modelagem analítica.

A principal contribuição do capítulo é um método para a modelagem estocástica do gerenciamento de recursos em processos de negócio representados em SAN. O método é composto por:

1. uma nova notação que permite enriquecer um modelo de processo de negócio com anotações que definem os requisitos de recursos das tarefas do processo modelado;

2. um método para incluir nos modelos em SAN as informações quantitativas que podem ser extraídas dessas anotações feitas no modelo de processo de negócio.

Esse arcabouço foi implementado como parte da ferramenta computacional `BP2SAN` [BP2].

Na Seção 5.1, nós definimos a notação para considerar o gerenciamento de recursos na modelagem do processo de negócio. Essa notação, embora simples, é rica o suficiente para capturar grande parte dos requisitos de recursos que um processo de negócio pode ter. A Seção 5.2 trata o problema da alocação de recursos sob a perspectiva da modelagem estocástica. Nessa seção, com o auxílio de um exemplo simples, nós definimos um método para modelar o gerenciamento de recursos em processos de negócio usando SAN. A Seção 5.3 descreve a ferramenta computacional que foi desenvolvida para implementar o método proposto.

5.1 Anotando o Gerenciamento de Recursos em Processos de Negócio

No Capítulo 4, nós definimos um algoritmo para converter automaticamente processos de negócio modelados em BPMN para SAN. O modelo em SAN gerado nessa conversão reflete o comportamento de uma instância de um processo de negócio, sem considerar os requisitos de recursos e a degradação que a contenção por recursos causa no desempenho do sistema (especialmente quando várias instâncias de processos são executadas em paralelo). Entretanto, se queremos uma análise de desempenho que realmente se aproxime dos valores esperados para as aplicações do

mundo real, o modelo em SAN deve conter a política de gerenciamento de recursos associada ao sistema em questão. Nas próximas seções, nós propomos uma notação para descrever o gerenciamento de recursos em modelos de processos de negócio. Nossa abordagem pode ser dividida em duas fases:

1. descrição dos recursos disponíveis;
2. descrição dos requisitos de recursos das tarefas.

5.1.1 Descrição dos Recursos Disponíveis

Na sequência, nós formalizamos a noção de recurso usada neste trabalho.

Definição 5.1. *Recurso*
Um recurso R *é uma quádrupla*

$$R = ([\textit{id do recurso}]; [\textit{quantidade}]; [\textit{capacidade de trabalho}]; [\textit{disciplina de acesso}]) \ ,$$

em que:

- [*id do recurso*] *é um identificador unívoco que especifica o tipo do recurso sendo modelado;*
- [*quantidade*] *é um número inteiro positivo que determina quantas unidades do recurso estão disponíveis para acesso;*
- [*capacidade de trabalho*] *é um número decimal positivo que define a quantidade média de trabalho que uma unidade do recurso é capaz de processar por unidade de tempo;*
- [*disciplina de acesso*] *é a estratégia que determina como as instâncias de tarefas são atribuídas ao recurso.*

Existem várias disciplinas de acesso possíveis. Podemos citar como exemplo as comumente encontradas na terminologia da Teoria das Filas [LZGS84]:

- *first-in-first-out* (FIFO) – primeiro a chegar, primeiro a sair, também conhecida por *first-come-first-served* (FCFS) – primeiro a chegar, primeiro a ser atendido pelo recurso;
- *last-in-first-out* (LIFO) – último a chegar, primeiro a sair, também conhecida por *last-come-first-served* (LCFS) – último a chegar, primeiro a ser atendido pelo recurso;
- *escolha aleatória* – uma instância de tarefa será aleatoriamente escolhida entre as demais que estão à espera do acesso ao recurso;
- *tempo compartilhado* – o tempo de processamento do recurso é igualmente dividido entre todas as instâncias paralelas de tarefas que necessitam acessá-lo;
- *sistemas de prioridade* – as instâncias de tarefas são selecionadas para acessar o recurso de acordo com o seu nível de prioridade. Um sistema de prioridade pode ser preemptivo (quando requisições que chegam de instâncias de maior prioridade podem interromper as instâncias que já estão usando o recurso) ou não-preemptivo (quando as instâncias que já estão usando o recurso não podem ser interrompidas).

Neste trabalho, nos restringimos a duas dessas disciplinas de acesso: a escolha aleatória e o tempo compartilhado.

Definição 5.2. *Conjunto de Recursos (CR)*
Um conjunto de recursos *(CR) de um processo de negócio* $\boldsymbol{p}$*, denotado por* $CR(\boldsymbol{p})$*, é o conjunto*

$$CR(\boldsymbol{p}) = \{R \mid R \text{ é um recurso requerido por alguma tarefa do processo de negócio } \boldsymbol{p}\} \ .$$

Exemplo 5.1. *O conjunto de recursos*

$$\begin{aligned} CR(\boldsymbol{p}) = \{ \quad & (\text{“servidor1”};\ 2;\ 5,0;\ \text{“Tempo Compartilhado”}), \\ & (\text{“servidor2”};\ 1;\ 10,0;\ \text{“Tempo Compartilhado”}), \\ & (\text{“impressora”};\ 3;\ 1,5;\ \text{“Escolha Aleatória”}) \quad \} \end{aligned}$$

corresponde a um modelo de processo de negócio $\boldsymbol{p}$ *com três diferentes tipos de recursos, tal que:*

- *existem 2 recursos do tipo “servidor1”, cada um processa 5 unidades de trabalho por unidade de tempo e possui como disciplina de acesso o tempo compartilhado;*
- *existe apenas 1 recurso do tipo “servidor2”, que processa 10 unidades de trabalho por unidade de tempo e possui como disciplina de acesso o tempo compartilhado;*
- *existem 3 recursos do tipo “impressora”, cada um processa 1,5 unidades de trabalho por unidade de tempo e possui como disciplina de acesso a escolha aleatória.*

5.1.2 Descrição dos Requisitos de Recursos

Para sermos capazes de considerar o gerenciamento de recursos nos modelos de avaliação de desempenho automaticamente gerados a partir dos modelos de processo de negócio, nós fazemos duas suposições sobre a utilização de recursos nos processos de negócio considerados neste trabalho:

- os requisitos de recursos são invariantes no tempo e independentes de estado – ou seja, a quantidade de trabalho e/ou o número de recursos requeridos por cada tarefa não varia com o tempo ou com as mudanças de estado do processo;
- quando uma tarefa requer mais de um recurso,ela só será executada quando *todos* os recursos que ela requer estiverem disponíveis. De forma análoga, todos os recursos em uso por uma instância de tarefa em execução serão liberados juntos, no fim da execução da tarefa. De modo contrário, se os recursos pudessem ser alocados e liberados de forma independente, seria possível constatar que a tarefa em questão não possui um comportamento indivisível e poderia ser remodelada como um conjunto de tarefas atômicas paralelas.

Tendo em vista essas suposições, as definições 5.3 e 5.4 formalizam o nosso conceito de requisitos de recursos para tarefas.

Definição 5.3. *Requisito de Recurso Simples (RRS)*
Um requisito de recurso simples *(RRS) de uma tarefa pode ser expresso por um par*

$$([\textit{id do recurso}]; [\textit{quantidade de trabalho}]) \ ,$$

em que:

- [*id do recurso*] *é o identificador do tipo de recurso requerido pela tarefa;*
- [*quantidade de trabalho*] *é um número real positivo que determina quantas unidades de trabalho da tarefa serão processadas pelo recurso.*

Definição 5.4. *Requisitos de Recursos (RR)*

Os requisitos de recursos *(RR) de uma tarefa* $\boldsymbol{a}$*, denotados por* $RR(\boldsymbol{a})$*, podem ser descritos por uma expressão que envolve RRSs e dois operadores lógicos de composição:* $\wedge$ *(**E**) ou* $\vee$ *(**OU**).*

*Em uma expressão de RR, o operador **E** tem uma precedência maior que o **OU**; parênteses podem ser usados para forçar a ordem de avaliação das operações.*

Exemplo 5.2. *Considere uma tarefa* $\boldsymbol{a}$*, com*

$$RR(\boldsymbol{a}) = (\text{“impressora”};\ 4,0) \wedge ((\text{“servidor1”};\ 5,5) \vee (\text{“servidor2”};\ 5,5)).$$

Os requisitos de recursos da tarefa $\boldsymbol{a}$ *determinam que a tarefa requer 1 recurso do tipo “impressora” para processar 4 unidades de trabalho e 1 recurso do tipo “servidor1” ou “servidor2” para processar 5,5 unidades de trabalho.*

Se uma tarefa requer mais de uma unidade de um mesmo tipo de recurso, então podemos expressar esse fato em sua expressão de RR por meio de diferentes RRSs com um mesmo tipo de recurso, como mostrado no Exemplo 5.3.

Exemplo 5.3. *Considere uma tarefa* $\boldsymbol{b}$*, com*

$$RR(\boldsymbol{b}) = (\text{“impressora”};\ 4,0) \wedge (\text{“impressora”};\ 2,0).$$

Os requisitos de recursos de $\boldsymbol{b}$ *determinam que a tarefa requer 2 recursos do tipo “impressora”: um para processar 4,0 unidades de trabalho e o outro para processar 2,0 unidades de trabalho.*

Nós optamos por representar um requisito por x unidades de um recurso R como x RRSs para R (como feito no Exemplo 5.3) porque isso permite que o planejador explicitamente defina como o trabalho total requerido de R será dividido entre as x unidades do recurso.

O conjunto dos requisitos de recursos de todas as tarefas de um modelo de processo de negócio nos dá a política de gerenciamento de recursos desse processo de negócio.

A seguir, faremos duas definições complementares que irão nos ajudar na discussão feita na Seção 5.2.

Definição 5.5. *Requisitos de Recursos (RR) em uma Forma Normal Conjuntiva (FNC) e em uma Forma Normal Disjuntiva (FND)*

Uma expressão de RR está em uma Forma Normal Conjuntiva *(FNC) quando ela é uma conjunção de cláusulas, sendo uma cláusula uma disjunção de RRSs.*

Analogamente, uma expressão RR está um uma Forma Normal Disjuntiva *(FND) quando ela é uma disjunção de cláusulas, sendo uma cláusula uma conjunção de RRSs.*

Uma conjunção é a operação associada ao operador $\wedge$ *(E), enquanto uma disjunção é a operação associada ao operador* $\vee$ *(OU).*

Definição 5.6. *Requisitos Disjuntos de Recursos (RDR)*

Os requisitos disjuntos de recursos *(RDR) de uma tarefa* $\boldsymbol{a}$*, denotados por* $RDR(\boldsymbol{a})$*, são o conjunto de todas as combinações possíveis de recursos que habilitam a execução de uma instância da tarefa* $\boldsymbol{a}$*.*

Todo conjunto de recursos $\mathcal{S} \in RDR(\boldsymbol{a})$ *deve respeitar a seguinte propriedade:* $\mathcal{S}$ *é um conjunto minimal de RRSs da tarefa* $\boldsymbol{a}$ *que, quando satisfeitos, são capazes de habilitar a execução de* $\boldsymbol{a}$*.*

Podemos obter o RDR a partir do RR de uma tarefa por meio dos seguintes passos:

1. se a expressão RR está em uma FNC, então converta-a em uma expressão equivalente em uma FND;
2. cada cláusula conjuntiva de uma expressão RR em uma FND irá originar um novo conjunto do RDR. Os elementos desse conjunto são os RRSs que aparecem na cláusula conjuntiva.

Para ilustrar o conceito de RDR, considere o Exemplo 5.2. A expressão RR do exemplo está em uma FNC. A expressão RR equivalente em uma FND é:

$$\begin{aligned} RR(\boldsymbol{a}) = \quad & (\text{“impressora”};\ 4,0) \wedge (\text{“servidor1”};\ 5,5)) \quad \vee \\ & (\text{“impressora”};\ 4,0) \wedge (\text{“servidor2”};\ 5,5)) \end{aligned}$$

Os requisitos disjuntos de recursos da tarefa $\boldsymbol{a}$ são:

$$\begin{aligned} RDR(\boldsymbol{a}) = \{ \quad & \{(\text{“impressora”};\ 4,0), (\text{“servidor1”};\ 5,5)\}, \\ & \{(\text{“impressora”};\ 4,0), (\text{“servidor2”};\ 5,5)\} \quad \} \end{aligned}$$

5.2 Inclusão do Gerenciamento de Recursos nos Modelos em SAN

Nossa abordagem para modelagem será introduzida com o auxílio de um exemplo simples de *processo de produção* descrito na sequência. Os processos de produção são um subconjunto dos processos de negócio. Embora os processos de produção não contenham as estruturas de controle de fluxo sofisticadas que podem existir em outros tipos de processos de negócio, eles possuem requisitos de recursos complexos e variados, que melhor ilustram a efetividade do nosso método de modelagem.

Exemplo 5.4. *Produção de peças de metal em uma pequena ferramentaria.*

Uma ferramentaria é um lugar onde peças de metal são produzidas por usinagem (i.e., um processo mecânico em que a peça é o resultado de um processo de remoção de material). A ferramentaria do nosso exemplo trabalha com máquinas CNC (*computer numerical controlled*), que fazem uma usinagem de alta precisão controlada por computador, quase sem intervenção humana. As máquinas CNC usinam automaticamente as peças a partir de modelos 3D previamente desenvolvidos por projetistas. A ferramentaria possui duas máquinas CNC idênticas, capazes de usinar apenas peças pequenas, e uma máquina CNC capaz de usinar tanto peças pequenas como peças grandes. Nós chamaremos o primeiro tipo de máquina de `CNC1` e o segundo tipo de `CNC2`. Cada máquina CNC é capaz de produzir somente um item por vez.

O processo de produção começa com a chegada de um pedido de fabricação de uma nova peça. Na sequência, dois projetistas trabalham juntos na confecção do modelo 3D e na programação da

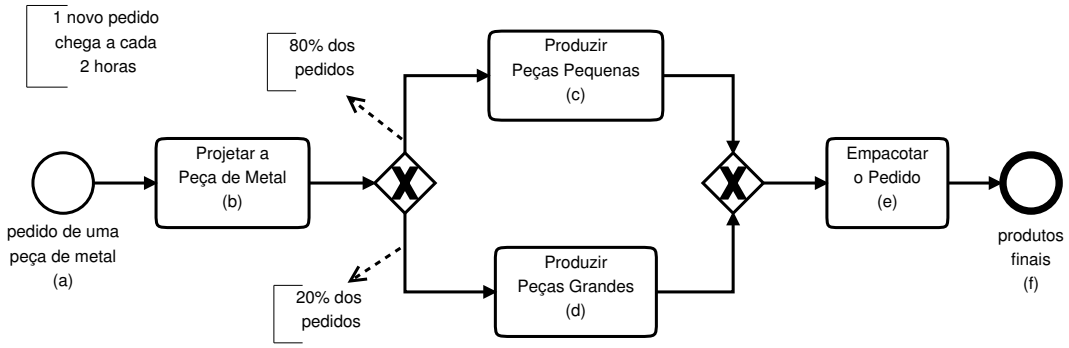


Figura 5.1: *Modelo em BPMN do processo de produção de uma pequena ferramentaria.*

máquina CNC que irá usinar a peça de metal. Um projetista é capaz de trabalhar no modelo de uma só peça por vez e a ferramentaria possui quatro projetistas. Quando o pedido de peça se refere a uma peça grande, então a produção requer o uso da `CNC2`. Caso contrário, a produção da peça pode utilizar a `CNC2` ou uma das unidades de `CNC1`.

Após sair da máquina CNC, a peça é pintada. A ferramentaria possui somente uma máquina de pintura. Dado que cada peça precisa receber várias camadas de tinta, a máquina de pintura precisa esperar que uma camada seque antes de pintar uma nova camada na peça. Sendo assim, em um dado momento do tempo, podemos ter várias peças sendo pintadas na máquina. Após sair da máquina de pintura, a peça é embalada por uma pessoa e o processo de produção é encerrado.

Para que o negócio de produção de peças seja rentável, a ferramentaria só pode aceitar pedidos requisitando uma quantidade de peças dentro de uma faixa pré-definida. A Figura 5.1 mostra o modelo em BPMN desse processo de produção.

Para definir o conjunto de recursos e os requisitos de recursos do processo de negócio do Exemplo 5.4, nós estamos considerando as seguintes informações adicionais sobre a ferramentaria e o seu processo de produção:

- um projetista leva em média 8 horas para fazer sozinho o modelo 3D de uma peça (capacidade de trabalho = 0,125 modelos 3D por hora);
- o tamanho médio de um pedido é 20 unidades no caso de uma peça pequena, e 10 unidades no caso de uma peça grande;
- uma `CNC1` gasta em média 10 minutos para usinar 1 bloco pequeno de metal (capacidade de trabalho = 6 blocos pequenos de metal por hora);
- a `CNC2` gasta em média 30 minutos para usinar 1 bloco grande de metal (capacidade de trabalho = 2 blocos grandes de metal por hora);
- 1 bloco grande de metal equivale a 2 blocos pequenos de metal;
- 1 peça pequena requer em média 3 blocos pequenos de metal (ou 1,5 bloco grande de metal) para ser produzida. Sendo assim, a produção de um pedido de peças pequenas requer em média $3 \times 20 = 60$ blocos pequenos de metal (ou 30 blocos grandes de metal);
- 1 peça grande requer em média 2 blocos grandes de metal para ser produzida. Sendo assim, a produção de um pedido de peças grandes requer em média $2 \times 10 = 20$ blocos grandes de metal;

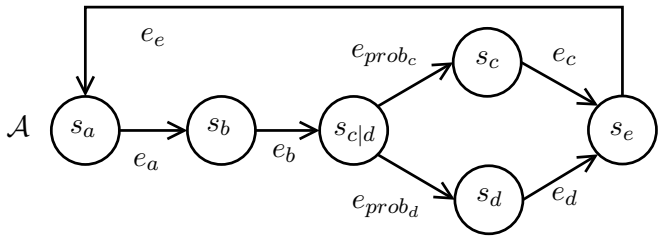


Figura 5.2: *Modelo em SAN da estrutura de controle de fluxo do processo da Figura 5.1.*

- a máquina de pintura leva 6 minutos para pintar 1 bloco pequeno de metal (capacidade de trabalho = 10 blocos pequenos de metal por hora);
- o empacotador leva 1 hora para embalar as peças de um pedido (capacidade de trabalho = 1 pedido por hora).

Considerando as definições 5.2 e 5.4, a declaração do conjunto de recursos e do conjunto de requisitos do processo de produção de peças ($\boldsymbol{p}_{\text{produção}}$) poderia ser a seguinte:

$$
\begin{aligned}
CR(\boldsymbol{p}_{\text{produção}}) = \{ \quad & (\text{“Projetista”};\ 4;\ 0,125;\ \text{“Escolha Aleatória”}),\\
& (\text{“CNC1”};\ 2;\ 6,0;\ \text{“Escolha Aleatória”}),\\
& (\text{“CNC2”};\ 1;\ 2,0;\ \text{“Escolha Aleatória”})\\
& (\text{“Máquina de Pintura”};\ 1;\ 10,0;\ \text{“Tempo Compartilhado”})\\
& (\text{“Empacotador”};\ 1;\ 1,0;\ \text{“Escolha Aleatória”}) \quad \}
\end{aligned}
$$

$$
\begin{aligned}
RR(b) &= (\text{“Projetista”};\ 0,5) \wedge (\text{“Projetista”};\ 0,5)\\
RR(c) &= ((\text{“CNC1”};\ 60,0) \vee (\text{“CNC2”};\ 30,0)) \wedge (\text{“Máquina de Pintura”};\ 60,0)\\
RR(d) &= (\text{“CNC2”};\ 20,0) \wedge (\text{“Máquina de Pintura”};\ 40,0)\\
RR(e) &= (\text{“Empacotador”};\ 1,0)
\end{aligned}
$$

5.2.1 Estrutura de Controle de Fluxo

Um primeiro modelo em SAN do processo da Figura 5.1 é ilustrado pela Figura 5.2. Esse modelo somente expressa a estrutura de controle de fluxo do processo. Nesse modelo, nós temos um estado e um evento SAN associado a cada tarefa do modelo em BPMN. Esse estado SAN indica que a tarefa está habilitada para execução. A execução de tarefa é representada pela transição de estado no autômato, causada pela ocorrência do evento SAN que modela a tarefa. Essa mudança de estado habilita a(s) tarefa(s) seguinte(s) no fluxo de sequência.

O desvio condicional exclusivo do modelo em BPMN da Figura 5.2 foi mapeado como o estado SAN $s_{c|d}$ e seus estados de saída s_c e s_d. Esses dois últimos estados estão conectados ao estado $s_{c|d}$ por meio transições associadas aos eventos e_{prob_c} e e_{prob_d}, respectivamente. A taxa associada aos eventos e_{prob_c} e e_{prob_d} será ponderada pelas probabilidades de ocorrência das tarefas c e d, que, de acordo com a anotação feita no modelo em BPMN, é de 80% e 20%, respectivamente. O Capítulo 4 mostrou mais detalhes sobre como as estruturas de BPMN podem ser mapeadas para modelos em SAN.

Nesse ponto da modelagem, ainda não é possível associar taxas aos eventos SAN que representam as tarefas do processo de negócio, já que essas taxas são impactadas pelos requisitos de recursos das tarefas e eles ainda não foram considerados nesse primeiro modelo em SAN.

Nas próximas seções, explicamos como considerar a declaração de recursos e requisitos do processo de negócio no aperfeiçoamento desse primeiro modelo em SAN criado. A abordagem que introduziremos pode ser sumarizada pelos seguintes passos:

1. enriquecimento do modelo de controle de fluxo com estados, eventos e transições adicionais para representar os requisitos de recursos das tarefas;
2. representação de instâncias paralelas do processo por meio de réplicas de autômatos;
3. inclusão no modelo de autômatos e eventos adicionais para representar os recursos e suas interações com os autômatos de "controle de fluxo";
4. definição de taxas funcionais para expressar: (i) a habilitação das tarefas quando os recursos dos quais elas dependem estiverem disponíveis; (ii) a variação das taxas de execução das tarefas em função da carga de trabalho do sistema.

5.2.2 Associação de Requisitos de Recursos a Tarefas

Um modelo de processo de negócio deveria levar em consideração o fato de que algumas tarefas dependem de recursos para serem executadas. Por essa razão, antes que a execução de uma tarefa seja habilitada, é preciso que os recursos dos quais a tarefa depende sejam alocados. Nós podemos expressar isso introduzindo no modelo em SAN estados e eventos adicionais para expressar:

1. a necessidade de se esperar pela disponibilidade dos recursos;
2. os requisitos disjuntos de recursos, que nos obrigarão a associar mais de um evento em SAN a uma mesma tarefa do modelo em BPMN.

Para cada tarefa do modelo em BPMN que possua requisitos de recursos, nós precisaremos de tantos estados adicionais no modelo em SAN quanto for o número de requisitos disjuntos de recursos (RDR) da tarefa. Como discutido na Seção 5.1.2, o RDR é derivado a partir da expressão RR da tarefa em uma FND.

Na declaração de requisitos do processo de produção de peças, somente a expressão $RR(c)$ está em uma FNC. A expressão equivalente em uma FND é:

$$\begin{aligned} RR(c) = \quad & ((\text{“CNC1”};\ 60) \wedge (\text{“Máquina de Pintura”};\ 60)) \quad \vee \\ & ((\text{“CNC2”};\ 30) \wedge (\text{“Máquina de Pintura”};\ 60)) \end{aligned}$$

Portanto, as RDRs do Exemplo 5.4 são:

$$\begin{aligned} RDR(b) &= \{\{(\text{“Projetista”};\ 0,5), (\text{“Projetista”};\ 0,5)\}\} \\ RDR(c) &= \{\{(\text{“CNC1”};\ 60,0), (\text{“Máquina de Pintura”};\ 60,0)\}, \\ &\qquad \{(\text{“CNC2”};\ 30,0), (\text{“Máquina de Pintura”};\ 60,0)\}\} \\ RDR(d) &= \{\{(\text{“CNC2”};\ 20,0), (\text{“Máquina de Pintura”};\ 40,0)\}\} \\ RDR(e) &= \{\{(\text{“Empacotador”};\ 1,0)\}\} \end{aligned}$$

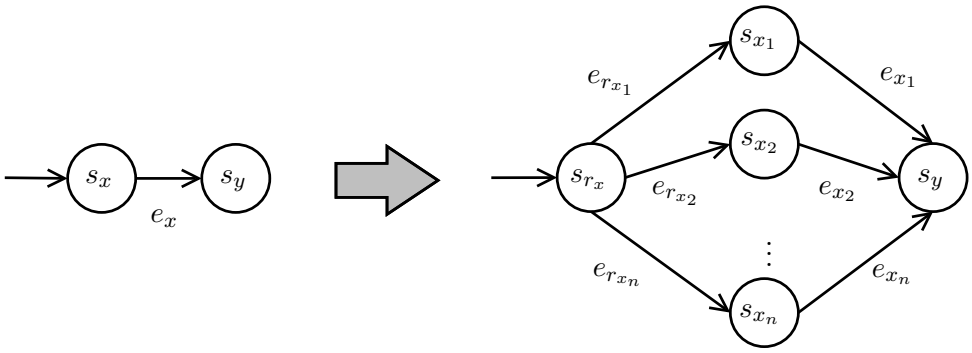


Figura 5.3: *Modificação feita na modelagem de uma tarefa para expressar os seus requisitos de recursos.*

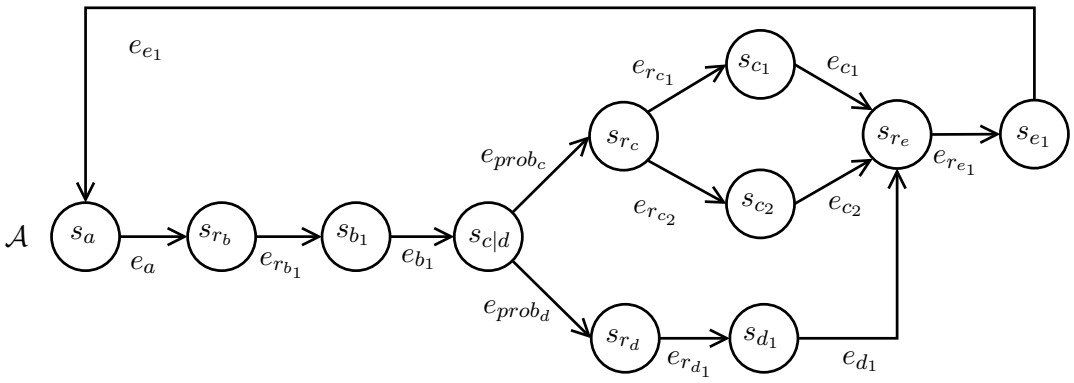
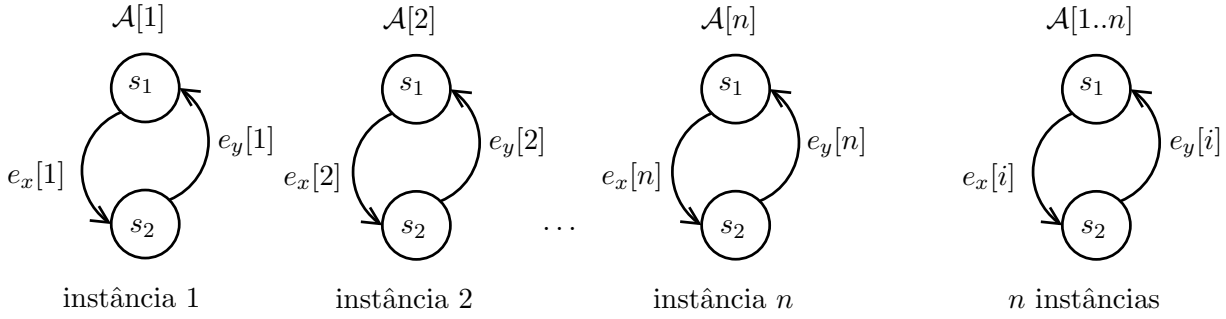


Figura 5.4: *Modelo em SAN da Figura 5.2 enriquecido com estados e eventos adicionais para expressar requisitos de recursos.*

A Figura 5.3 mostra como a modelagem de uma tarefa x precisa ser modificada para expressar os seus requisitos de recursos. Considere que x tem n requisitos disjuntos de recursos (i.e., $RDR(x) = \{\mathcal{S}_{x_1}, \mathcal{S}_{x_2}, \ldots, \mathcal{S}_{x_n}\}$ em que $\mathcal{S}_{x_i}$, com $1 \leq i \leq n$, é um conjunto de RRSs de x). No modelo em SAN parcial mais à esquerda na Figura 5.3, uma tarefa x foi modelada da mesma forma que as tarefas foram modeladas na Figura 5.2. No modelo em SAN parcial mais à direita, nós temos o estado s_{r_x}, representando o estado do processo em que x está esperando pela disponibilidade dos recursos de pelo menos um de seus conjuntos de requisitos disjuntos $\mathcal{S}_{x_i}$, antes de ser habilitado para execução. O processo passará de um estado s_{r_x} para um estado s_{x_i} com a ocorrência de um dos eventos $e_{r_{x_i}}$ (com $1 \leq i \leq n$). A ocorrência de um evento $e_{r_{x_i}}$ indica que todos os RRSs em $\mathcal{S}_{x_i}$ foram satisfeitos e então a tarefa x pode ser executada.

Usando esse mapeamento para tarefas em SAN, nós teremos n eventos (e_{x_i}) para representar a execução da tarefa, cada um com sua respectiva taxa. Isso reflete o fato de que a taxa de execução da tarefa é dada pelos recursos dos quais ele depende; se uma tarefa possui conjuntos alternativos de requisitos de recursos, então ela também possuirá taxas de execução alternativas expressas em função desses conjuntos alternativos.

Modificando a modelagem de tarefas feita na Figura 5.2 de acordo com o exemplo apresentado na Figura 5.3, nós obtemos o modelo em SAN da Figura 5.4. É importante observar que, apesar de todas a mudanças feitas no autômato, a estrutura de controle de fluxo especificada no modelo em SAN da Figura 5.2 foi mantida.



(a) *Um modelo em SAN simples com autômatos replicados* **(b)** *Notação simplificada.*

Figura 5.5: *Emprego da notação simplificada para denotar réplicas de autômatos em SAN.*

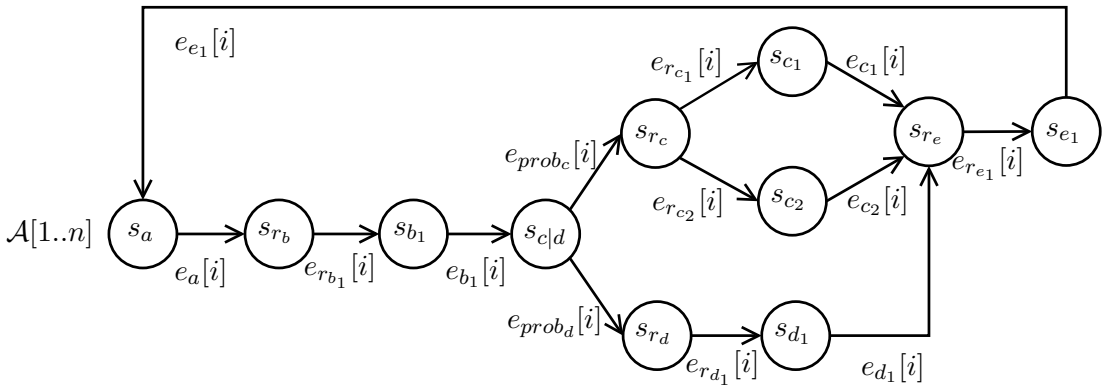


Figura 5.6: *Modelo em SAN da Figura 5.4 com n réplicas (instâncias).*

5.2.3 Modelagem de Instâncias Paralelas por meio da Replicação de Autômatos

O modelo da Figura 5.4 expressa o comportamento de somente uma instância do processo de produção da ferramentaria. Entretanto, novos pedidos de peças podem chegar enquanto um outro pedido já está sendo produzido na ferramentaria (e isso pode causar contenção por recursos).

Para analisar como o desempenho é impactado pela carga de trabalho, é necessário considerarmos o comportamento do sistema quando várias instâncias estão sendo executadas em paralelo. Isso pode ser feito por meio da replicação de autômatos do modelo em SAN. Cada réplica representa uma instância do processo. Dessa forma, o número de réplicas define a carga de trabalho do sistema considerada na análise.

A Figura 5.5a mostra um simples modelo em SAN constituído por n réplicas de um mesmo autômato. Nós usaremos esse modelo para ilustrar a notação simplificada que adotaremos no texto a partir de agora para representar réplicas de autômatos e seus eventos. Uma notação similar foi utilizada por Benoit et al. [BBFP04] na discussão de técnicas de agregação para a redução do espaço de estados de modelos SAN com autômatos replicados.

Um conjunto de n eventos replicados como $\{e_x[1], e_x[2], \ldots, e_x[n]\}$ será denotado por $e_x[1..n]$. De forma análoga, um conjunto de n autômatos replicados $\{\mathcal{A}[1], \mathcal{A}[2], \ldots, \mathcal{A}[n]\}$ será denotado por $\mathcal{A}[1..n]$. A Figura 5.5b mostra essa notação simplificada aplicada ao modelo da Figura 5.5a. Nessa figura, temos os eventos $e_x[i]$ e $e_y[i]$. Nesse contexto, i indica o índice do autômato atual. Portanto, para o autômato $\mathcal{A}[1]$ temos $e_x[1]$ e $e_y[1]$, para $\mathcal{A}[2]$ temos $e_x[2]$ e $e_y[2]$, e assim por diante.

A Figura 5.6 mostra o modelo replicado do processo de fabricação da ferramentaria, usando a notação simplificada para denotar n réplicas (instâncias). Nós não definiremos um número de-

terminado de instâncias para o modelo nesse momento, porque esse é um parâmetro usado para expressar a carga de trabalho do sistema. Como esse parâmetro pode ser facilmente configurado (quando usamos a notação simplificada para especificar as réplicas dos autômatos), nossa sugestão é que o modelo em SAN seja analisado para variados números de instâncias, para avaliar como o desempenho é impactado pelas diferentes cargas de trabalho.

5.2.4 Inclusão de Autômatos Adicionais para Representar Recursos

Na Figura 5.6, temos um modelo em SAN que expressa os requisitos de recursos de um processo de negócio e a possibilidade de instâncias paralelas. No entanto, esse modelo em SAN não considera os recursos e suas respectivas disciplinas de acesso.

Como discutido na Seção 5.1, neste trabalho nós consideramos recursos com dois tipos de disciplinas de acesso: a *escolha aleatória* e o *tempo compartilhado*.

Na disciplina de acesso de escolha aleatória, somente uma instância de tarefa pode acessar o recurso por vez. Quando um recurso em uso é liberado, uma das instâncias à espera do recurso será aleatoriamente selecionada para acessá-lo. Sabemos que, na prática, geralmente há uma estratégia não aleatória (e.g., FIFO, LIFO, etc.) para selecionar uma instância entre as demais que também estão esperando pela disponibilidade do recurso. Contudo, em um dado momento da execução do processo, o número de instâncias à espera de um recurso de acesso exclusivo não mudará com a estratégia de seleção usada. Por essa razão, sob a perspectiva da avaliação de desempenho, a escolha aleatória generaliza as disciplinas de acesso em que o recurso é acessado sob exclusão mútua.

Um recurso com disciplina de acesso de escolha aleatória pode ser diretamente expresso no modelo em SAN por um autômato dedicado. A Figura 5.7 mostra o modelo em SAN da Figura 5.6 enriquecido com os autômatos $\mathcal{A}_{\text{Projetista}}$, $\mathcal{A}_{\text{CNC1}}$, $\mathcal{A}_{\text{CNC2}}$, e $\mathcal{A}_{\text{Empacotador}}$, correspondentes aos recursos com escolha aleatória do processo da ferramentaria.

O autômato de um recurso *res* com m unidades possui $m+1$ estados (s_{res_i}, $0 \leq i \leq m$), cada um expressando uma das possíveis quantidades em uso do recurso em um dado momento do tempo (i.e., s_{res_0} corresponde a 0 unidades do recurso em uso, s_{res_1} corresponde a 1 unidade do recurso em uso, e assim por diante).

De acordo com os seus requisitos de recursos, a tarefa b do processo de produção da ferramentaria (a confecção do modelo 3D da peça de metal) requer 2 unidades do recurso "Projetista" para sua execução. O autômato $\mathcal{A}_{\text{Projetista}}$ passa de um estado s_{des_i} para o estado $s_{\text{des}_{i+2}}$ (em que $0 \leq i \leq 2$) com a ocorrência de um evento do conjunto $e_{r_{b_1}}[1..n]$. Como esses eventos também aparecem nos autômatos $\mathcal{A}[1...n]$, eles são *eventos sincronizantes*. Sendo assim, um autômato $\mathcal{A}[j]$ (em que $1 \leq j \leq n$) somente passará do estado $s_{r_{b_1}}$ para o estado s_{b_1} quando $\mathcal{A}_{\text{Projetista}}$ passar de um estado s_{des_i} para o estado $s_{\text{des}_{i+2}}$ (em que $0 \leq i \leq 2$), indicando que dois recursos disponíveis do tipo "Projetista" foram alocados para a tarefa b da instância j e estão correntemente em uso.

Quando o estado do autômato $\mathcal{A}_{\text{Projetista}}$ é s_{des_4}, todas as unidades do recurso estão correntemente em uso e nenhuma transição do estado $s_{r_{b_1}}$ para o estado s_{b_1} será possível enquanto dois projetistas não forem liberados. Dois recursos são liberados com a ocorrência de um evento do conjunto $e_{b_1}[1..n]$, i.e., quando $\mathcal{A}_{\text{Projetista}}$ passa de um estado s_{des_i} para o estado $s_{\text{des}_{i-2}}$ (em que $2 \leq i \leq 4$) e um autômato $\mathcal{A}[j]$ (em que $1 \leq j \leq n$) passa do estado s_{b_1} para o estado $s_{c|d}$, indicando que a instância j terminou a execução da tarefa b e liberou dois projetistas.

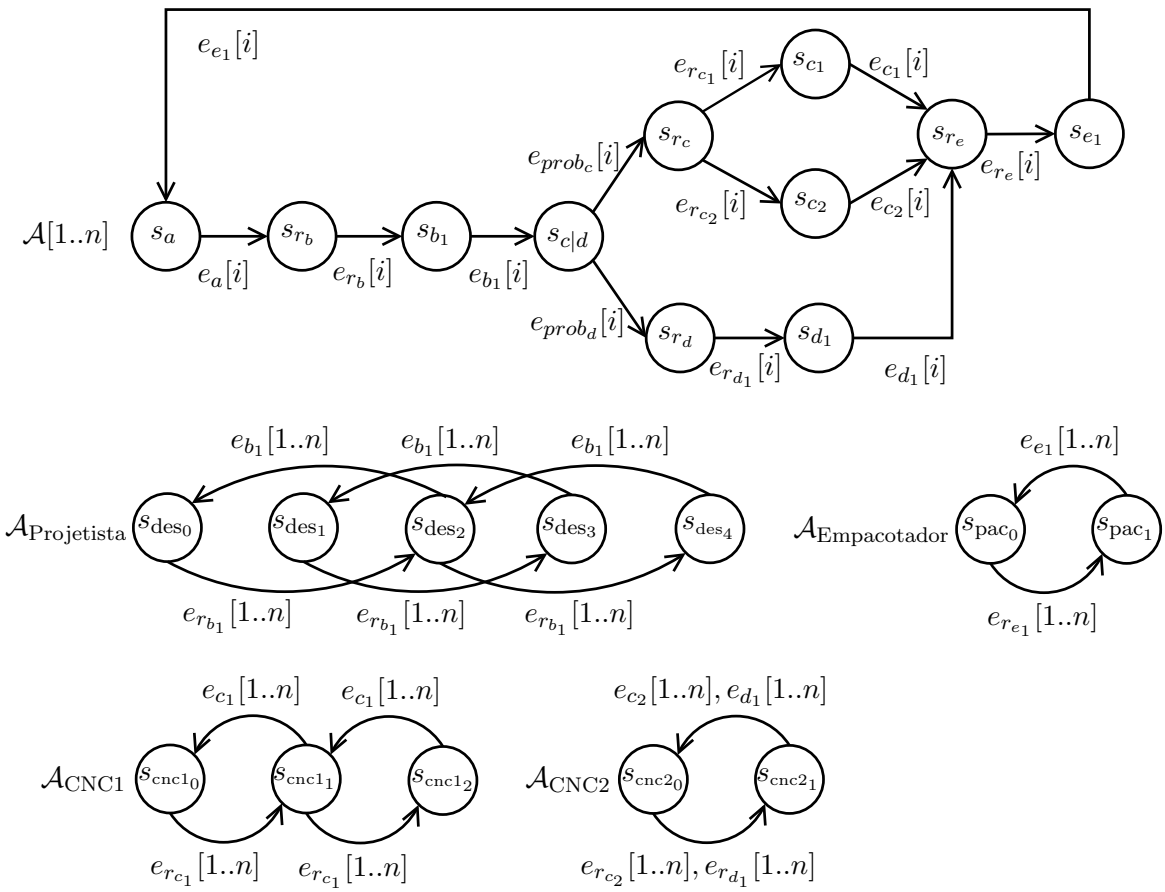


Figura 5.7: *Modelo em SAN completo do processo de produção da ferramentaria.*

O mesmo comportamento sincronizado descrito para $\mathcal{A}_{\text{Projetista}}$ também ocorre entre os autômatos de instâncias $\mathcal{A}[1..n]$ e os outros autômatos de recursos $\mathcal{A}_{\text{CNC1}}$, $\mathcal{A}_{\text{CNC2}}$ e $\mathcal{A}_{\text{Empacotador}}$. Entretanto, nesses outros recursos, somente uma unidade de cada recurso é alocada (ou liberada) pela instância de tarefa por vez.

É importante observar que em $\mathcal{A}_{\text{CNC2}}$, a transição do estado s_{cnc2_0} para o estado s_{cnc2_1} ocorre com a execução de um evento que pode estar em $e_{r_{c_2}}[1..n]$ ou em $e_{r_{d_1}}[1..n]$, para expressar o fato que o recursos `CNC1` aparece nos requisitos tanto da tarefa c quanto da tarefa d. Analogamente, o recurso pode ser liberado por um evento em $e_{c_2}[1..n]$ ou em $e_{d_1}[1..n]$.

Na disciplina de acesso de tempo compartilhado, a capacidade de processamento do recurso será dividida igualitariamente entre as instâncias de tarefas que correntemente requerem o seu uso. Diferentemente dos recursos com escolha aleatória, um recurso com tempo compartilhado não pode ser expresso por autômatos adicionais no modelo em SAN. Eles precisam ser diretamente expressos nas taxas dos eventos que representam a execução das tarefas. Mesmo os recursos com escolha aleatória também podem ser modelados usando somente taxas funcionais, sem a necessidade de autômatos adicionais. Na próxima seção, discutiremos isso com mais detalhes.

5.2.5 Definição das Taxas dos Eventos do Modelo

O último passo envolvido na inclusão de recursos e requisitos no modelo em SAN do Exemplo 5.4 é a definição das taxas associadas aos eventos. Nós faremos isso por meio de uma declaração

textual das constantes e funções, usando uma sintaxe similar à aceita como entrada para a definição de modelos SAN na ferramenta PEPS.

Declaração das Constantes

Antes de definir as taxas dos eventos, definiremos algumas constantes que funcionarão como parâmetros de configuração do nosso modelo em SAN. Elas foram diretamente extraídas do modelo de processo de negócio e de sua declaração de recursos e requisitos. Essas constantes farão parte da definição das taxas. Os valores dos parâmetros podem ser convenientemente mudados, para fornecer análises variadas sem a necessidade de mudanças na estrutura dos autômatos do modelo.

Para cada recurso do processo de negócio, nós definimos uma constante indicando a sua quantidade e outra indicando sua capacidade de trabalho (em unidades de trabalho por unidade de tempo):

qtde_Projetista	=	4;	capTrabalho_Projetista	=	0,125;
qtde_CNC1	=	2;	capTrabalho_CNC1	=	6,0;
qtde_CNC2	=	1;	capTrabalho_CNC2	=	2,0;
qtde_MaquinaPintura	=	1;	capTrabalho_MaquinaPintura	=	10,0;
qtde_Empacotador	=	1;	capTrabalho_Empacotador	=	1,0;

Para cada tarefa no modelo, nós definimos uma constante indicando a quantidade de trabalho que a tarefa requer de cada um dos recursos nos seus conjuntos de requisitos:

trabalhoRequerido_B1_srr1_Projetista	=	0,5;
trabalhoRequerido_B1_srr2_Projetista	=	0,5;
trabalhoRequerido_C1_srr1_CNC1	=	60,0;
trabalhoRequerido_C1_srr2_MaquinaPintura	=	60,0;
trabalhoRequerido_C2_srr1_CNC2	=	30,0;
trabalhoRequerido_C2_srr2_MaquinaPintura	=	60,0;
trabalhoRequerido_D1_srr1_CNC2	=	20,0;
trabalhoRequerido_D1_srr2_MaquinaPintura	=	40,0;
trabalhoRequerido_E1_srr1_Empacotador	=	1,0;

Para cada desvio condicional exclusivo divergente existente no modelo, nós definimos as probabilidades associadas aos seus fluxos de sequência de saída:

prob_C	=	0,8;	prob_D	=	0,2;

Finalmente, nós precisamos definir uma taxa constante, a `tx_instantanea`, para expressar o pequeno tempo de execução associado aos eventos "artificialmente" inseridos no modelo para representar decisões de roteamento do fluxo (como os eventos $e_{prob_c}[1..n]$ e $e_{prob_d}[1..n]$) ou a alocação de recursos (como os eventos $e_{r_{b_1}}[1..n], e_{r_{c_1}}[1..n]$, etc.). Como SAN não possui transições instantâneas, essa taxa constante deve possuir um valor arbitrário maior que as outras taxas no modelo, para que ela não impacte de modo significativo as medidas de desempenho extraídas a partir do modelo em SAN. No nosso exemplo, nós escolhemos o valor 50,0.

tx_instantanea	=	50,0;

É possível dividir os eventos que criamos em nosso modelo em SAN em quatro grupos:

1. eventos que indicam a ocorrência de um evento de início do modelo em BPMN;

2. eventos que indicam um roteamento probabilístico no modelo em BPMN;
3. eventos que indicam que requisitos de recursos de uma tarefa do modelo em BPMN foram satisfeitos;
4. eventos que indicam a execução de uma tarefa do modelo em BPMN.

A taxa associada a eventos do primeiro grupo é um valor constante e não pode ser automaticamente inferido a partir dos dados fornecidos na descrição do processo (i.e., modelo em BPMN + declaração de recursos e requisitos). Um especialista de domínio pode definir o valor mais apropriado ou, como também acontece com o número de instâncias paralelas a ser considerado no modelo, ele pode ser variado na análise, para expressar diferentes cargas de trabalho no sistema. No modelo em SAN do nosso exemplo, os eventos desse tipo são os eventos no conjunto $e_a[1..n]$.

Para os eventos do segundo grupo, precisamos definir taxas capazes de expressar a probabilidade associada à decisão de roteamento à qual se referem. No modelo em SAN do nosso exemplo, temos dois conjuntos de eventos desse tipo: $e_{prob_c}[1..n]$ e $e_{prob_d}[1..n]$.

Em um modelo em SAN, uma condição de disputa governa o comportamento do sistema quando ele tem que evoluir a partir de um estado no qual mais de uma transição está habilitada. Nesse caso, quanto mais "rápida" a transição for (i.e., quanto maior for a taxa da transição), mais frequentemente ela ganhará a disputa (i.e., ela ocorrerá). Por essa razão, nós podemos usar taxas ponderadas para expressar roteamentos probabilísticos.

Para expressar as probabilidades associadas a cada fluxo de saída de um desvio de decisão, usaremos uma taxa constante (para expressar o custo do roteamento) multiplicada pela probabilidade do fluxo de saída.

`tx_prob_C`	=	`tx_instantanea` × `prob_C`;
`tx_prob_D`	=	`tx_instantanea` × `prob_D`;

Como mencionado na Seção 2.3.3, o formalismo SAN possui o conceito de *taxas funcionais*, taxas de eventos que são dadas por funções sobre o espaço de estados do sistema. A seguir, usaremos essas funções para definir as taxas dos eventos dos grupos 3 e 4.

Declaração das Funções Auxiliares

Usando a função nb (descrita na Seção 2.3.3), nós definimos para cada recurso do modelo uma função que nos dá o número de unidades correntemente em uso:

`f_qtdeUsada_Projetista`	=	$2 \times nb(\mathcal{A}[1..n], s_{b_1})$;
`f_qtdeUsada_CNC1`	=	$nb(\mathcal{A}[1..n], s_{c_1})$;
`f_qtdeUsada_CNC2`	=	$nb(\mathcal{A}[1..n], s_{c_2}) + nb(\mathcal{A}[1..n], s_{d_1})$;
`f_qtdeUsada_MaquinaPintura`	=	$nb(\mathcal{A}[1..n], s_{c_1}) + nb(\mathcal{A}[1..n], s_{c_2}) + nb(\mathcal{A}[1..n], s_{d_1})$;
`f_qtdeUsada_Empacotador`	=	$nb(\mathcal{A}[1..n], s_{e_1})$;

Pelas discussões feitas nas seções 5.2.3 e 5.2.4, sabemos que quando um autômato em $\mathcal{A}[1..n]$ está no estado s_{b_1}, por exemplo, então o primeiro (e único) conjunto disjunto de recursos requeridos para que a tarefa b seja executada já foi alocado e está em uso (nesse caso, dois projetistas). Por essa razão, o número total de projetistas correntemente em uso (`f_qtdeUsada_Projetista`) é dado pelo número de instâncias no estado s_{b_1} ($nb(\mathcal{A}[1..n], s_{b_1})$) multiplicado por dois.

Para os demais recursos, o raciocínio é análogo. É importante notar que os recursos CNC2 e a Máquina de Pintura são requisitos de mais de uma tarefa. Consequentemente, o número de unidades correntemente em uso é dado pela soma das contagens (*nb*) sobre diferentes estados.

Agora, definiremos as funções booleanas que devolvem a disponibilidade dos recursos. Para cada recurso, criaremos uma função para cada diferente quantidade que uma tarefa do modelo pode requerer do recurso. Usando as funções de disponibilidade, nós definimos funções para indicar quando uma tarefa está ou não habilitada para execução em função da disponibilidade dos recursos dos quais ela depende.

Em um modelo sem autômatos adicionais para recursos:

```
f_estaDisponivel_Projetista_2   = ((qtde_Projetista -
                                    f_qtdeUsada_Projetista) ≥ 2);
f_estaDisponivel_CNC1_1         = ((qtde_CNC1 - f_qtdeUsada_CNC1) ≥ 1);
f_estaDisponivel_CNC2_1         = ((qtde_CNC2 - f_qtdeUsada_CNC2) ≥ 1);
f_estaDisponivel_Empacotador_1  = ((qtde_Empacotador -
                                    f_qtdeUsada_Empacotador) ≥ 1);
f_existe_MaquinaPintura_1       = (qtde_MaquinaPintura ≥ 1);

f_estaHabilitado_r_B1           = f_estaDisponivel_Projetista_2;
f_estaHabilitado_r_C1           = (f_estaDisponivel_CNC1_1  E
                                    f_existe_MaquinaPintura_1);
f_estaHabilitado_r_C2           = (f_estaDisponivel_CNC2_1  E
                                    f_existe_MaquinaPintura_1);
f_estaHabilitado_r_D1           = (f_estaDisponivel_CNC2_1  E
                                    f_existe_MaquinaPintura_1);
f_estaHabilitado_r_E1           = f_estaDisponivel_Empacotador_1;
```

Em um modelo com autômatos adicionais para recursos:

```
f_existe_MaquinaPintura_1       = (qtde_MaquinaPintura ≥ 1);

f_estaHabilitado_r_C1           = f_existe_MaquinaPintura_1;
f_estaHabilitado_r_C2           = f_existe_MaquinaPintura_1;
f_estaHabilitado_r_D1           = f_existe_MaquinaPintura_1;
```

Como discutido na Seção 5.2.4, os recursos de tempo compartilhado são expressos no modelo por meio de taxas funcionais (veremos isso nas próximas definições de funções). No entanto, os recursos cuja disciplina de acesso é a escolha aleatória podem ser expressos no modelo em SAN de duas maneiras: (i) usando somente taxas funcionais, ou (ii) usando autômatos adicionais (como feito na Figura 5.7) combinados com taxas funcionais. Quando usamos autômatos adicionais para modelar recursos com escolha aleatória, a alocação e a liberação de um recurso por uma instância de tarefa é controlada pelos eventos sincronizantes.

Por essa razão, só precisamos definir funções de disponibilidade para recursos de escolha aleatória em um modelo em SAN que não inclui autômato adicionais para representar esses recursos. Nesse caso, as funções de disponibilidade devem devolver *verdadeiro* (valor 1) se a quantidade indicada do recurso está disponível para uso, e *falso* (valor 0) no caso contrário[1].

[1] Uma função em SAN sempre devolve um valor numérico. Mesmo quando o valor de retorno da função é a avaliação de uma expressão booleana, o valor de retorno será 1 para indicar *verdadeiro*, e 0 para indicar *falso*

No caso de um recurso de tempo compartilhado, nós só precisamos saber se o recurso existe na quantidade requerida pelas tarefas do modelo (já que uma unidade do recurso nunca estará "indisponível" para uma tarefa).

Para cada recurso cuja disciplina de acesso é tempo compartilhado, nós definimos uma função que devolve a capacidade de trabalho do recurso na perspectiva de uma tarefa que está utilizando-o. Essa função nos dá a capacidade de trabalho nominal do recurso dividida pelo número de instâncias de tarefas correntemente compartilhando o acesso ao recurso:

```
f_capTrabalhoCompartilhada_MaquinaPintura  =  capTrabalho_MaquinaPintura /
                                                 f_qtdeUsada_MaquinaPintura;
```

Na sequência, temos as definições das taxas de processamento dos recursos para cada requisito de recurso no qual ele aparece. Essa taxa é o inverso do tempo necessário para o recurso processar o trabalho demandado pela tarefa.

```
f_tx_B1_srr1_Projetista      =  capTrabalho_Projetista /
                                      trabalhoRequerido_B1_srr1_Projetista;
f_tx_B1_srr2_Projetista      =  capTrabalho_Projetista /
                                      trabalhoRequerido_B1_srr2_Projetista;
f_tx_C1_srr1_CNC1            =  capTrabalho_CNC1 / trabalhoRequerido_C1_srr1_CNC1;
f_tx_C1_srr2_MaquinaPintura  =  f_capTrabalhoCompartilhada_MaquinaPintura /
                                      trabalhoRequerido_C1_srr2_MaquinaPintura;
f_tx_C2_srr1_CNC2            =  capTrabalho_CNC2 / trabalhoRequerido_C2_srr1_CNC2;
f_tx_C2_srr2_MaquinaPintura  =  f_capTrabalhoCompartilhada_MaquinaPintura /
                                      trabalhoRequerido_C2_srr2_MaquinaPintura;
f_tx_D1_srr1_CNC2            =  capTrabalho_CNC2 / trabalhoRequerido_D1_srr1_CNC2;
f_tx_D1_srr2_MaquinaPintura  =  f_capTrabalhoCompartilhada_MaquinaPintura /
                                      trabalhoRequerido_D1_srr2_MaquinaPintura;
f_tx_E1_srr1_Empacotador     =  capTrabalho_Empacotador /
                                      trabalhoRequerido_E1_srr1_Empacotador;
```

Declaração das Taxas Funcionais

Depois das declarações feitas na Seção 5.2.5, nós estamos aptos a definir as taxas finais associadas aos eventos do nosso modelo em SAN.

Os eventos do grupo 3 (que indicam quando os requisitos de recursos das tarefas foram satisfeitos) estão associados às seguintes taxas funcionais:

Em um modelo sem autômatos adicionais para recursos:

```
f_tx_r_B1  =  f_estaHabilitado_r_B1 × tx_instantanea;
f_tx_r_C1  =  f_estaHabilitado_r_C_1 × tx_instantanea;
f_tx_r_C2  =  f_estaHabilitado_r_C_2 × tx_instantanea;
f_tx_r_D1  =  f_estaHabilitado_r_D1 × tx_instantanea;
f_tx_r_E1  =  f_estaHabilitado_r_E1 × tx_instantanea;
```

Em um modelo com autômatos adicionais para recursos:

```
f_tx_r_B1  =  tx_instantanea;
f_tx_r_C1  =  tx_instantanea;
f_tx_r_C2  =  tx_instantanea;
f_tx_r_D1  =  tx_instantanea;
f_tx_r_E1  =  tx_instantanea;
```

Quando `f_estaHabilitado_r_B` devolve 0 (indicando que os recursos requeridos por b não estão disponíveis), o valor de retorno de `f_tx_r_B1` também será 0, indicando que o evento $e_{r_{b_1}}$ (de alocação de recursos para a tarefa b) não pode ocorrer no estado corrente do sistema. Em outro caso, a taxa de $e_{r_{b_1}}$ será maior que 0, indicando que a alocação dos recursos para b pode ocorrer no estado corrente do sistema. A mesma interpretação é válida para as outras funções do bloco acima.

As taxas funcionais associadas aos eventos do grupo 4, que representam a execução de uma tarefa do processo de negócio, são dadas na sequência. No caso de tarefas que dependem de somente um recurso, a taxa de execução será determinada pelo tempo requerido pelo recurso para processar a quantidade de trabalho demandada. Para tarefas que dependem de mais de um recurso, a taxa de execução será determinada pelo recurso mais lento (i.e., a menor taxa entre as taxas dos recursos dos quais a tarefa depende).

```
f_tx_B1  =  min(f_tx_B1_srr1_Projetista, f_tx_B1_srr2_Projetista);
f_tx_C1  =  min(f_tx_C1_srr1_CNC1, f_tx_C1_srr2_MaquinaPintura);
f_tx_C2  =  min(f_tx_C2_srr1_CNC2, f_tx_C2_srr2_MaquinaPintura);
f_tx_D1  =  min(f_tx_D1_srr1_CNC2, f_tx_D1_srr2_MaquinaPintura);
f_tx_E1  =  f_tx_E1_srr1_Empacotador;
```

A Tabela 5.1 mostra as taxas associadas a cada um dos eventos do modelo em SAN na Figura 5.7. É importante relembrar que não é possível definir de forma automática uma taxa para os eventos de início ($e_a[1..n]$), já que ela não é dada em termos do gerenciamento de recursos do processo de negócio. O mesmo acontecerá com as tarefas de um modelo de processo de negócio que não possuem requisitos de recursos. Nesses casos, a taxa dos eventos pode ser especificada pelos especialistas de domínio.

5.3 Implementação

O método introduzido neste capítulo para a descrição do gerenciamento de recursos em modelos de processos de negócio e o seu mapeamento para modelos em SAN foi implementado como parte da ferramenta computacional `BP2SAN`. O núcleo da ferramenta, como dito anteriormente, é o algoritmo de conversão definido no Capítulo 4.

A ferramenta `BP2SAN` é capaz de converter diagramas de processo em BPMN bem definidos para modelos em SAN que refletem a estrutura de controle de fluxo de tarefas modelada inicialmente para o processo de negócio. Além disso, quando anotações relativas ao gerenciamento de recursos associado ao modelo em BPMN são fornecidas como entrada à ferramenta, os modelos em SAN automaticamente gerados condicionam a execução das tarefas à disponibilidade dos recursos dos quais elas dependem.

As entradas para a `BP2SAN` são modelos em BPMN textualmente descritos usando a linguagem DOT e (opcionalmente) arquivos descrevendo os recursos e requisitos associados aos modelos de entrada. As declarações de recursos e requisitos são feitas em um formato análogo ao especificado nas definições 5.2 e 5.4.

Tabela 5.1: *Os eventos do modelo em SAN da Figura 5.7 e suas respectivas taxas.*

Evento(s)	Taxa	Função no Modelo de Processo de Negócio
$e_{r_{b_1}}[1..n]$	f_tx_r_B1	alocação de recursos
$e_{b_1}[1..n]$	f_tx_B1	execução de tarefas
$e_{prob_c}[1..n]$	tx_prob_C	roteamento probabilístico
$e_{prob_d}[1..n]$	tx_prob_D	roteamento probabilístico
$e_{r_{c_1}}[1..n]$	f_tx_r_C1	alocação de recursos
$e_{c_1}[1..n]$	f_tx_C1	execução de tarefas
$e_{r_{c_2}}[1..n]$	f_tx_r_C2	alocação de recursos
$e_{c_2}[1..n]$	f_tx_C2	execução de tarefas
$e_{r_{d_1}}[1..n]$	f_tx_r_D1	alocação de recursos
$e_{d_1}[1..n]$	f_tx_D1	execução de tarefas
$e_{r_{e_1}}[1..n]$	f_tx_r_E1	alocação de recursos
$e_{e_1}[1..n]$	f_tx_E1	execução de tarefas

Os modelos em SAN gerados pela BP2SAN são textualmente expressos na sintaxe aceita pela ferramenta PEPS.

A ferramenta BP2SAN e mais detalhes sobre o seu uso estão publicamente disponíveis em seu sítio *web* [BP2].

5.4 Conclusão do Capítulo

Neste capítulo, nós definimos uma nova notação que possibilita enriquecer um modelo de processo de negócio com informações sobre o seu gerenciamento de recursos. Na sequência, amparados por um estudo de caso, definimos um método para incorporar as informações do gerenciamento de recursos a um modelo em SAN de um processo de negócio. Com o método apresentado neste capítulo, nós concluímos a definição do arcabouço proposto para a geração de modelos completos de avaliação de desempenho a partir de modelos de processos de negócio.

Com esse arcabouço, o gerenciamento de recursos é primeiramente modelado em um alto nível de abstração. O modelo estocástico que expressa como a execução das tarefas é impactada pelo recursos dos quais elas dependem é automaticamente inferido pelo nosso método a partir dos modelos anotados de processo de negócio. Apenas algumas poucas habilidades estatísticas são demandadas dos projetistas de processos, para definir a capacidade de trabalho dos recursos no modelo na forma de simples taxas médias de execução.

A implementação desse arcabouço de modelagem deu origem à uma ferramenta computacional, a BP2SAN. Com o auxílio de um solucionador para SAN (como a ferramenta PEPS), variados índices de desempenho podem ser extraídos dos modelos de processo de negócio em SAN gerados pela ferramenta. O Capítulo 6 discute como esses índices podem ser obtidos.

Os resultados descritos neste capítulo geraram um artigo que foi submetido ao *International Journal of Innovative Computing, Information and Control* (IJICIC), para a edição especial intitulada *Intelligent and Innovative Computing in Business Process Management* [BFV11c].

Capítulo 6

Extração de Índices de Desempenho para Processos de Negócio

Nós podemos extrair índices de desempenho a partir de um modelo SAN por meio da definição de *funções de integração* sobre o espaço de estados do sistema. A avaliação dessas funções sobre a distribuição de probabilidades estacionária do modelo nos dá as medidas de desempenho. Por exemplo, o rendimento do recurso `CNC1` do Exemplo 5.4 é dado por:

$$\texttt{capTrabalho\_CNC1} \times \sum_{i \in S} \pi(i) \ ,$$

em que S é o conjunto de todos os estados globais do modelo em SAN da Figura 5.7 nos quais o recurso `CNC1` está em uso (i.e., os estados globais em que $\texttt{f\_qtdeUsada\_CNC1} > 0$) e $\pi(i)$ é a probabilidade do estado global i na distribuição estacionária do modelo.

Na ferramenta PEPS, podemos especificar as funções que definem os índices de desempenho juntamente com a definição textual do modelo SAN. Após solucionar numericamente o modelo, a ferramenta devolve como resultado os valores integrados dessas funções sobre as probabilidades dos estados do sistema no regime estacionário. A seguir, usando como base o exemplo do processo de produção da ferramentaria, definido na Seção 5.2, especificamos algumas funções para extrair índices de desempenho na forma como elas são definidas na ferramenta PEPS. Algumas das funções que usaremos a seguir foram previamente definidas na Seção 5.2.5.

Os modelos SAN automaticamente gerados pela ferramenta `BP2SAN` incluem funções de integração para a obtenção dos índices de desempenho mais comumente usados e que independem de especificidades do processo de negócio fornecido como entrada para a ferramenta.

6.1 Índices de Desempenho Frequentemente Empregados

6.1.1 Número de Unidades em Uso de um Recurso

Integrando a função `f_qtdeUsada_Projetista` sobre as probabilidades em regime estacionário nós obteremos o *número médio* de projetistas ocupados na ferramentaria no regime estacionário do sistema.

6.1.2 Taxa de Utilização de um Recurso

A seguir, definimos uma função que devolve *verdadeiro* (1) quando o recurso `CNC2` está em uso:

$$\texttt{f\_estaEmUso\_CNC2} = (\texttt{f\_qtdeUsada\_CNC2} > 0) \ .$$

A integração da função `f_estaEmUso_CNC2` sobre as probabilidades em regime estacionário nos dá a *taxa de utilização* de `CNC2` (i.e., a porcentagem de tempo em que o recurso está em uso).

Para recursos com mais de uma unidade, podemos extrair uma taxa de utilização para cada número possível de unidades do recurso que podem estar simultaneamente em uso no sistema. Por exemplo, considere a seguinte função que devolve *verdadeiro* (1) quando as quatro unidades do recurso `Projetista` está simultaneamente em uso:

$$\texttt{f\_estaEmUso\_Projetista\_4} = (\texttt{f\_qtdeUsada\_Projetista} == 4)\ .$$

A integração da função `f_estaEmUso_Projetista_4` sobre as probabilidades em regime estacionário nos dá a porcentagem de tempo em que os quatro projetistas estão ocupados ao mesmo tempo.

6.1.3 Tamanho da Fila de Espera de um Recurso

Para definir um índice de desempenho que nos dê o número médio de pedidos esperando pela disponibilidade de projetistas, podemos integrar $nb(A[1..n], s_{r_b})$ sobre as probabilidades em regime estacionário. O estado s_{r_b} indica que a instância do processo está esperando pela disponibilidade dos recursos requeridos para a execução da tarefa b.

6.1.4 Rendimento de um Recurso

Vamos definir uma função que devolve a taxa (capacidade de trabalho) do recurso `CNC1` multiplicada pelo seu número de unidades em uso:

$$\texttt{f\_taxaUtil\_CNC1} = (\texttt{f\_qtdeUsada\_CNC1} > 0) \times \texttt{capTrabalho\_CNC1}\ .$$

A integração da função `f_taxaUtil_CNC1` sobre as probabilidades em regime estacionário nos dá o *rendimento* das máquinas CNC1, ou seja, o número efetivo de blocos de metal que as duas unidades do recurso juntas processam por hora.

Quando o recurso possui apenas uma unidade, o seu rendimento é equivalente a sua capacidade de trabalho multiplicada pela sua taxa de utilização.

6.1.5 Rendimento de uma Tarefa

O rendimento de uma tarefa pode ser definido de forma análoga ao rendimento de um recurso. Entretanto, as tarefas não possuem uma taxa constante associada a elas. Por essa razão, precisamos considerar a taxa funcional da tarefa.

Para obter o rendimento da tarefa e, definiremos uma função auxiliar que devolve a taxa de e multiplicada pelo número de instâncias de e habilitadas para execução no estado:

$$\texttt{f\_taxaUtil\_E} = nb(\mathcal{A}[1..n], s_{e_1}) \times \texttt{f\_tx\_E1}\ .$$

Da integração de `f_taxaUtil_E` sobre as probabilidades em regime estacionário, nós obtemos o rendimento da tarefa b, ou seja, o número efetivo de pedidos empacotados por hora na ferramentaria.

6.1.6 Tempo de Serviço (ou Tempo de Atendimento)

O *tempo de serviço* (ou *tempo de atendimento*) do processo de negócio é o tempo médio necessário para a execução completa de uma instância do processo. Para calculá-lo, precisamos da probabilidade do estado inicial de um autômato de instância. Considere a função `f_probEstadoInicial` definida como:

$$\texttt{f\_probEstadoInicial} = (nb(\{\mathcal{A}[1]\}, s_a) > 0) \ .$$

Da integração de `f_probEstadoInicial` sobre as probabilidades em regime estacionário, nós obtemos a probabilidade $\pi(s_0)$ de um autômato de instância estar no seu estado inicial (que no caso do nosso exemplo é o estado s_a).

Sabendo que

$$\pi(s_0) = \frac{\text{tempo entre 2 chegadas de pedidos}}{\text{tempo entre 2 chegadas de pedidos } + \text{ tempo de serviço}}$$

e que o tempo entre duas chegadas de pedidos na ferramentaria é $\frac{1}{\text{taxa de } e_a}$, o tempo de serviço é dado pela fórmula:

$$\text{tempo de serviço} = \frac{1 - \pi(s_0)}{\text{taxa de } e_a \times \pi(s_0)} \ .$$

6.1.7 Rendimento do Processo

O *rendimento do processo* é o número médio de instâncias do processo de negócio que são completadas por unidade de tempo.

Para calculá-lo, precisamos da probabilidade $\pi_i(s_0)$ do estado inicial de cada um dos autômatos de instância $\mathcal{A}_i$ do modelo. O cálculo dessa probabilidade pode ser feito de forma semelhante ao definido na Seção 6.1.6.

Sabendo que

$$\pi_i(s_0) = \frac{\text{tempo entre 2 chegadas de pedidos}}{\text{tempo entre 2 chegadas de pedidos } + \text{ tempo de serviço em } i}$$

e que o tempo de atendimento é o inverso do rendimento, o rendimento do autômato da instância i é dado pela fórmula:

$$\text{rendimento}_i = \frac{\text{taxa de } e_a \times \pi_i(s_0)}{1 - \pi_i(s_0)} \ .$$

O rendimento total do processo é dado pela soma dos rendimentos de cada autômato de instância:

$$\text{rendimento}_{\text{total}} = \sum_{\mathcal{A}_i \in \mathcal{A}[1..N]} \frac{\text{taxa de } e_a \times \pi_i(s_0)}{1 - \pi_i(s_0)} \ .$$

6.2 Alguns Resultados para o Processo de Produção da Ferramentaria

Usando a `BP2SAN`, geramos um modelo em SAN a partir do modelo em BPMN anotado da Figura 5.1. Em seguida, usando a ferramenta PEPS, nós analisamos alguns índices de desempenho do processo. Calculamos a taxa de utilização dos recursos no modelo e o tempo de atendimento do

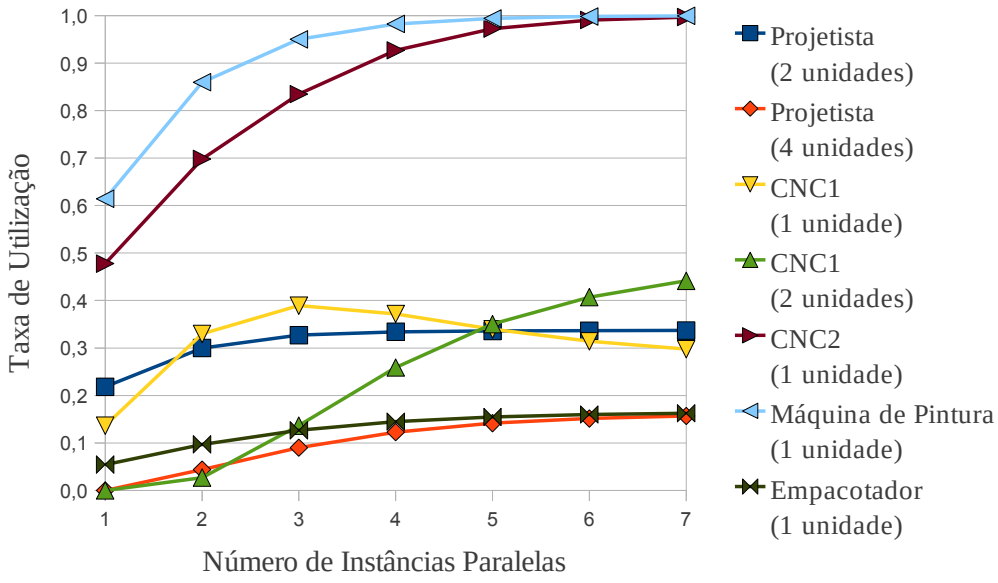


Figura 6.1: *Variação da taxa de utilização dos recursos em função da carga de trabalho no processo de produção da ferramentaria.*

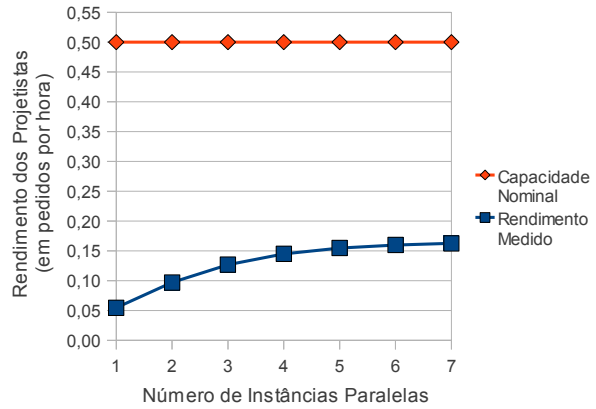
processo de negócio. Nós usamos como taxa de chegada o valor 0,5 pedidos por hora, ou seja, a cada duas horas (em média) um novo pedido chegar na ferramentaria. Nós analisamos o modelo para um número de instâncias em paralelo variando de um até sete, para expressar diferentes cargas de trabalho.

O gráfico na Figura 6.1 mostra a variação da taxa de utilização dos recursos em função do número de instâncias paralelas. No gráfico, podemos observar que as máquinas CNC e a máquina de pintura se tornam os recursos mais sobrecarregados com o aumento da carga de trabalho. Com cinco instâncias paralelas, os recursos `CNC2` e Máquina de Pintura quase alcançam o nível de saturação, enquanto os projetistas e os empacotadores estão ociosos em mais da metade do tempo. Esses resultados sugerem que a ferramentaria deveria considerar a possibilidade de aquisição de novas máquinas para poder aumentar a utilização dos seus recursos humanos, ou reduzir o número de projetistas.

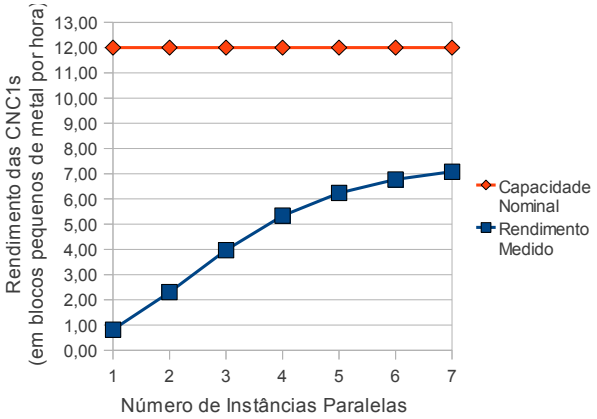
No gráfico da Figura 6.1, podemos observar também que a curva associada a utilização de uma só unidade do recurso `CNC1` começa a decrescer a partir de três instâncias paralelas no processo. Essa diminuição é explicada pelo fato de que, com o aumento da carga de trabalho no sistema, a porcentagem de tempo em que temos somente uma unidade do recurso em uso diminui, enquanto a proporção de tempo em que as duas unidades de `CNC1` estão em uso aumenta de forma rápida, como mostra a curva correspondente no gráfico.

A Figura 6.2 mostra os gráficos de rendimento de cada um dos tipos de recursos modelados no processo. Nesses gráficos, a linha vermelha mostra a capacidade nominal do recurso, que é dada por sua capacidade de trabalho multiplicada por seu número de unidades. A linha azul mostra o rendimento conjunto de todas as unidades do recurso, em função das diferentes cargas de trabalho analisadas.

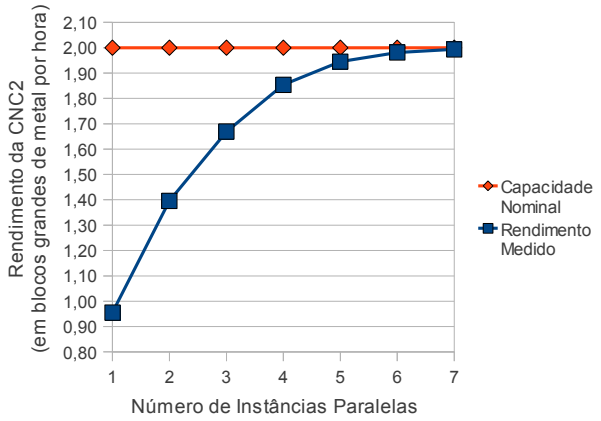
A linha cheia no gráfico da Figura 6.3 mostra como o tempo de atendimento do processo de negócio é impactado pelo número de pedidos sendo tratados paralelamente na ferramentaria (de acordo com os resultados obtidos a partir do modelo em SAN e o auxílio da ferramenta PEPS). Por tempo de atendimento, entende-se o tempo necessário para tratar completamente um pedido



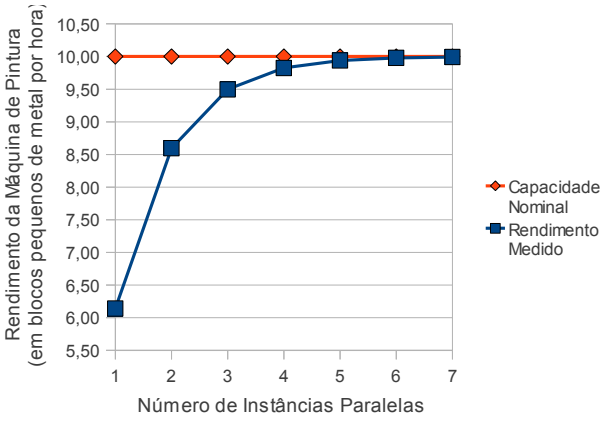
(a) *Rendimento dos Projetistas.*



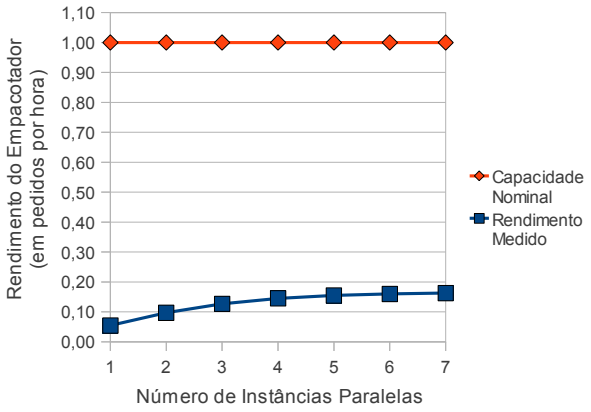
(b) *Rendimento das* CNC1s.



(c) *Rendimento da* CNC2.



(d) *Rendimento da Máquina de Pintura.*



(e) *Rendimento do Empacotador.*

Figura 6.2: *Variação do rendimento dos recursos em função da carga de trabalho no processo de produção da ferramentaria.*

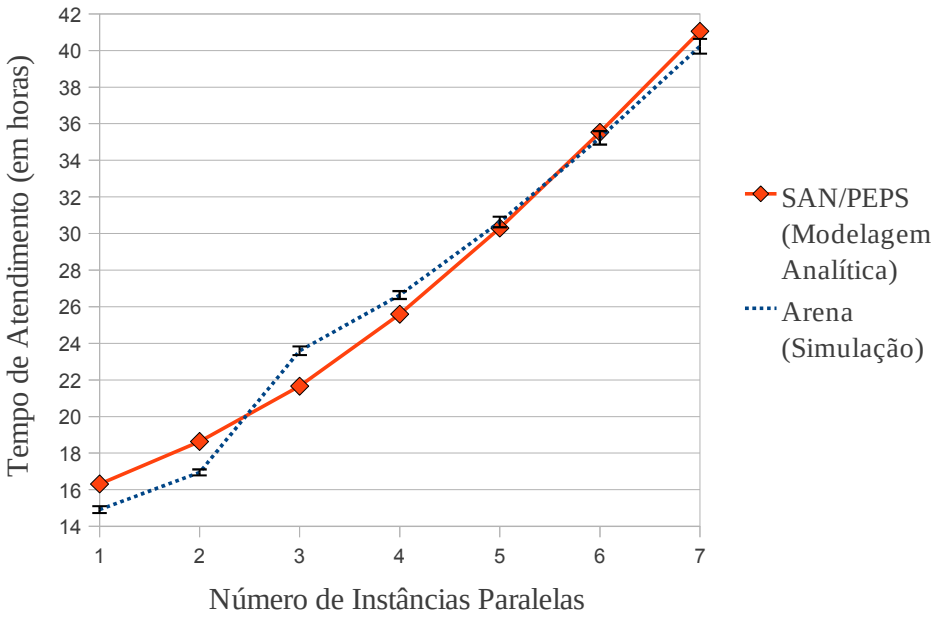


Figura 6.3: *Variação do tempo de atendimento em função da carga de trabalho no processo de fabricação na ferramentaria.*

Tabela 6.1: *Tamanho do espaço de estados do modelo em SAN e a ordem de magnitude do tempo de computação da solução do modelo em função da carga de trabalho no processo da ferramentaria.*

Instâncias Paralelas	Tam. do Espaço de Estados Produto	Tam. do Espaço de Estados Alcançável	Tempo de Computação da Solução do Modelo
1	11	11	≈ 0 segundos
2	121	111	10^{-2} segundos
3	1.331	1.056	10^{-1} segundos
4	14.641	9.612	10^{1} segundos
5	161.051	84.456	10^{2} segundos
6	1.771.561	720.576	10^{3} segundos
7	19.487.171	5.995.296	10^{4} segundos

de peças metálicas na ferramentaria. No gráfico, é possível observar que o tempo para processar um pedido por vez é aproximadamente 16 horas. O tempo de atendimento ultrapassa 40 horas quando temos sete pedidos sendo tratados paralelamente na ferramentaria.

A Tabela 6.1 mostra o tamanho do espaço de estados do modelo em SAN da Figura 5.7 para cada número de instâncias paralelas analisado em nossos experimentos. A tabela também mostra a ordem de magnitude do tempo de computação necessário para a ferramenta PEPS solucionar numericamente cada modelo em um computador com processador Intel® Xeon® com 2,6 GHz e 32 GB de memória RAM.

É importante observarmos que, com sete instâncias paralelas, o modelo do processo de produção da ferramentaria ultrapassou 5 milhões de estados alcançáveis. Modelos como esse, com grandes espaços de estados, podem ser considerados intratáveis em outras técnicas de modelagem analítica. Entretanto, devido às características do formalismo mencionadas na Seção 2.3.3, com SAN podemos analisar sistemas de grande escala mais frequentemente que com outros formalismos Markovianos.

Validação dos Resultados Obtidos

Para validar os resultados obtidos através do nosso arcabouço de análise de desempenho, nós comparamos esses resultados com os gerados por outro método de análise bem estabelecido: a simulação. Nós projetamos e analisamos o processo de produção da ferramentaria em um simulador de eventos discretos de propósito geral chamado Arena [AM07, Are].

O simulador Arena é uma ferramenta flexível, que nos permite especificar os modelos de simulação de forma gráfica, para os mais variados domínios de aplicação. Ele possui conectores básicos para a especificação de fluxos de tarefas (como sequência, escolha e paralelismo) e nos permite associar às tarefas tempos de execução (que podem ser constantes ou podem seguir uma distribuição especificada pelo usuário). Além disso, ele possui mecanismos para a definição de recursos de acesso exclusivo e para a associação desses recursos às tarefas do modelo.

No processo de produção da ferramentaria, nós temos tanto recursos de acesso exclusivo quanto recursos de tempo compartilhado. Além disso, o tempo de execução das tarefas é definido em função da capacidade de trabalho dos recursos alocados para a execução. Os recursos de tempo compartilhado e as taxas funcionais não são facilmente modelados nos simuladores.

Para podermos modelar o processo da ferramentaria no Arena, fizemos a seguinte simplificação no modelo para transformar a taxa do recurso de tempo compartilhado em uma taxa constante:

- se o número n de instâncias paralelas no modelo é menor que três, então

  $$\texttt{f\_capTrabalhoCompartilhada\_MaquinaPintura} = \texttt{capTrabalho\_MaquinaPintura}/n$$

- senão

  $$\texttt{f\_capTrabalhoCompartilhada\_MaquinaPintura} = \texttt{capTrabalho\_MaquinaPintura}/3$$

  já que o grau máximo de paralelismo no uso da máquina de pintura é três (ou seja, o número total de máquinas CNC).

Para todo número i de instâncias paralelas do processo tal que $1 \leq i \leq 7$, nós realizamos 30 simulações do modelo (com os mesmos valores de parâmetros usados no modelo SAN), cada uma simulando o funcionamento do processo no período de um ano. A linha tracejada no gráfico da Figura 6.3 mostra o tempo médio de serviço obtido a partir dos experimentos de simulação. Cada medida está apresentada com um intervalo de confiança de 95%.

6.3 Conclusão do Capítulo

Neste capítulo, nós mostramos como alguns índices de desempenho comumente usados podem ser extraídos de modelos de processos de negócio em SAN. Para ilustrar esses índices, nós mostramos os resultados obtidos na análise de desempenho do processo de negócio definido no Exemplo 5.4, considerando diferentes números de instâncias paralelas.

Nós comparamos os tempos de serviço obtidos a partir do modelo em SAN com os obtidos por meio de um outro método de avaliação de desempenho, a simulação. Pelo que podemos observar no gráfico da Figura 6.3, os resultados obtidos por meio dos dois métodos foram satisfatoriamente similares. A diferença entre as linhas é devida às características inerentes a cada método, como acurácia, expressividade, etc.

A Seção 6.3.1 expõe alguns dos contrastes entre a análise de desempenho de processos de negócio via simulação e o arcabouço para modelagem analítica que propusemos neste trabalho.

6.3.1 Modelagem Analítica × Simulação

Neste trabalho, nós nos dedicamos à análise preditiva de desempenho de processos de negócio e, para isso, nos baseamos em modelos analíticos. Contudo, esse tipo de análise preditiva também pode ser feita por meio de simulação.

Existem várias ferramentas computacionais de simulação especificamente criadas para o domínio da GPN (Jansen-vullers e Netjes [JVN06] estudaram algumas dessas ferramentas). Mas grande parte dos simuladores de processos de negócio não foram desenvolvidos objetivando a análise de desempenho, mas sim a *análise de custo* dos processos. Nesses casos, os simuladores possibilitam a modelagem de recursos cuja propriedade principal é o custo por unidade de tempo. As tarefas são associadas a recursos e, consequentemente, o custo de uma tarefa pode ser dado em função da quantidade de tempo que ela requer dos seus recursos. Embora os conceitos de *custo* e *tempo de execução* possam parecer relacionados, é importante não confundi-los já que eles são calculados de maneira distinta. Por exemplo, como Magnani e Montesi enfatizaram em seu trabalho [MM07], o custo de um conjunto de tarefas é dado pela soma dos custos individuais de cada tarefa, mesmo quando elas são executadas em paralelo; o mesmo não ocorre com o tempo de execução.

Embora algumas das ferramentas de simulação para o domínio da GPN permitam a extração de índices de desempenho a partir dos modelos, elas não possuem as funcionalidades necessárias para a especificação de um modelo preciso. De forma geral, elas apenas nos permitem associar tempos de execução às tarefas e probabilidades aos fluxos de saída de um ponto de decisão no processo. Existem ferramentas, como a WoPeD (*Workflow Petri Net Designer*) [EF08], que possibilitam a modelagem de recursos de acesso exclusivo, mas não permitem a definição de requisitos sofisticados de recursos (como, por exemplo, conjuntos alternativos de dependências de recursos). Além disso, esses tipos de modelo de simulação não nos permitem capturar a degradação de desempenho causado pela contenção por recursos; o tempo de execução das tarefas não é dado em função dos seus requisitos de recursos, nem em função da carga do sistema.

Existem também vários simuladores de eventos discretos de propósito geral que podem ser aplicados na análise de desempenho de processos de negócio. Essas ferramentas possuem linguagens de modelagem mais expressivas que as encontradas nas ferramentas de GPN. Por outro lado, a construção de um modelo de processo de negócio complexo usando essas linguagens não é uma tarefa fácil, pois ela demanda bastante tempo e um conhecimento aprofundado do funcionamento do simulador. Todas as estruturas sofisticadas de ramificação e junção de fluxos existentes nos modelos de processo de negócio precisam ser denotadas em termos dos comandos disponíveis na ferramenta de simulação. Alguns simuladores poderosos (como o Simpy [MV03] e o Arena [AM07]) possuem mecanismos para facilitar a modelagem de pontos de congestão, em que as tarefas precisam ser enfileiradas para a obtenção do acesso ao recurso. Entretanto, esses mecanismos não se aplicam aos recursos de tempo compartilhado, que são difíceis de serem implementados em simulações.

A precisão das medidas de desempenho obtidas na análise também é uma questão importante na comparação entre simulação e modelagem analítica, independentemente do domínio de aplicação. Se estamos somente interessados na estimação de índices de desempenho dados por valores médios, a simulação pode prover resultados satisfatórios. Entretanto, quando precisamos estimar a proba-

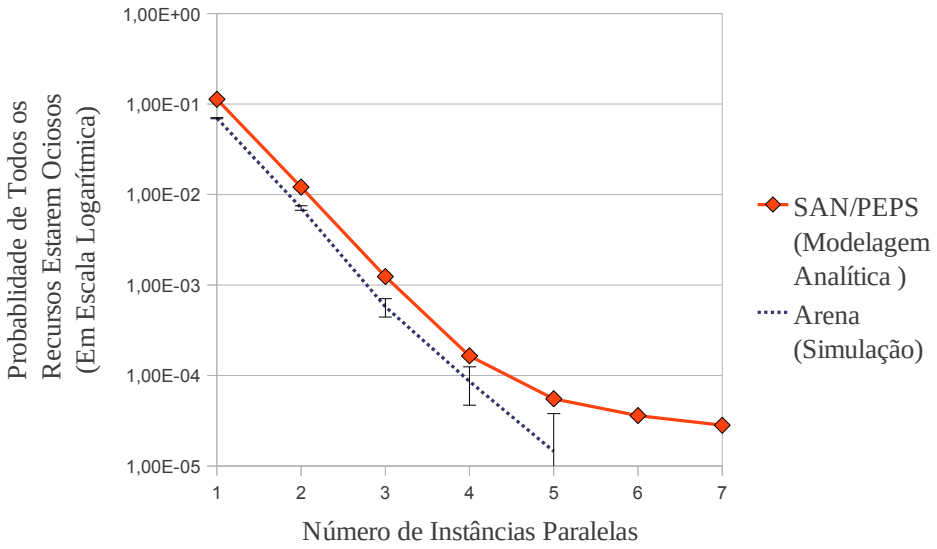


Figura 6.4: *Variação da probabilidade de todos os recursos estarem ociosos em função da carga de trabalho no processo da ferramentaria.*

bilidade da ocorrência de eventos raros (ou seja, uma probabilidade muito pequena), a simulação não é a melhor abordagem para a análise, uma vez que o número de experimentos necessários para prover probabilidades estimadas dentro de um intervalo de confiança aceitável pode ser proibitivo. Em ambos os casos, a modelagem analítica nos dá resultados mais precisos.

Para ilustrar o problema da diferença entre a precisão das duas técnicas, calculamos a probabilidade de todos os recursos estarem ociosos, ao mesmo tempo, no processo da ferramentaria. Essa probabilidade decresce em função do aumento da carga de trabalho no processo. O gráfico na Figura 6.4 mostra os valores de probabilidades obtidos por meio da modelagem analítica com SAN/PEPS e o da simulação na ferramenta Arena (com um intervalo de confiança de 95%).

Nesse gráfico, podemos observar que a precisão dos valores obtidos via simulação diminui com o aumento da carga de trabalho (para um número fixo de simulações). Além disso, no exemplo analisado, a ferramenta Arena não pôde calcular com precisão suficiente a probabilidade do evento raro para seis e sete instâncias paralelas no processo de negócio. Em todas as simulações realizadas para essas quantidades de instâncias, os valores obtidos para a probabilidade foram zero.

Capítulo 7

Conclusões

7.1 Considerações Finais

Neste trabalho de doutorado, nós definimos um novo arcabouço para a geração automática de modelos de avaliação de desempenho a partir de modelos de processos de negócio e informações sobre o seu gerenciamento de recursos.

Esse arcabouço é composto por uma nova notação, que possibilita a especificação dos recursos e da forma como eles são usados no processo de negócio, e por um método para gerar modelos em SAN a partir de processos de negócio modelados usando BPMN e a nova notação.

Nesse arcabouço de modelagem, é possível capturar três aspectos de um processo de negócio que estão intrinsecamente relacionados ao seu desempenho:

1. a estrutura de controle de fluxo de tarefas;
2. os requisitos de recursos das tarefas;
3. o comportamento do processo quando múltiplas instâncias estão executando em paralelo, disputando o acesso a um conjunto finito de recursos.

Diferentemente de outros trabalhos relacionados, no nosso arcabouço de modelagem os tempos de execução de uma tarefa são determinados em função da quantidade de trabalho que elas representam e da capacidade de trabalho dos recursos dos quais elas dependem para serem executadas. O modelo em SAN gerado a partir desse arcabouço é capaz de refletir como o desempenho do processo é afetado pela contenção por recursos conforme a carga de trabalho do sistema aumenta.

Com a notação que definimos, é possível especificar tarefas com requisitos de recursos sofisticados. Uma tarefa pode depender tanto de uma conjunção de recursos (i.e., um conjunto de recursos) quanto de uma disjunção de conjunções de recursos (i.e., conjuntos alternativos de recursos). O nosso método prevê a modelagem de recursos de acesso exclusivo e de recursos de tempo compartilhado.

As características do formalismo SAN contribuíram para a simplicidade da nossa abordagem. Usando SAN, fomos capazes de modelar recursos e requisitos de maneira direta. As instâncias paralelas dos processos de negócio puderam ser facilmente consideradas na modelagem com o uso do conceito de réplicas de autômatos. As taxas funcionais nos permitiram modelar os relacionamentos que existem entre as taxas das tarefas e os requisitos de recursos das mesmas.

Os especialistas de negócio não precisam ter conhecimentos específicos de modelagem estocástica para especificar o gerenciamento de recursos de um processo de negócio usando o notação que propusemos. A descrição do gerenciamento de recursos é feita em um alto nível de abstração. Todo

o arcabouço estocástico necessário para modelar a aleatoriedade e a variabilidade dos processo de negócio e seus requisitos de recursos é automaticamente inferido pelo nosso método.

O nosso arcabouço de modelagem foi implementado na BP2SAN, uma ferramenta computacional capaz de converter automaticamente diagramas de processos em BPMN anotados para modelos em SAN. Usando um software solucionador para SAN (como a ferramenta PEPS), os especialistas de negócio podem predizer vários índices de desempenho a partir dos modelos em SAN automaticamente gerados. Entre os índices de desempenho que podem ser obtidos, podemos destacar o rendimento médio das tarefas, a taxa de utilização e o rendimento médio dos recursos, e o tempo de serviço e o rendimento médios do processo de negócio.

Os parâmetros que descrevem o comportamento quantificável do sistema podem ser facilmente ajustados para expressar diferentes cargas de trabalho ou diferentes capacidades de recursos. Analisando como os índices de desempenho são afetados pela variação dos parâmetros, somos capazes de identificar ineficiências do processo ou dependências inesperadas entre as tarefas do modelo e, com isso, fazer um melhor provisionamento de recursos para o processo de negócio.

7.2 Sugestões para Pesquisas Futuras

Para a continuidade do trabalho desenvolvido neste projeto de doutorado, nós temos quatro sugestões de pesquisa. As duas primeiras podem ser desenvolvidas a curto prazo, enquanto as demais requerem mais esforços de pesquisa. As propostas são descritas a seguir.

Adaptação do Algoritmo de Conversão para uma Subclasse Maior de BPMN

Para o algoritmo definido no Capítulo 4, consideramos como entradas válidas somente modelos de processo de negócio construídos com os objetos de BPMN listados na Tabela 2.1 e descritos na Seção 2.1.1.

Apesar de já lidarmos com uma subclasse de BPMN bastante expressiva, seria interessante estender o algoritmo de conversão para diagramas de processo contendo outros tipos de objetos ou até mesmo para os demais tipos de modelos existentes em BPMN, como os diagramas de colaboração e os diagramas de coreografia.

Por exemplo, nos diagramas de processo e de colaboração de BPMN, existem vários objetos que podem ser ativados com a ocorrência de um evento externo ao processo de negócio (como as tarefas de recebimento de mensagens e os desvios de fluxo baseados em eventos). Se pudermos associar uma taxa de ocorrência a esses eventos externos, então também poderemos modelar esses objetos em SAN. Quando esses eventos externos são gerados por processos de negócio cujo modelo é conhecido, podemos, em alguns casos, estimar a taxa de geração dos eventos por meio da análise de desempenho desse modelo.

Decomposição Automática de Processos de Negócio

O tamanho do espaço de estados pode restringir o uso de modelos Markovianos na avaliação de desempenho de sistemas de grande escala. Existem técnicas de decomposição para a análise de cadeias de Markov que ajudam a reduzir essa restrição. A ideia principal dessas técnicas é decompor a cadeia de Markov em subcadeias que podem ser resolvidas separadamente e cujos resultados podem ser agregados para fornecer os resultados da cadeia inicial [Ros96].

Motivados por essa noção de decomposição, nossa proposta é a de decompor um modelo de processo de negócio antes mesmo que ele seja convertido em um modelo estocástico para avaliação de desempenho. Com isso, a semântica das construções usadas para modelar o controle de fluxo dos processos de negócio poderia auxiliar no desenvolvimento de um novo método de decomposição específico para o domínio de aplicação.

Entretanto, é importante ressaltarmos que nem todos os modelos de processo de negócio especificam comportamentos que podem ser completamente decompostos em termos de outros mais elementares. Por essa razão, para que o algoritmo de decomposição possa ser definido, nós precisamos identificar e caracterizar a classe de modelos de processos de negócio que são completamente decomponíveis. Alguns trabalhos [Mur89, OFMP09] sugerem que técnicas de redução de grafos podem ajudar nessa tarefa.

Mineração de Modelos Estocásticos a partir de *Logs* de Processos de Negócio

A *mineração de processos* é um assunto relacionado à subárea de análise de processos de negócio. O objetivo da mineração de processos é extrair um modelo explícito de processo a partir de *logs* contendo sequências de eventos gerados pela execução das instâncias dos processos [vdA11]. Um dos interesses da mineração de processos é a identificação das relações causais existentes entre os eventos registrados no *log*.

As técnicas de mineração de processos existentes atualmente ainda possuem limitações. A maioria é capaz de inferir modelos determinísticos, somente para uma classe restrita de processos (os que não contêm estruturas complexas de controle de fluxo – como ciclos, dependências de longa distância, etc.). Além disso, algumas técnicas são suscetíveis aos ruídos que podem existir nos *logs* utilizados como entrada.

Uma abordagem promissora no contexto da mineração de processos é a estatística, que leva em consideração dados como a frequência de ocorrência das sequências de eventos, o intervalo de tempo entre as ocorrências, etc. Acreditamos que é possível usar uma abordagem estatística para inferir modelos estocásticos a partir do *log* de execução de processos. O uso de modelos estocásticos pode permitir o desenvolvimento de uma técnica de mineração menos suscetível a ruídos, além de prover uma representação que melhor captura a variabilidade dos processos de negócio.

Aplicação da Teoria de Campo Médio para a Análise de Desempenho de Processos de Negócio

Como vimos nos modelos estocásticos de processos de negócio apresentados neste texto, quando consideramos múltiplas instâncias de processos executando paralelamente, o espaço de estados do modelo cresce de maneira muito rápida. Entretanto, seria interessante podermos analisar o comportamento de um processo de negócio considerando milhares de instâncias paralelas.

Le Boudec et al. [BB08] estudaram o comportamento de sistemas compostos por objetos idênticos que interagem entre si, em que a evolução de cada objeto é dada por uma cadeia de Markov com um número de estados finito. Os autores mostraram que, quando o sistema é composto por um grande número de objetos, a medida de ocupação do sistema converge para um sistema dinâmico determinístico (o *campo médio*) cuja dimensão é o número de estados de um objeto individual. Esse resultado vem da aplicação da *Teoria de Campo Médio* (da Mecânica Es-

tatística), cuja ideia principal é substituir todas as interações entre os objetos de um sistema por uma interação média ou efetiva e, dessa forma, passar de uma descrição microscópica para uma descrição macroscópica do sistema.

Outros trabalhos mais recentes [BGT08, BCFH11, GGB10] exploram o uso desse resultado na análise de desempenho de sistemas de grande escala. Esses trabalhos assinalam a possibilidade da aplicação da técnica na análise de processos de negócio.

Apêndice A

Cadeias de Markov em Tempo Contínuo

Um processo de Markov nos permite modelar a incerteza em sistemas do mundo real que evoluem dinamicamente no tempo. Os conceitos básicos de um processo de Markov são os *estados* e as *transições entre estados*. Os estados são discretos e contáveis. As transições entre estados são modeladas por um processo estocástico de *tempo discreto* ou de *tempo contínuo*, definido por uma distribuição *geométrica* ou *exponencial*, respectivamente.

Em um processo de Markov, o comportamento probabilístico futuro do processo depende somente do estado atual do processo e não é influenciado pelo seu histórico. Essa propriedade é chamada de *Propriedade de Markov*.

A representação gráfica de uma cadeia de Markov é dada por uma máquina de estados finitos, em que cada estado da cadeia de Markov corresponde a um estado da máquina de estados e cada transição está associada a uma *probabilidade* (no caso de um modelo em tempo discreto) ou a uma *taxa* (no caso de um modelo em tempo contínuo).

As únicas distribuições de probabilidade que não possuem "memória" são as geométricas (no caso discreto) e as exponenciais (no caso contínuo). Na teoria das probabilidades, uma variável aleatória X é dita sem memória se

$$P\{X > (t+s) \mid X > s\} = P\{X > t\} \qquad \text{para } s, t \geq 0 \ .$$

Se X é, por exemplo, uma variável que indica o intervalo de tempo entre duas ocorrências de um dado evento, podemos dizer que a probabilidade desse intervalo de tempo ser maior que $t+s$ dado que ao menos s unidades de tempo já transcorreram desde a última ocorrência do evento equivale à probabilidade inicial desse intervalo de tempo ser maior que t.

Todas as definições encontradas neste apêndice foram extraídas de [Ros96, Tij03, DR77].

A.1 Conceitos Básicos

Em Cadeias de Markov em Tempo Contínuo (CMTCs), os tempos entre sucessivas transições de estados são exponencialmente distribuídos, enquanto a sucessão dos estados é descrita por um modelo em Cadeia de Markov de Tempo Discreto (CMTD).

Já que a distribuição exponencial é uma distribuição *sem memória*, o resultado futuro do processo estocástico depende somente do estado presente e independe dos estados anteriores do processo (propriedade de Markov).

Definição A.1. *Cadeia de Markov de Tempo Contínuo (CMTC)*

Um processo estocástico em tempo contínuo $\{X_t, t \geq 0\}$ *com um espaço discreto* I *de estados é uma Cadeia de Markov de Tempo Contínuo se, para todo* $s, t \geq 0$ *e inteiros não negativos* $i, j, x(u), 0 \leq u \leq s$,

$$P\{X_{(t+s)} = j \mid X_s = i,\ X_u = x(u),\ 0 \leq u < s\} = P\{X_{(t+s)} = j \mid X_s = i\}$$

Em uma CMTC a distribuição condicional do estado futuro do processo no tempo $t+s$*, dado o seu estado corrente no tempo* s *e todos os estados anteriores, depende somente do estado corrente e independe do passado.*

Quando $P\{X_{(t+s)} = j \mid X_s = i\}$ *independe de* s*, então dizemos que a CMTC é* homogênea.

No restante deste capítulo, sempre que nos referirmos a uma CMTC, estaremos nos referindo a uma CMTC homogênea.

Denotamos por $p_{ij}(t)$ a probabilidade condicional de que uma CMTC passe do estado i para o estado j depois de um tempo t, ou seja:

$$p_{ij}(t) = P\{X_{(t+s)} = j \mid X_s = i\}, \qquad i, j \in I.$$

Uma CMTC também pode ser definida como um processo Markoviano de salto, descrito pelas seguintes regras:

1. quando o sistema salta para o estado i, então ele fica em i por um período de tempo exponencialmente distribuído com média $1/\nu_i$, independentemente de como o sistema alcançou o estado i e de quanto tempo ele levou para chegar lá;

2. quando o sistema deixa o estado i, ele salta para o estado j $(j \neq i)$ com uma probabilidade p_{ij}, independentemente da duração da sua permanência em i. Além disso, $\sum_{j \neq i} p_{ij} = 1, \forall i \in I$.

Uma CMTC é dita *regular* se, em qualquer intervalo de tempo finito, o número de transições é finito com probabilidade 1. No restante desta seção, toda a vez em que nos referirmos a uma CMTC, estaremos nos referindo a uma cadeia regular.

A.2 Taxas de Transição

Seja q_{ij} definida por

$$q_{ij} = \nu_i p_{ij}, \quad \forall i, j \in I \text{ e com } i \neq j.$$

Dado que ν_i é a taxa com a qual o processo sai do estado i e p_{ij} é a probabilidade dele ir para o estado j, segue que q_{ij} é a *taxa de transição* de i para j (ou seja, a taxa com que o processo faz uma transição para o estado j quando ele está no estado i).

As taxas de transição q_{ij} determinam as taxas de tempo de permanência ν_i e as probabilidades de transição p_{ij} por meio das seguintes fórmulas:

$$\nu_i = \sum_{j \neq i} q_{ij} \qquad \text{e} \qquad p_{ij} = q_{ij}/\nu_i \qquad \forall i, j \in I.$$

A representação matemática de uma CMTC pode ser dada por sua *matriz de taxas de transição*

Q. Cada elemento Q_{ij}, $\forall\ i, j \in I$, é dado por

$$Q_{ij} = \begin{cases} q_{ij} = \nu_i p_{ij} & \text{se } i \neq j \\ -\sum_{k \neq i} Q_{ik} & \text{se } i = j \end{cases} .$$

A matriz Q também é chamada de *gerador infinitesimal* da CMTC.

A.3 Classificação de Estados

Definição A.2. *Tempo de Retorno a um Estado*

Seja $\{X_t, t \geq 0\}$ *uma CMTC. Vamos denotar por* T_1 *o instante da primeira transição. Para cada* $j \in I$*, definimos:*

$$T_j = min\{t \geq T_1 \mid X_t = j\}$$

Se o estado inicial i *for diferente de* j*,* T_j *representa o instante da primeira visita a* j*. Caso o estado inicial* i *seja igual a* j*, então* T_j *representa o instante do primeiro retorno ao estado* j*.*

Se admitirmos que o estado inicial i não é absorvente, a probabilidade de que a cadeia visite eventualmente o estado j dado que ela partiu de i, denotada por F_{ij}, pode ser descrita como:

$$F_{ij} = P\{T_j < \infty \mid X_0 = i\}$$

Sendo assim, F_{jj} denota a probabilidade da ocorrência do primeiro retorno a j.

Definição A.3. *Estado Recorrente*

Um estado $j \in I$ *pertencente a uma CMTC* $\{X_t, t \geq 0\}$ *é dito* recorrente *se* $F_{jj} = 1$*.*

Definição A.4. *Estado Recorrente Positivo*

Um estado $j \in I$ *pertencente a uma CMTC* $\{X_t, t \geq 0\}$ *é dito* recorrente positivo *se*

$$F_{jj} = 1 \qquad e \qquad \mathbb{E}(T_j \mid X_0 = j) < \infty ,$$

ou seja, em um estado recorrente positivo, o tempo médio de recorrência a si mesmo é finito.

Definição A.5. *Cadeia Irredutível*

Uma CMTC $\{X_t, t \geq 0\}$ *é dita* irredutível *se para todo par de estados* $(i, j) \in I$*,* $F_{ij} > 0$*, ou seja, qualquer estado na cadeia é acessível a partir de um outro estado da cadeia.*

Definição A.6. *Cadeia Ergódica*

Uma CMTC $\{X_t, t \geq 0\}$ *é dita* ergódica *se ela é irredutível e se todos os seus estados são recorrentes positivos.*

A.4 Análise em Regime Estacionário

Seja $\{X_t, t \geq 0\}$ uma CMTC ergódica. É possível provar que existe uma distribuição de probabilidades $\{\pi_j : j \in I\}$ tal que

$$lim_{t\to\infty} p_{ij}(t) = \pi_j$$

independentemente do estado inicial i.

O vetor de probabilidades π é a única solução para o conjunto de equações lineares

$$\nu_j \pi_j = \sum_{k \neq j} q_{kj} \pi_k, \quad j \in I \tag{A.1}$$

$$\sum_{j \in I} \pi_j = 1 \tag{A.2}$$

nas incógnitas π_j, $j \in I$.

As equações lineares A.1 são chamadas de *equações de equilíbrio* do processo de Markov. A Equação A.2 é uma equação de *normalização*.

As equações de equilíbrio também podem ser escritas na forma matricial, como

$$\pi Q = 0 \tag{A.3}$$

em que Q é o gerador infinitesimal da cadeia.

A distribuição π é a *distribuição estacionária* da CMTC. A probabilidade π_j pode ser interpretada como *a fração de tempo no regime estacionário em que o processo está no estado j com probabilidade 1.*

As equações de equilíbrio expressam o princípio que, no regime estacionário do sistema, vale a igualdade:

$$\text{taxa de saída do estado } j = \text{taxa de entrada no estado } j.$$

A.5 Análise Transiente

Em muitas situação práticas, estamos interessados no comportamento transiente do sistema, ao invés do comportamento no regime estacionário. Por meio da análise transiente, podemos obter resultados como os seguintes:

- o estado do processo no final de um dado intervalo de tempo;
- o tempo de espera até a ocorrência de um dado evento;
- o tempo de permanência em um conjunto de estados durante um dado intervalo de tempo;
- o número de ocorrências de determinados eventos em um dado intervalo de tempo.

As probabilidades transientes de uma CMTC $\{X_t, t \geq 0\}$ são definidas por

$$p_{ij}(t) = P\{X_t = j \mid X_0 = i\}, \qquad i, j \in I \ .$$

Um dos possíveis métodos para se calcular essas probabilidades são as equações de Kolmogorov, definidas a seguir.

Definição A.7. *Equações Regressivas de Kolmogorov*
Seja $\{X_t, t \geq 0\}$ *uma CMTC. Para todo* $i, j \in I$ *e* $t > 0$,

$$p'_{ij}(t) = \sum_{k \neq j} q_{ik} p_{kj}(t) - \nu_i p_{ij}(t) \ . \tag{A.4}$$

Definição A.8. *Equações Progressivas de Kolmogorov*
Seja $\{X_t, t \geq 0\}$ *uma CMTC. Para todo* $i, j \in I$ *e* $t > 0$,

$$p'_{ij}(t) = \sum_{k \neq j} q_{kj} p_{ik}(t) - \nu_j p_{ij}(t) \ . \tag{A.5}$$

O conjunto de equações A.4 é chamado de equações regressivas, porque o cálculo da distribuição de probabilidade do estado no tempo $t + h$ está condicionado ao estado do caminho de volta ao tempo h, ou seja, o cálculo é iniciado por

$$p_{ij}(t+h) = \sum_k P\{X_{t+h} = j \mid X_0 = i, X_h = k\} P\{X_h = k \mid X_0 = i\} = \sum_k p_{kj}(t) p_{ik}(h) \ .$$

De forma análoga, as equações progressivas A.5 condicionam o estado no tempo $t+h$ ao estado no tempo t:

$$p_{ij}(t+h) = \sum_k p_{ik}(t) p_{kj}(h) \ .$$

Apêndice B

Álgebra Tensorial

Este capítulo apresenta a definição dos operadores da Álgebra Tensorial Clássica (ATC) e da sua extensão Álgebra Tensorial Generalizada (ATG), de acordo com a as definições extraídas do trabalho de Fernandes et al. [FPS98, Fer98].

B.1 Álgebra Tensorial Clássica

A Álgebra Tensorial Clássica é definida por dois operadores matriciais: o *produto tensorial* e a *soma tensorial*.

Considere duas matrizes A e B que são os geradores infinitesimais de duas CMTCs (CMTDs). A soma tensorial de A e B, denotada por $A \oplus B$, equivale à matriz do gerador infinitesimal das duas CMTCs (CMTDs) combinadas. O produto tensorial de A e B, denotado por $A \otimes B$, descreve saltos simultâneos ou dependências entre as CMTCs (CMTDs).

Considere as seguintes matrizes A e B:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & a_{42} & b_{43} & b_{44} \end{pmatrix} .$$

O produto tensorial $C = A \otimes B$ é dado por:

$$C = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix}$$

$$= \left(\begin{array}{cccc|cccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{11}b_{14} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} & a_{12}b_{14} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{11}b_{24} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} & a_{12}b_{24} \\ a_{11}b_{31} & a_{11}b_{32} & a_{11}b_{33} & a_{11}b_{34} & a_{12}b_{31} & a_{12}b_{32} & a_{12}b_{33} & a_{12}b_{34} \\ a_{11}b_{41} & a_{11}a_{42} & a_{11}b_{43} & a_{11}b_{44} & a_{12}b_{41} & a_{12}a_{42} & a_{12}b_{43} & a_{12}b_{44} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{21}b_{14} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} & a_{22}b_{14} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{21}b_{24} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} & a_{22}b_{24} \\ a_{21}b_{31} & a_{21}b_{32} & a_{21}b_{33} & a_{21}b_{34} & a_{22}b_{31} & a_{22}b_{32} & a_{22}b_{33} & a_{22}b_{34} \\ a_{21}b_{41} & a_{21}a_{42} & a_{21}b_{43} & a_{21}b_{44} & a_{22}b_{41} & a_{22}b_{42} & a_{22}b_{43} & a_{22}b_{44} \end{array}\right) .$$

O produto tensorial de duas matrizes A e B de dimensões $\alpha_1 \times \alpha_2$ e $\beta_1 \times \beta_2$, respectivamente, é uma matriz de dimensões $\alpha_1\beta_1 \times \alpha_2\beta_2$, que pode ser vista como $\alpha_1\alpha_2$ blocos, cada um contendo $\beta_1\beta_2$ elementos. O elemento de $C = A \otimes B$ que está na posição (k, l) do bloco (i, j) possui o valor $a_{ij}b_{kl}$, i.e., $c_{[ik][jl]} = a_{ij}b_{kl}$.

A soma tensorial de duas matrizes quadradas A e B é definida em termos do produto tensorial:

$$A \oplus B = A \otimes I_{n_2} + I_{n_1} \otimes B$$

em que n_1 é a ordem de A, n_2 é a ordem de B, I_{n_i} é a matriz identidade de ordem n_i e "+" é a operação usual de soma de matrizes.

Por exemplo, considere as seguintes matrizes A e B:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} .$$

A soma tensorial $C = A \oplus B$ é dada por:

$$C = \left(\begin{array}{ccc|ccc} a_{11} & 0 & 0 & a_{12} & 0 & 0 \\ 0 & a_{11} & 0 & 0 & a_{12} & 0 \\ 0 & 0 & a_{11} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22} & 0 & 0 \\ 0 & a_{21} & 0 & 0 & a_{22} & 0 \\ 0 & 0 & a_{21} & 0 & 0 & a_{22} \end{array}\right) + \left(\begin{array}{ccc|ccc} b_{11} & b_{12} & b_{13} & 0 & 0 & 0 \\ b_{21} & b_{22} & b_{23} & 0 & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 0 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 0 & b_{31} & b_{32} & b_{33} \end{array}\right)$$

$$= \left(\begin{array}{ccc|ccc} a_{11}+b_{11} & b_{12} & b_{13} & a_{12} & 0 & 0 \\ b_{21} & a_{11}+b_{22} & b_{23} & 0 & a_{12} & 0 \\ b_{31} & b_{32} & a_{11}+b_{33} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22}+b_{11} & b_{12} & b_{13} \\ 0 & a_{21} & 0 & b_{21} & a_{22}+b_{22} & b_{23} \\ 0 & 0 & a_{21} & b_{31} & b_{32} & a_{22}+b_{33} \end{array}\right) .$$

O elemento de $C = A \oplus B$ que está na posição (k, l) do bloco (i, j) possui o valor $a_{ij}\delta_{kl} + b_{kl}\delta_{ij}$, ou seja, $c_{[ik][jl]} = a_{ij}\delta_{kl} + b_{kl}\delta_{ij}$, em que δ_{ij} é o elemento da $i^{\text{ésima}}$ linha e $j^{\text{ésima}}$ coluna de uma matriz identidade definida como $\delta_{ij} = 1$ para $i = j$ e $\delta_{ij} = 0$ para $i \neq j$. Dado que os dois lados da operação de soma de matrizes precisam possuir a mesma dimensão, segue que a soma tensorial está apenas definida para matrizes quadradas.

B.2 Álgebra Tensorial Generalizada

A Álgebra Tensorial Generalizada (ATG) é uma extensão da ATC. A ATG introduziu na ATC o conceito de *elementos funcionais* (ou seja, elementos não constantes). Um elemento funcional é uma função avaliada em $\mathbb{R}$ de acordo com um conjunto de parâmetros compostos por linhas de

uma ou mais matrizes.

A operação de produto na ATG é denotada por $\otimes_g$. O valor do elemento da posição (k, l) do bloco (i, j) de um produto tensorial generalizado $C = A(B) \otimes_g B(A)$ é dado por

$$C_{[ik][jl]} = a_{ij}(b_k)b_{kl}(a_i).$$

A operação de soma na ATG, denotada por $\oplus_g$, também é análoga à soma da ATC.
O valor do elemento da posição (k, l) do bloco (i, j) de $C = A \oplus_g B$ é dado por

$$C_{[ik][jl]} = a_{ij}(b_k)\delta_{kl} + b_{kl}(a_i)\delta_{ij} \ .$$

Referências Bibliográficas

[AM07] Tayfur Altiok e Benjamin Melamed. *Simulation modeling and analysis with Arena*. Academic Press, Newark, NJ, EUA, 2007. 89, 90

[Are] Arena. Arena® simulation software. http://www.arenasimulation.com/. [Acessado em 11 de julho de 2011]. 89

[Bal01] Gianfranco Balbo. Introduction to stochastic Petri nets. Em Ed Brinksma, Holger Hermanns, e Joost-Pieter Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 de *Lecture Notes in Computer Science*, páginas 84–155. Springer, 2001. 21, 22

[Bal07] Gianfranco Balbo. Introduction to generalized stochastic Petri nets. Em *Formal Methods for Performance Evaluation*, volume 4486 de *Lecture Notes in Computer Science*, páginas 83–131. Springer, Bertinoro, Itália, maio 2007. xi, 20, 21, 22

[BB87] Tommaso Bolognesi e Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59, 1987. 25

[BB08] Michel Benaïm e Jean-Yves Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65:823–838, novembro 2008. 95

[BBFP04] Anne Benoit, Leonardo Brenner, Paulo H. L. Fernandes, e Brigitte Plateau. Aggregation of stochastic automata networks with replicas. *Linear Algebra and its Applications*, 386:111–136, 2004. 60, 74

[BCD+95] Gianfranco Balbo, Gianni Conte, Susanna Donatelli, Giuliana Franceschinis, e Marco Ajmone Marsan. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, Inc., novembro 1995. 22

[BCFH11] Rena Bakhshi, Lucia Cloth, Wan Fokkink, e Boudewijn R. Haverkort. Mean-field framework for performance evaluation of push-pull gossip protocols. *Performance Evaluation*, 68:157–179, fevereiro 2011. 96

[BFPS07] Leonardo Brenner, Paulo H. L. Fernandes, Brigitte Plateau, e Ihab Sbeity. PEPS2007 - Stochastic Automata Networks software tool. Em *Proceedings of the Fourth International Conference on Quantitative Evaluation of Systems*, páginas 163–164, Edimburgo, Reino Unido, setembro 2007. IEEE Computer Society. 37, 61

[BFS05] Leonardo Brenner, Paulo H. L. Fernandes, e Afonso Sales. The need for and the advantages of generalized tensor algebra for Kronecker structured representations. *International Journal of Simulation: Systems, Science & Technology*, 6(3–4):52–60, 2005. 35

[BFV09] Kelly R. Braghetto, João E. Ferreira, e Jean-Marc Vincent. Comparison of modeling approaches to business process performance evaluation. Relatório Técnico 7065, INRIA, outubro 2009. 5, 50

[BFV10] Kelly R. Braghetto, João E. Ferreira, e Jean-Marc Vincent. Performance analysis modeling applied to business processes. Em *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim'10, páginas 122:1–122:8, Orlando, FL, EUA, abril 2010. ACM. 5, 50

[BFV11a] Kelly R. Braghetto, João E. Ferreira, e Jean-Marc Vincent. From Business Process Model and Notation to Stochastic Automata Network. Relatório Técnico RT-MAC-2011-03, Universidade São Paulo, março 2011. 5, 63

[BFV11b] Kelly R. Braghetto, João E. Ferreira, e Jean-Marc Vincent. Performance evaluation of business processes through a formal transformation to SAN. Em *Proceedings of the 8th European Performance Engineering Workshop*, EPEW 2011, Borrowdale, Reino Unido, outubro 2011. [Artigo aceito para publicação]. 5, 63

[BFV11c] Kelly R. Braghetto, João E. Ferreira, e Jean-Marc Vincent. Performance evaluation of resource-aware business processes using stochastic automata networks. *International Journal of Innovative Computing, Information and Control*, 2011. [Artigo submetido]. 5, 82

[BG96] Marco Bernardo e Roberto Gorrieri. Extended Markovian process algebra. Em *Proceedings of Concurrency Theory*, volume 1119 de *Lecture Notes in Computer Science*, páginas 315–330, Pisa, Itália, agosto 1996. Springer. 26

[BGT08] Andrea Bobbio, Marco Gribaudo, e Miklós Telek. Analysis of large scale interacting systems by mean field method. Em *Proceedings of the Fifth International Conference on Quantitative Evaluation of Systems*, páginas 215–224, Saint-Malo, França, setembro 2008. IEEE Computer Society. 96

[BK84] Jan Bergstra e Jan Willem Klop. Process algebra for synchronous communication. *Information and Control*, 60(1–3):109–137, 1984. 25

[BK08] Christel Baier e Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008. 16

[BP2] BP2SAN. From Business Processes to Stochastic Automata Networks. http://www.ime.usp.br/~kellyrb/bp2san/. [Acessado em 11 de julho de 2011]. 5, 61, 65, 82

[BPM] BPMI. Business Process Management Initiative. http://www.bpmi.org. [Acessado em 11 de julho de 2011]. 2

[Bre04] Leonardo Brenner. Agregação de redes de autômatos estocásticos. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, janeiro 2004. 32, 60

[Bre09] Leonardo Brenner. *Réseaux d'Automates Stochastiques: Analyse trasitoire en temps continu et algèbre tensorielle pour une sémantique en temps discret*. Tese de Doutorado, Institut Polytechnique de Grenoble, França, setembro 2009. 32

[BRvU00] Jörg Becker, Michael Rosemann, e Christoph von Uthmann. Guidelines of business process modeling. Em Wil van der Aalst, Jörg Desel, e Andreas Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 de *Lecture Notes in Computer Science*, páginas 241–262. Springer, 2000. 1, 9

[CGH+03] Catherine Canevet, Stephen Gilmore, Jane Hillston, Matthew Prowse, e P. Perdita Stevens. Performance modelling with the unified modelling language and stochastic process algebras. *IEE Proceedings Computers and Digital Techniques*, 150(2):107–120, março 2003. 6

[CGHT07] Allan Clark, Stephen Gilmore, Jane Hillston, e Mirco Tribastone. Stochastic process algebras. Em *Formal Methods for Performance Evaluation*, volume 4486 de *Lecture Notes in Computer Science*, páginas 132–179. Springer, Bertinoro, Itália, maio 2007. 26

[CJIMS06] Gianfranco Ciardo, Robert L. Jones III, Andrew S. Miner, e Radu I. Siminiceanu. Logic and stochastic modeling with SMART. *Performance Evaluation*, 63:578–608, junho 2006. 25

[CKO92] Bill Curtis, Marc I. Kellner, e Jim Over. Process modeling. *Communications of the ACM*, 35:75–90, setembro 1992. 9

[CT96] Gianfranco Ciardo e Marco Tilgner. On the use of Kronecker operators for the solution of generalized stochastic Petri nets. Relatório Técnico 96-35, Institute for Computer Applications in Science and Engineering (ICASE), maio 1996. 37

[DDO08] Remco M. Dijkman, Marlon Dumas, e Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software Technology*, 50(12):1281–1294, 2008. 6

[Don94] Susanna Donatelli. Superposed generalized stochastic Petri nets: Definition and efficient solution. Em *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*, volume 815 de *Lecture Notes in Computer Science*, páginas 258–277, Zaragoza, Espanha, junho 1994. Springer. 37

[DR77] Carlos A. B. Dantas e Flávio W. Rodrigues. *Tópicos de Processos Estocásticos*. Instituto de Matemática Pura e Aplicada, 1977. 97

[EF08] Andreas Eckleder e Thomas Freytag. WoPeD 2.0 goes BPEL 2.0. Em *15th German Workshop on Algorithms and Tools for Petri Nets, Algorithmen und Werkzeuge für Petrinetze*, CEUR Workshop Proceedings, páginas 75–80, Rostock, Alemanha, 2008. CEUR-WS.org. 90

[Esh02] Rik Eshuis. *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*. Tese de Doutorado, University of Twente, novembro 2002. 6

[Fer98] Paulo H. L. Fernandes. *Méthodes Numériques pour la Solution de Systèmes Markoviens à Grand Espace d'États*. Tese de Doutorado, Institut Polytechnique de Grenoble, França, fevereiro 1998. 4, 32, 33, 103

[FFN91] Gerard Florin, Céline Fraize, e Stéphane Natkin. Stochastic Petri nets: Properties, applications and tools. *Microelectronics Reliability*, 31(4):669–697, 1991. 17, 22

[FM03] Paul J. Fortier e Howard Edgar Michel. *Computer systems performance evaluation and prediction*. Enterprise computing. Digital Press, 2003. 14

[FPS98] Paulo H. L. Fernandes, Brigitte Plateau, e William J. Stewart. Efficient descriptor-vector multiplications in stochastic automata networks. *Journal of the ACM*, 45:381–414, maio 1998. 103

[GGB10] Nicolas Gast, Bruno Gaujal, e Jean-Yves Le Boudec. Mean field for Markov Decision Processes: from discrete to continuous optimization. Relatório Técnico 7239, École Polytechnique Fédérale de Lausanne, abril 2010. 96

[GN00] Emden R. Gansner e Stephen C. North. An open graph visualization system and its applications to software engineering. *Software – Practice & Experience*, 30:1203–1233, setembro 2000. 61

[Her98] Ulrich Herzog. Stochastic process algebras: Benefits for performance evaluation and challenges. Em *Proceedings of Concurrency Theory*, volume 1466 de *Lecture Notes in Computer Science*, páginas 366–372, Nice, França, setembro 1998. Springer. 16, 26

[HHK02] H. Hermanns, U. Herzog, e J. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274:43–87, março 2002. 15

[HHM98] Holger Hermanns, Ulrich Herzog, e Vassilis Mertsiotakis. Stochastic process algebras – between LOTOS and Markov chains. *Computer Networks*, 30(9–10):901–924, 1998. 26

[Hil96] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, Nova York, NY, EUA, 1996. 26, 27

[HK01] Jane Hillston e Lëila Kloul. An efficient Kronecker representation for PEPA models. Em *Process Algebra and Probabilistic Methods. Performance Modelling and Verification*, volume 2165 de *Lecture Notes in Computer Science*, páginas 120–135, Aachen, Alemanha, setembro 2001. Springer. 37

[Hoa78] Charles Antony Richard Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, agosto 1978. 25

[HR98] Jane Hillston e Marina Ribaudo. Stochastic process algebras: A new approach to performance modeling. Em *Modeling and Simulation of Advanced Computer Systems*, páginas 235–256, Nova York, NY, EUA, 1998. Gordon Breach. 26

[Jai91] Raj Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley Professional Computing. John Wiley & Sons, Inc., Nova York, NY, EUA, 1991. 2, 15, 17

[JVN06] Monique Jansen-Vullers e Mariska Netjes. Business process simulation – a tool survey. Em *Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, CPN'06, páginas 80–96, Aarhus, Denmark, outubro 2006. University of Aarhus, Denmark. 90

[LGMC04] Juan Pablo López-Grao, José Merseguer, e Javier Campos. From UML activity diagrams to stochastic Petri nets: application to software performance engineering. *SIGSOFT Software Engineering Notes*, 29:25–36, janeiro 2004. 6

[Lil05] Davida J. Lilja. *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2005. 17

[LZGS84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, e Kenneth C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., Upper Saddle River, NJ, EUA, 1984. 17, 66

[MCB84] Marco Ajmone Marsan, Gianni Conte, e Gianfranco Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, maio 1984. 22

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989. 25

[Mil99] Robin Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999. 25

[MM07] Matteo Magnani e Danilo Montesi. BPMN: How much does it cost? An incremental approach. Em *Proceedings of the 5th International on Conference Business Process Management*, Lecture Notes in Computer Science, páginas 80 – 87, Brisbane, Australia, setembro 2007. Springer. 90

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–574, abril 1989. 18, 95

[MV03] Klaus Müller e Tony Vignaux. SimPy: Simulating systems in Python. *ONLamp.com Python DevCenter*, fevereiro 2003. 90

[Ofe] BPMN Berlin Ofensive. BPMN Poster (versão em português) – traduzido por Lucinéia Heloisa Thom e Cirano Iochpe. http://www.bpmb.de/index.php/BPMNPoster. [Acessado em 11 de julho de 2011]. 10

[OFMP09] Márcio K. Oikawa, João E. Ferreira, Simon Malkowski, e Calton Pu. Towards algorithmic generation of business processes: From business step dependencies to process algebra expressions. Em *Proceedings of the 7th International Conference on Business Process Management*, Lecture Notes in Computer Science, páginas 80–96, Ulm, Alemanha, setembro 2009. Springer. 95

[OL09] Cesar Oliveira e Ricardo Lima. Performance analysis of resource-constrained business processes: a formal approach based on stochastic Petri nets. Relatório técnico, Universidade Federal do Pernambuco, março 2009. 6

[OLAR09] Cesar Oliveira, Ricardo Lima, Thiago Andre, e Hajo A. Reijers. Modeling and analyzing resource-constrained business processes. Em *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, SMC'09, páginas 2824–2830, San Antonio, TX, EUA, outubro 2009. IEEE Press. 6

[OMG] OMG. Object Management Group. http://www.omg.org/. [Acessado em 11 de julho de 2011]. 2, 10

[OMG10] OMG. Unified modeling language specification (UML), version 2.3, 2010. 6

[OMG11a] OMG. BPMN 2.0 by example, version 1.0 (non-normative), 2011. xi, 61

[OMG11b] OMG. Business process model and notation (BPMN), version 2.0, 2011. 2, 10, 12, 13

[PA91] Brigitte Plateau e Karim Atif. Stochastic automata network for modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093 –1108, 1991. 4, 32

[PEPa] PEPA. PEPA Plug-in Project. http://www.dcs.ed.ac.uk/pepa/tools/plugin. [Acessado em 11 de julho de 2011]. 31

[PEPb] PEPS. Performance Evaluation of Parallel Systems. http://www-id.imag.fr/Logiciels/peps/. [Acessado em 11 de julho de 2011]. 37

[Pla85] Brigitte Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *ACM SIGMETRICS Performance Evaluation Review*, 13(2):147–154, agosto 1985. 4, 32

[PQZ08] Davide Prandi, Paola Quaglia, e Nicola Zannone. Formal analysis of BPMN via a translation into COWS. Em *Proceedings of the 10th international conference on Coordination Models and Languages*, páginas 249–263, Oslo, Noruega, junho 2008. Springer. 6

[Rei03] Hajo A. Reijers. *Design and control of workflow processes: business process management for the service industry*. Springer, Berlim, Alemanha, 2003. 6

[Ros96] Sheldon M. Ross. *Stochastic Processes*. John Wiley & Sons, Inc., Nova York, NY, EUA, 1996. 94, 97

[Ros97] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, Nova York, NY, EUA, 1997. 16

[Rus07] Nicholas Charles Russell. *Foundations of Process-Aware Information Systems*. Tese de Doutorado, Queensland University of Technology, Brisbane, Austrália, dezembro 2007. 9

[RvdAtHE05] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, e David Edmond. Workflow resource patterns: Identification, representation and tool support. Em *Advanced Information Systems Engineering*, volume 3520 de *Lecture Notes in Computer Science*, páginas 11–42, Porto, Portugal, junho 2005. Springer. 13

[Sal03] Afonso Sales. Formalismos estruturados de modelagem para sistemas Markovianos complexos. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, junho 2003. 32

[Sal09] Afonso Sales. *Réseaux d'Automates Stochastiques: Génération de l'espace d'états atteignables et Multiplication vecteur-descripteur pour une sémantique en temps discret*. Tese de Doutorado, Institut Polytechnique de Grenoble, França, setembro 2009. 32

[SAP95] William J. Stewart, Karim Atif, e Brigitte Plateau. The numerical solution of stochastic automata networks. *European Journal of Operational Research*, 86(3):503–525, novembro 1995. 4

[SC81] Charles H. Sauer e K. Mani Chandy. *Computer Systems Performance Modeling*. Prentice Hall, Englewood Cliffs, NY, EUA, 1981. 6

[SMA] SMART. Stochastic Model checking Analyzer for Reliability and Timing. http://www.cs.ucr.edu/~ciardo/SMART/. [Acessado em 11 de julho de 2011]. 25

[Smi10] Michael J. A. Smith. Abstraction and model checking in the PEPA Plug-In for Eclipse. Em *Proceedings of the Seventh International Conference on the Quantitative Evaluation of Systems*, QEST'10, páginas 155–156, Williamsburg, VA, EUA, setembro 2010. IEEE Computer Society. 31

[STA05] August-Wilhelm Scheer, Oliver Thomas, e Otmar Adam. *Process Modeling using Event-Driven Process Chains*, páginas 119–145. John Wiley & Sons, Inc., 2005. 6

[Tij03] Henk C. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons, Ltd., West Sussex, Reino Unido, 2003. 97

[vdA99] Wil M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639 – 650, julho 1999. 6

[vdA11] Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlim, Alemanha, 2011. 95

[vdAtHKB03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartosz Kiepuszewski, e Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003. 9, 42

[vdAtHW03] Wil van der Aalst, Arthur ter Hofstede, e Mathias Weske. Business process management: A survey. Em *Proceedings of the International Conference on Business Process Management*, volume 2678 de *Lecture Notes in Computer Science*, páginas 1019–1019, Eindhoven, Holanda, junho 2003. Springer. 1

[vDMvdA06] Boudewijn F. van Dongen, Jan Mendling, e Wil M. P. van der Aalst. Structural patterns for soundness of business process models. Em *Proceedings of the Tenth IEEE International Enterprise Distributed Object Computing Conference*, páginas 116–128, Hong Kong, China, outubro 2006. IEEE Computer Society. 14

[WFM] WFMC. Workflow Management Coalition. http://www.wfmc.org. [Acessado em 11 de julho de 2011]. 39

[WfM99] WfMC. Workflow management coalition terminology and glossary. Relatório Técnico WfMC-TC-1011, Workflow Management Coalition, Brussels, fevereiro 1999. 9

[WG08] Peter Y. H. Wong e Jeremy Gibbons. A process semantics for BPMN. Em *Proceedings of the 10th International Conference on Formal Methods and Software Engineering*, volume 5256 de *Lecture Notes in Computer Science*, páginas 355–374, Kitakyushu, Japão, outubro 2008. Springer. 6