Técnicas de Mineração de Dados para Descoberta de Modelos de Processos de Negócio

Kelly Rosa Braghetto

Orientador: João Eduardo Ferreira
Co-orientador: Roberto M. Cesar Jr.

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

São Paulo, 10 de Dezembro de 2008.

* Este trabalho recebe apoio financeiro da FAPESP *

Conteúdo

Gerenciamento de Processos de Negócio (GPN)

O GPN envolve métodos, técnicas e ferramentas para apoiar o projeto, a execução e a análise operacional dos processos de negócio.

- Definições tradicionais de *workflow* priorizam a execução dos processos operacionais;
- O GPN apóia também a fase de análise dos processos de negócio, cobrindo aspectos negligenciados pelos produtos de *workflow* tradicionais, como diagnóstico e simulação.

Mineração de Processos I

O objetivo da Mineração de Processos é extrair um modelo explícito de processo a partir de *logs* contendo seqüências de eventos gerados por um sistema computacional em execução.

Os meios de representação de um modelo de processo são:

Arcabouços formais

⇓

Exs: Redes de Petri e Álgebras de Processos

⇓

Modelos que podem ser analisados

Linguagens de especificação

⇓

Exs: BPEL e BPMN

⇓

Modelos mais apropriados para a execução

Mineração de Processos II

A Mineração de Processos é análoga à Inferência Gramatical: o problema de aprendizagem de uma gramática a partir de um conjunto amostral de sentenças.

Muitos dos algoritmos para inferência gramatical não são robustos a ruídos, além de possuírem como entrada um conjunto de dados positivos e um conjunto de dados negativos de exemplo.

Problemas: Um arquivo de *log* só contém exemplos positivos e, além disso, pode conter ruídos.

Mineração de Processos - Complicações

Para avaliar uma técnica de mineração de processos, as seguintes questões são importantes:

- É capaz de trabalhar com processos de grande porte (grande número de atividades)?
- Trata somente processos seqüenciais?
- Trata processos que contêm ciclos?
- Permite atividades duplicadas no modelo de processo?
- Identifica dependências de longa distância entre as atividades?
- Trata possíveis ruídos nos dados de entrada?
- Gera modelos que são úteis para a análise do processo?

Objetivo deste Trabalho

Desenvolver uma nova abordagem mista (algorítmica + estatística) para a mineração de processos, que gere modelos em álgebra de processos estocástica.

Justificativa:

- as técnicas de mineração de processos atuais possuem limitações $\Rightarrow$ descoberta de modelos somente para uma classe restrita de processos e susceptibilidade a ruídos;
- tratamento clássico para ruídos e a ausência de exemplos negativos é introduzir probabilidades no modelo, ou seja, inferir gramáticas ou autômatos estocásticos;
- as álgebras de processos são bastante apropriadas para a modelagem e análise de processos concorrentes complexos.

Representação Formal de Processos

O uso de um formalismo na representação de processos é importante, pois ele fornece uma descrição precisa e não-ambígua do processo, além de garantir um arcabouço para a análise de propriedades do sistema.

Os formalismos mais utilizados na modelagem de sistemas concorrentes são:

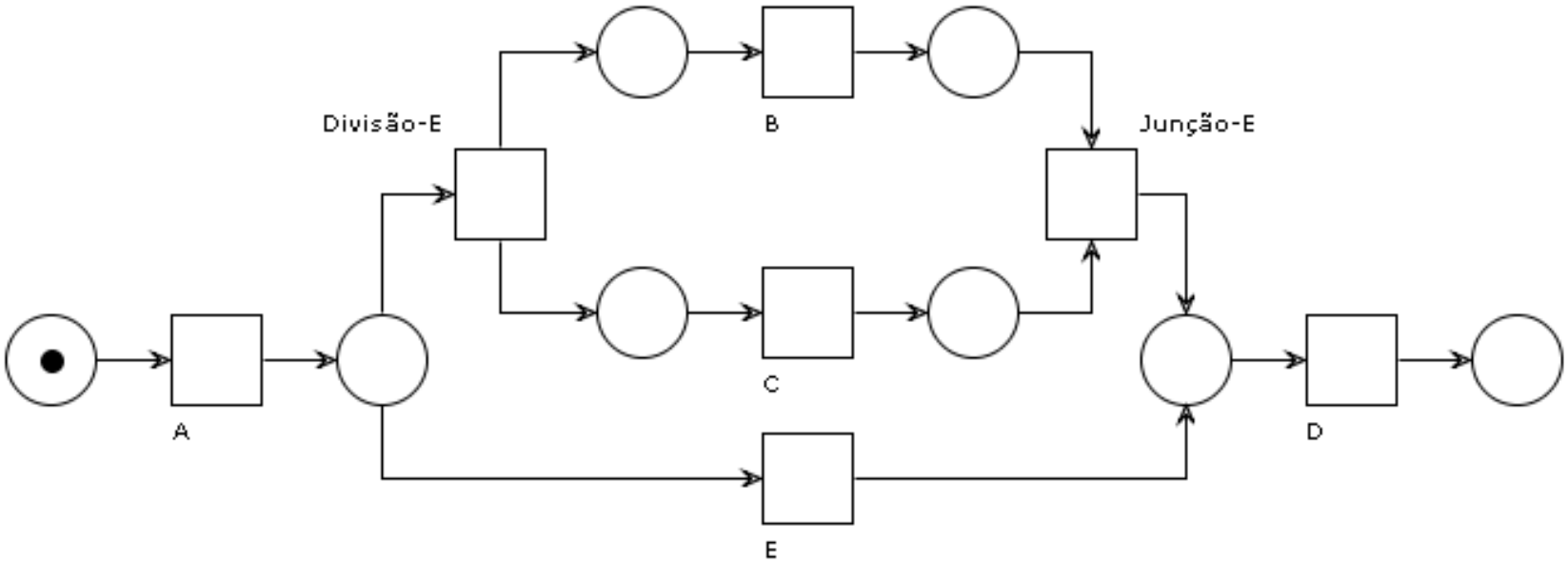
- redes de Petri $\Rightarrow$ representação gráfica; amplamente difundidas;
- álgebras de processos $\Rightarrow$ representação textual; inerentemente composicionais.

Redes de Petri

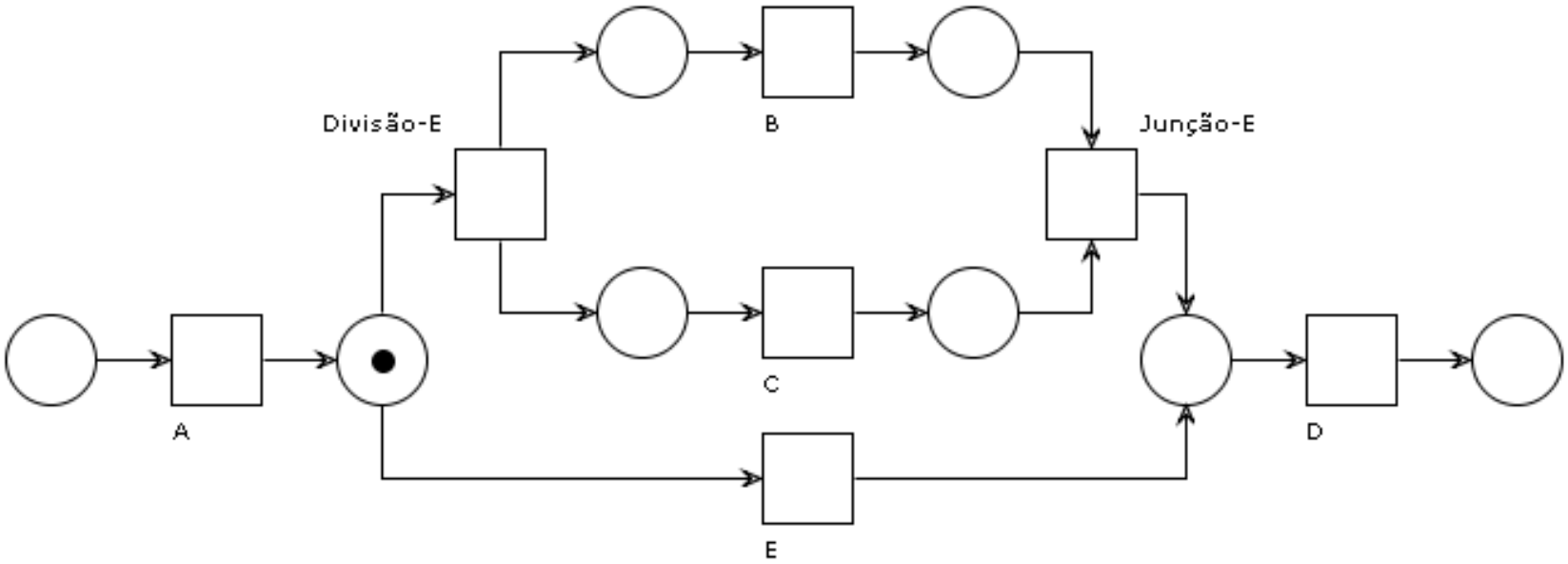
Componentes de uma rede de Petri:

- lugar: modela uma condição que deve ser satisfeita para que o disparo da ação seja realizado;
- transição: pode ser compreendida como uma ação/evento, um processamento de um sinal, ou ainda uma tarefa;
- arco orientado: liga um lugar a uma transição ou vice-versa, encadeando condições e eventos;
- marca ou ficha: representa um recurso disponível;
- peso: cada arco possui um peso associado a ele; o peso indica quantas marcas uma transição consome de um lugar de entrada ou quantas marcas uma transição acrescenta em um lugar de saída.

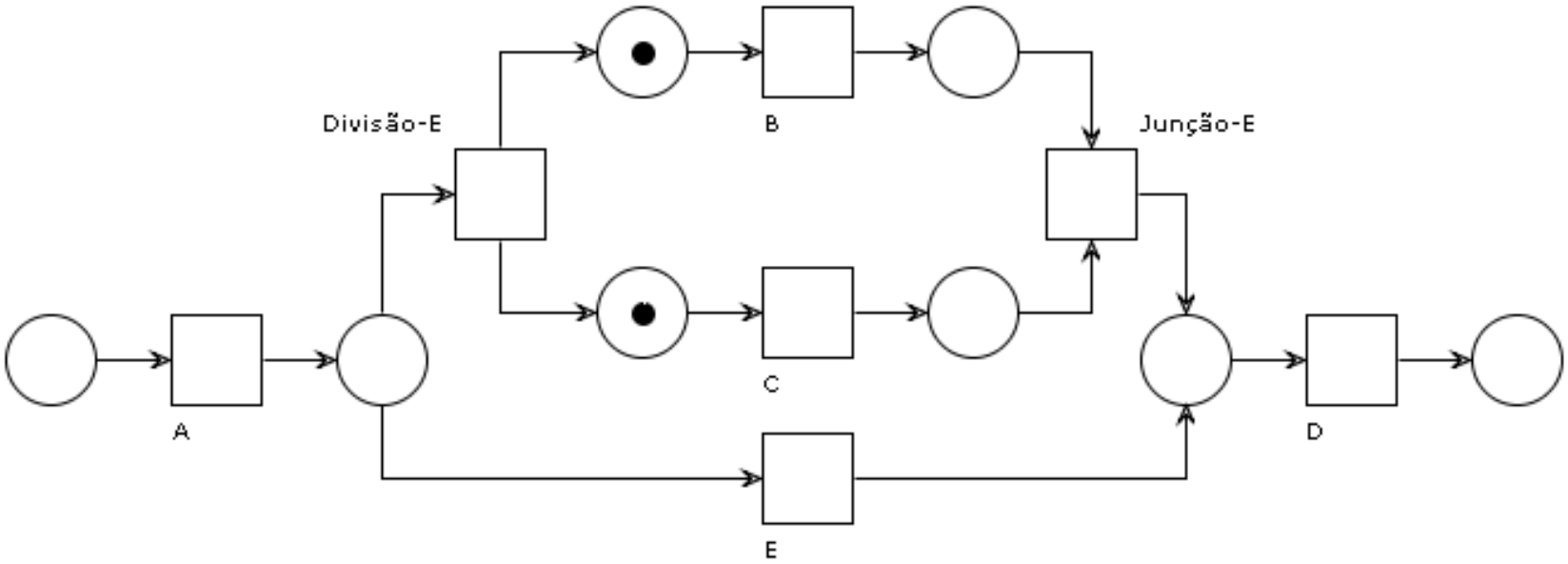
Redes de Petri - Exemplo



Redes de Petri - Exemplo



Redes de Petri - Exemplo



Álgebras de Processos

Representam processos algebricamente, na forma de termos.

Uma álgebra de processos é composta por:

- um conjunto de símbolos de ações (ou eventos);
- um conjunto de operadores – utilizados para compor as ações;
- as regras de transição – que definem a semântica operacional da linguagem;
- um conjunto de axiomas (ou leis equacionais) – descrevem as propriedades dos operadores e pode também especificar quando dois processos são considerados equivalentes.

Álgebras de Processos Estocásticas

Características das álgebras de processos clássicas:

- o tempo associado às ações é implícito;
- os modelos são não-determinísticos.

$\Rightarrow$ A ausência de quantificação do tempo e da incerteza impossibilita a análise de desempenho.

As álgebras de processos estocásticas (APEs) associam a cada ação do modelo uma variável aleatória que representa a sua duração:

- assume-se que as variáveis aleatórias são exponencialmente distribuídas;
- todo modelo de processo descrito em uma APE possui uma cadeia de Markov associada;
- medidas de desempenho podem ser extraídas a partir do modelo.

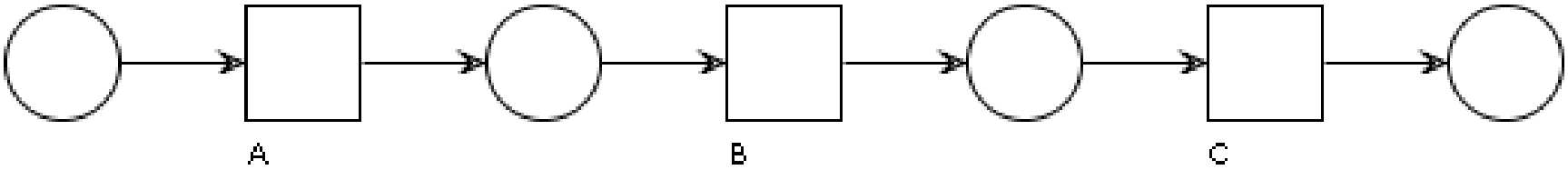
APEs – Sintaxe

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

Considerando que Q e R são processos, C é uma constante e $L \subseteq \mathcal{A}$, tem-se que:

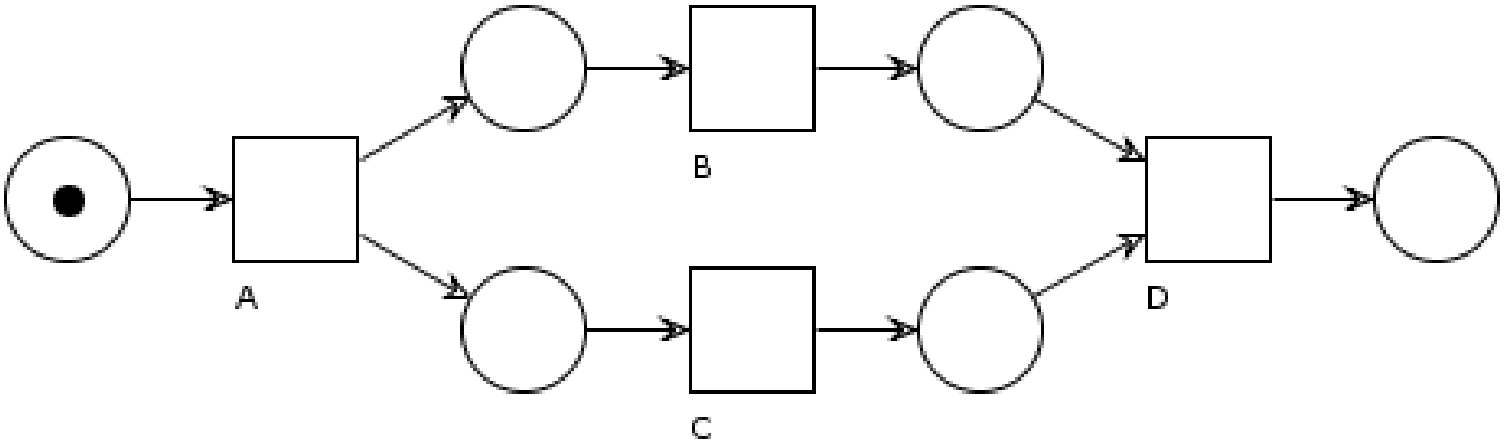
- (α, r) – ação atômica;
- $Q.R$ – composição seqüencial entre Q e R;
- $Q + R$ – escolha entre Q e R;
- $Q \underset{L}{\bowtie} R$ – cooperação entre Q e R, sendo que L define os pontos de sincronização entre os processos;
- $Q \| R$ – cooperação entre Q e R sem pontos de sincronização entre os processos (paralelismo total).

Seqüência



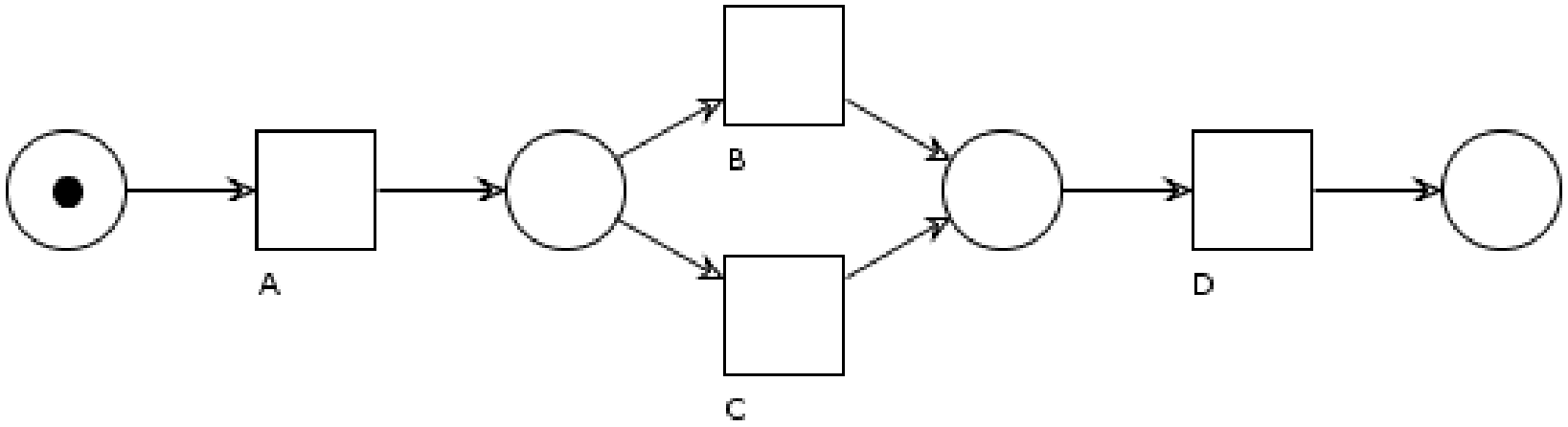
$$P \stackrel{def}{=} A.B.C$$

Divisão-E (ou Paralelismo) e Junção-E (ou Sincronização)



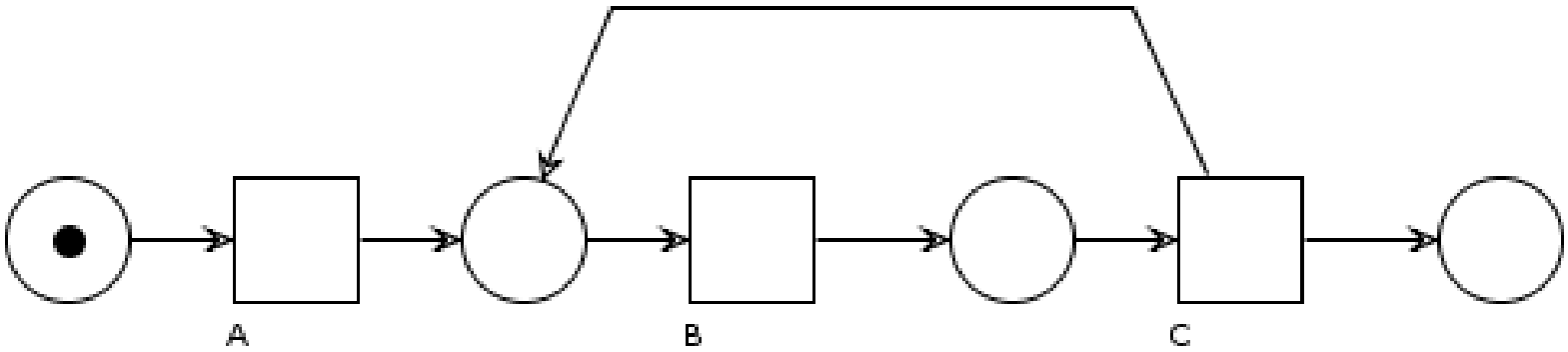
$$P \stackrel{def}{=} A.(B||C).D$$

Divisão-OU (ou Escolha) e Junção-OU (ou Unificação)



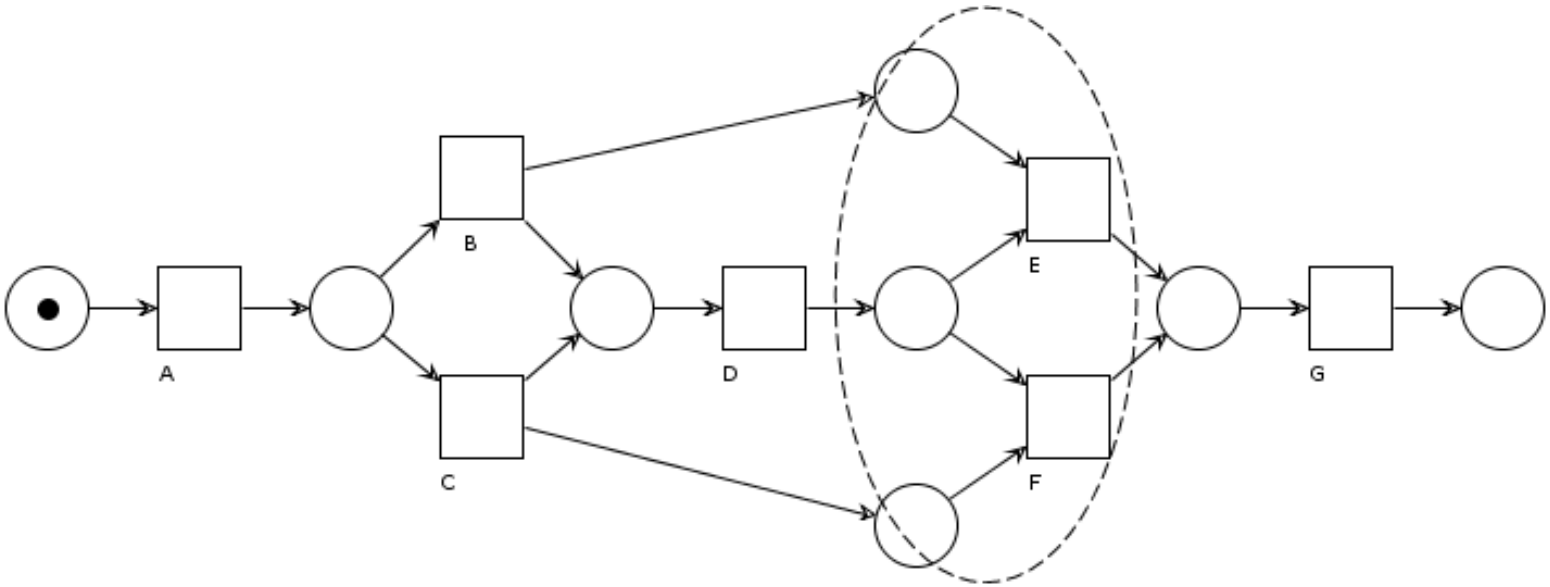
$$P \stackrel{def}{=} A.(B + C).D$$

Ciclo



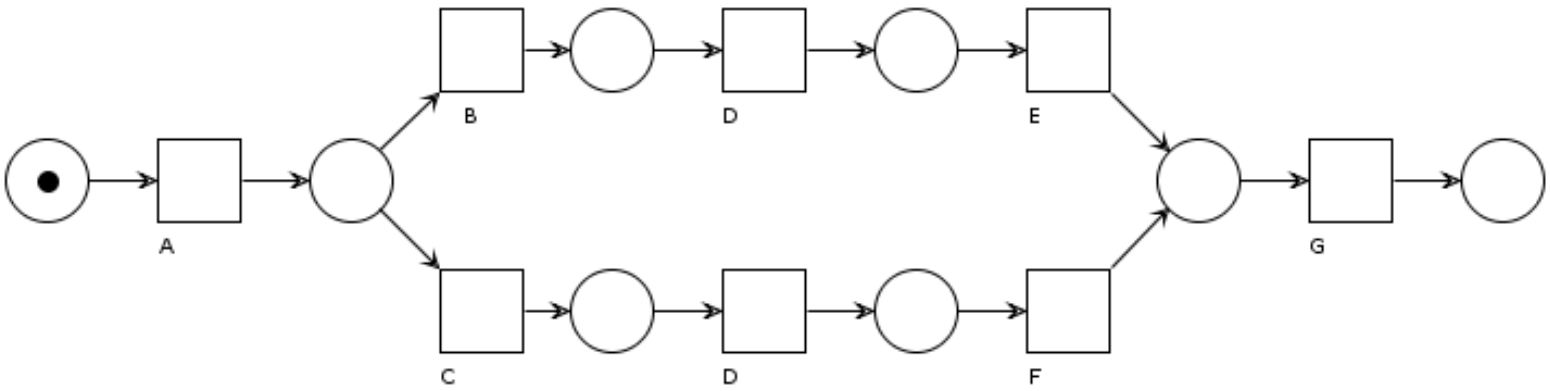
$$P \stackrel{def}{=} A.P_1 \qquad P_1 \stackrel{def}{=} B.(C + C.P_1)$$

Escolha-Dependente



$$P \stackrel{def}{=} A.((B.D.E) + (C.D.F)).G$$

Atividades Repetidas



$$P \stackrel{def}{=} A.((B.D.E) + (C.D.F)).G$$

Logs de Processos

Características dos *logs* empregados neste trabalho:

- cada evento registrado se refere a uma atividade (i.e., um passo bem definido do processo);
- as atividade são atômicas, as execuções são representadas por um único evento;
- cada evento registrado pertence a uma instância de processo;
- os eventos do *log* são totalmente ordenados de acordo com a identificação da instância a que pertencem e a sua data e hora de execução.

Logs de Processos – Exemplo

ID da Instância	ID da Atividade
instância 1	atividade A
instância 1	atividade B
instância 1	atividade C
instância 1	atividade D
instância 2	atividade A
instância 2	atividade C
instância 2	atividade D
instância 3	atividade A
instância 3	atividade C
instância 3	atividade B
instância 3	atividade D
instância 4	atividade A
instância 4	atividade C
instância 4	atividade B
instância 4	atividade D

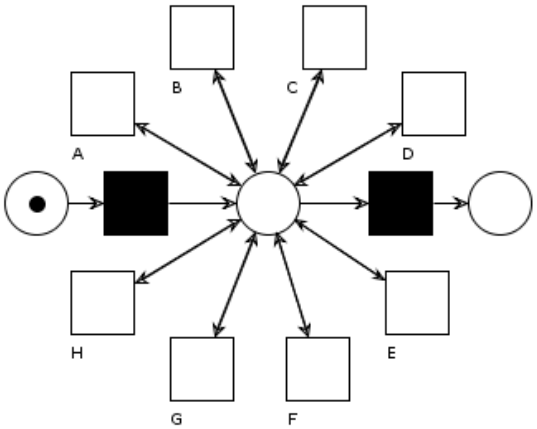
Conformidade entre Logs e Modelos Gerados

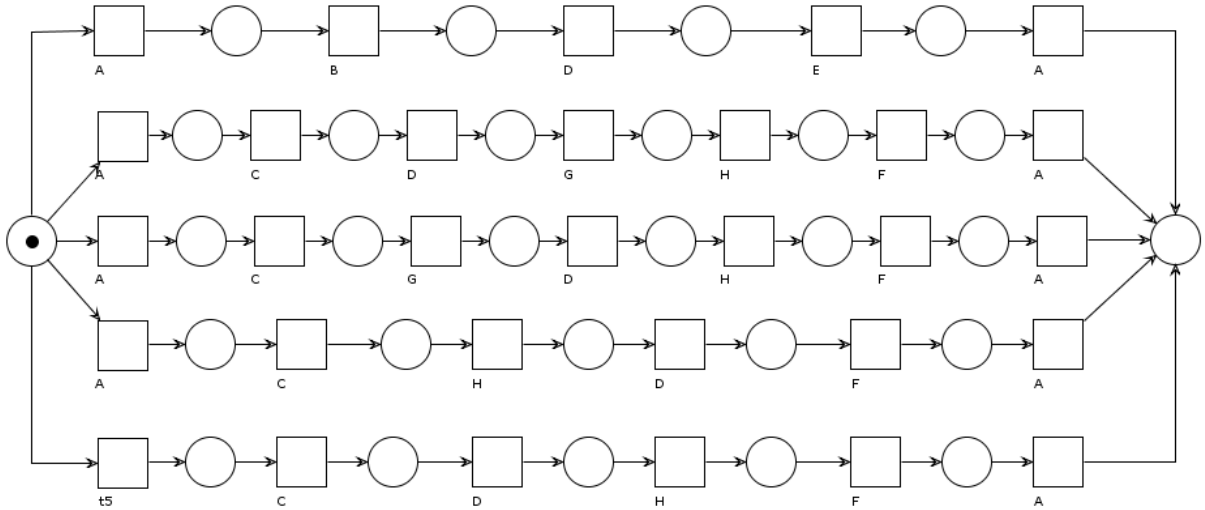
Conformidade pode ser medida em relação a duas dimensões:

- completude – um modelo é completo quando ele reconhece todas as seqüências do *log*;
- precisão
  - estrutural – um modelo é preciso estruturalmente quando ele é minimal com relação à estrutura requerida para refletir de forma clara os comportamentos do *log*;
  - comportamental – um modelo é preciso comportamentalmente quando ele é minimal com relação ao comportamento, de modo que ele represente de forma mais próxima possível só o que é observado no *log*.

Conformidade – Exemplos

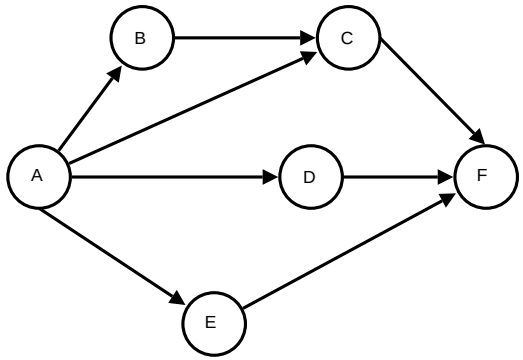
Nº de Instâncias	Seqüências
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA





Grafo de Dependências

- Desenvolvido por Agrawal, Gunopulos e Leymann em 1998;
- Um dos primeiros métodos puramente algorítmicos desenvolvidos para a mineração de processos;
- O método não gera um modelo explícito do processo, mas um grafo representando as dependências existentes entre as atividades observadas no *log*;
- A partir do grafo de dependências não é possível diferenciar casos de paralelismo no processo de casos de escolha entre ramos de execução diferentes $\Rightarrow$ as condições de controle do fluxo não são detectadas pelo algoritmo.



$$\{(ABCF), (ACDF), (ADEF), (AECF)\}$$

Algoritmo α

- Desenvolvido por Aalst, Weijters e Maruster em 2004;
- Um dos primeiros trabalhos bem sucedidos em mineração de processos;
- Gera modelos representados em WF-nets;
- Se baseia em relações de ordenação entre as atividades extraídas do *log* (ex: seqüência direta, causalidade direta e paralelismo potencial);
- A partir das relações, infere a quantidade de lugares que existem na rede e os arcos que os conectam às transições;
- Problemas: não leva em consideração a freqüência com a qual as seqüências aparecem no *log* (não trata ruídos), não detecta escolhas-dependentes e não funciona para processos com atividades repetidas.

Mineração Genética de Processos

- Desenvolvido por Medeiros, Weijters e Aalst em 2005;
- Capaz de detectar construções mais complexas (como escolhas-dependentes) porque a abordagem genética permite realizar uma busca global de dependências entre as atividades;
- Representação interna: matriz causal (representação de redes de Petri na forma matricial);
- Função-objetivo: baseada em dois critérios – completude e precisão;
- Operadores genéticos: incluem/excluem lugares na rede e modificam arcos de entrada e saída das transições;
- Problemas: modelos com atividades repetidas, dificuldade em tratar muitas atividades (grande espaço de busca) e geração de modelos excessivamente generalizados.

Método MARKOV

- Desenvolvido por Cook e Wolf em 1995;
- Combina técnicas estatísticas e algorítmicas;
- Usa modelos de Markov para encontrar a seqüência de eventos mais provável e algoritmicamente converte essas probabilidades em estados e transições entre estados (MEFs);
- A extensão do método para a descoberta de concorrência utiliza métricas (entropia, contagens, periodicidade, etc.);
- Problema: as probabilidades e as métricas só geram um modelo correto quando aplicadas sobre *logs* de sistemas muito bem "comportados".

Metodologia

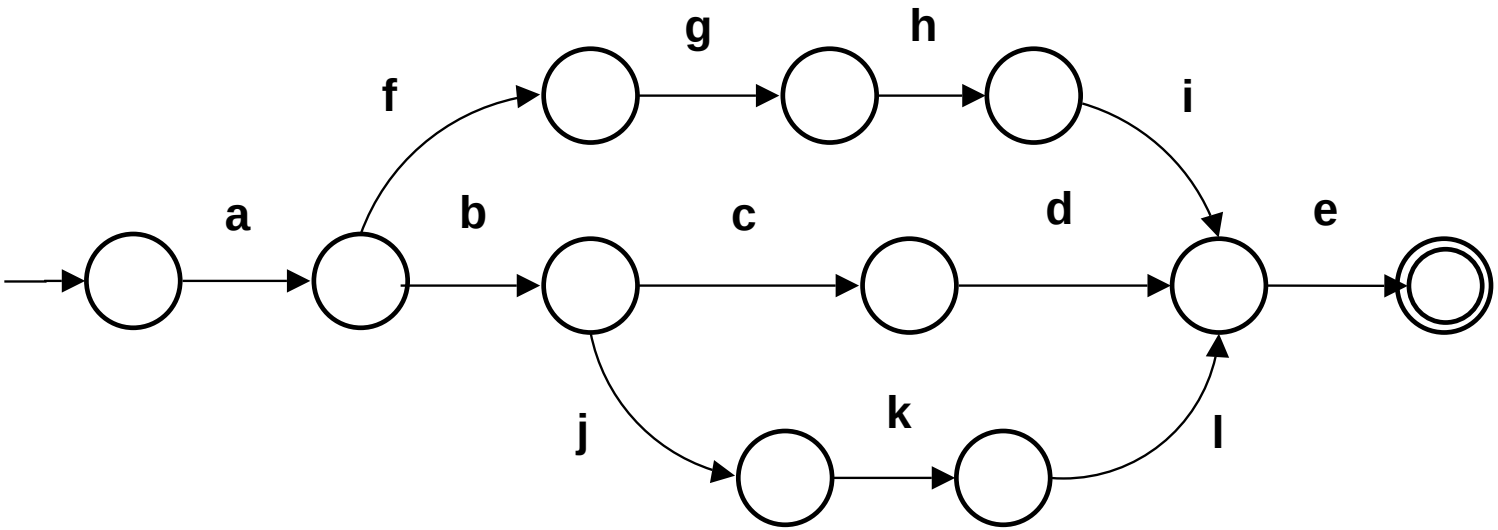
O trabalho foi dividido em duas fases:

1. Construção do “esboço” estrutural do modelo do processo a partir de um *log* de eventos;
2. Refinamento, que dará origem a um modelo estocástico e mais generalizado.

Abordagem para a primeira fase:

- modelos são representados por expressões em álgebra de processos (sem taxas associadas às ações);
- expressões algébricas de processos são representadas por um conjunto de máquinas de estados finitos não-determinísticas (MEFs).

Geração do Esboço Estrutural do Processo



a.((f.g.h.i) + (b.(c.d + j.k.l))).e

Aceita as seqüências: *AFGHIE*, *ABCDE* e *ABJKLE*

Descrição do Algoritmo I

Algoritmo é composto pelos seguintes passos:

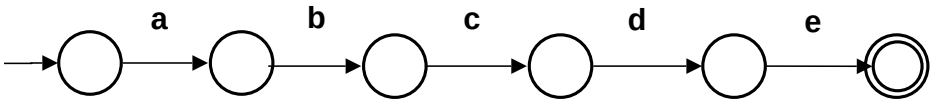
1. cada nova seqüência lida do *log* de entrada é “testada” sobre cada uma das MEFs presentes no conjunto atual de MEFs mantido pelo algoritmo;
2. caso a seqüência possua um prefixo e/ou sufixo comum à alguma das seqüências reconhecidas pelas MEFs, o tamanho do prefixo e/ou sufixo comum deve ser armazenado;
3. após a seqüência ter sido “testada” sobre todas as MEFs do conjunto, seleciona-se a MEF que possui a maior soma dos tamanhos de seu prefixo e sufixo comuns à seqüência;
4. caso não exista uma MEF com prefixo e/ou sufixo comum à seqüência, a MEF trivial da seqüência é criada e adicionada ao conjunto atual de MEFs do algoritmo;

Descrição do Algoritmo II

5. caso exista, a MEF com maior prefixo/sufixo será alterada para passar a reconhecer também a nova seqüência. As possíveis formas de se alterar uma MEF foram divididas em três casos;
6. a expressão algébrica final do processo inferido a partir do *log* é a composição por meio do operador de escolha (“+”) das expressões algébricas associadas a cada uma das MEFs pertencentes ao conjunto gerado pelo algoritmo.

Caso 1: Prefixos em Comum I

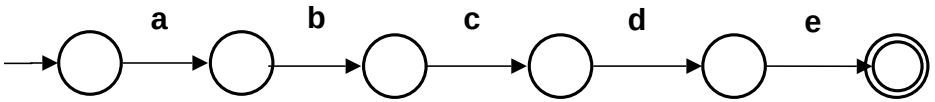
Seqüência *ABCDE*



$a.b.c.d.e$

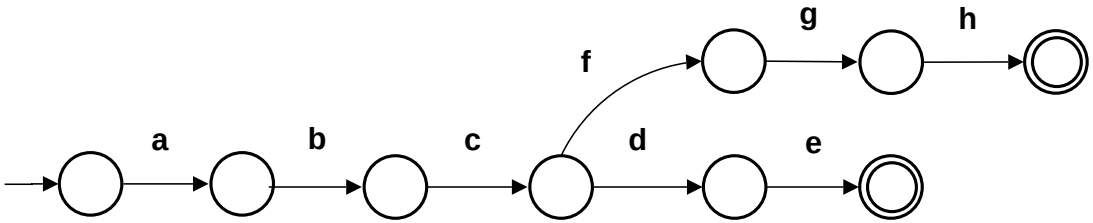
Caso 1: Prefixos em Comum I

Seqüência *ABCDE*



$$a.b.c.d.e$$

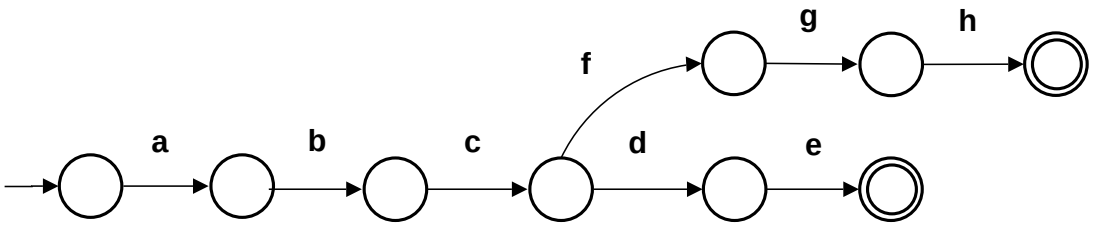
Adição da seqüência *ABCFGH*



$$a.b.c.((f.g.h) + (d.e))$$

Caso 1: Prefixos em Comum II

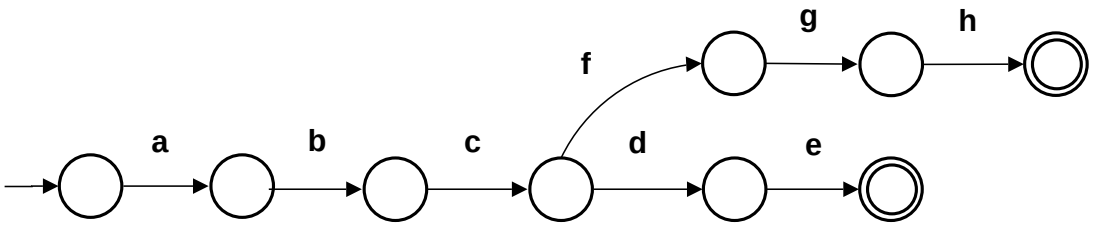
Seqüências *ABCDE* e *ABCFGH*



$$a.b.c.((f.g.h) + (d.e))$$

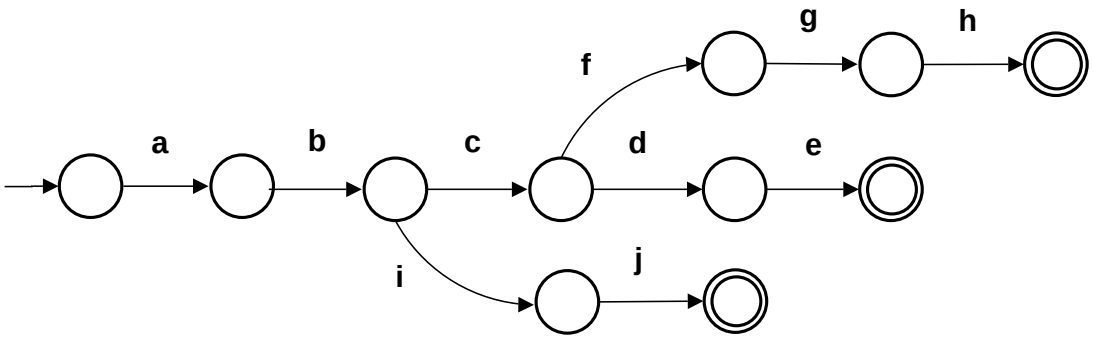
Caso 1: Prefixos em Comum II

Seqüências *ABCDE* e *ABCFGH*



$$a.b.c.((f.g.h) + (d.e))$$

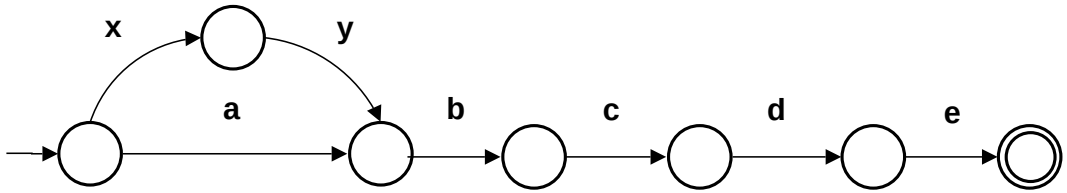
Adição da seqüência *ABIJ*



$$a.b.((i.j) + (c.((f.g.h) + (d.e))))$$

Caso 1: Construção Inválida

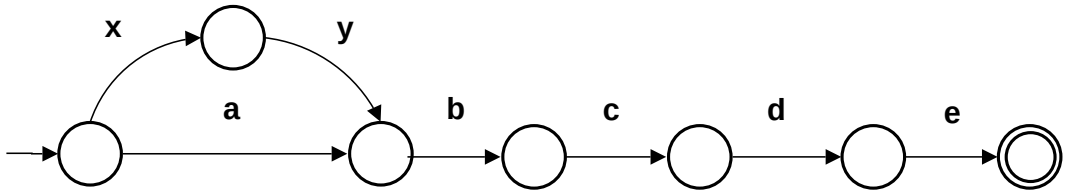
Seqüências *ABCDE* e *XYBCDE*



$$((x.y) + a).b.c.d.e$$

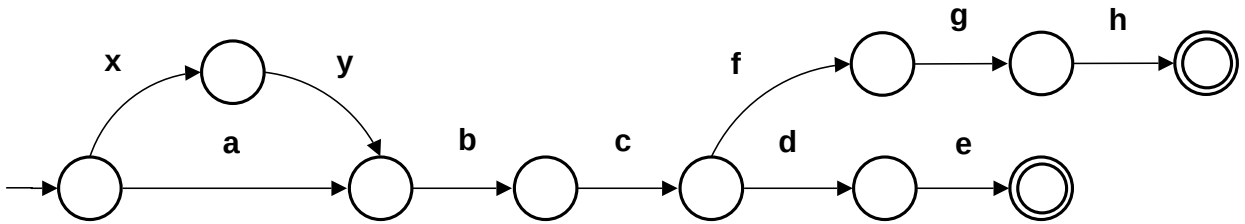
Caso 1: Construção Inválida

Seqüências *ABCDE* e *XYBCDE*



$$((x.y)+a).b.c.d.e$$

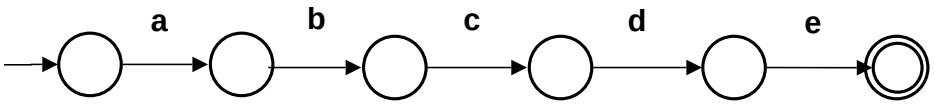
Adição da seqüência *ABCFGH*



$$((x.y)+a).b.c.((f.g.h)+(d.e))$$

Caso 2: Sufixos em Comum I

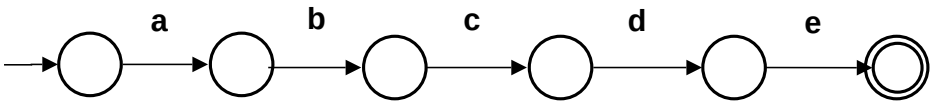
Seqüência *ABCDE*



$a.b.c.d.e$

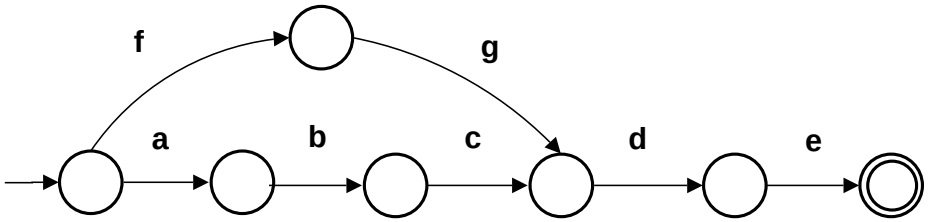
Caso 2: Sufixos em Comum I

Seqüência *ABCDE*



$$a.b.c.d.e$$

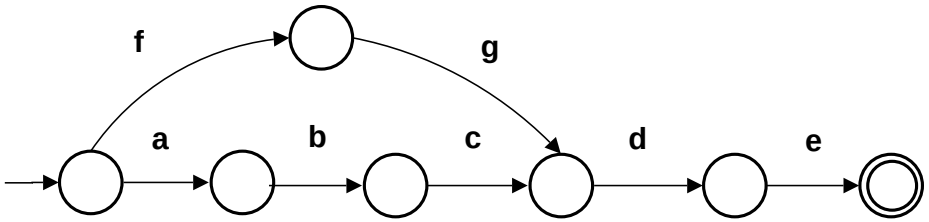
Adição da seqüência *FGDE*



$$((f.g) + (a.b.c)).d.e$$

Caso 2: Sufixos em Comum II

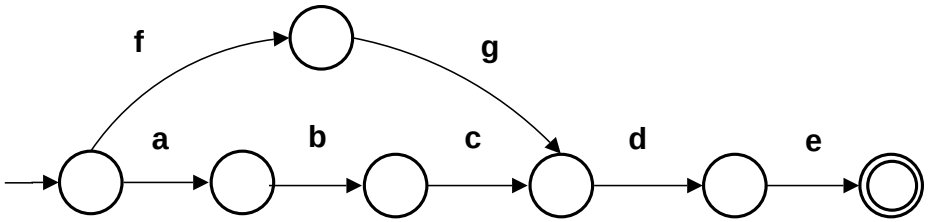
Seqüências *ABCDE* e *FGDE*



$$((f.g) + (a.b.c)).d.e$$

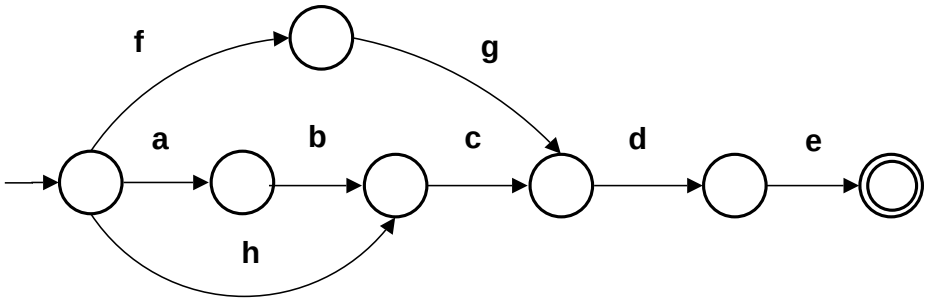
Caso 2: Sufixos em Comum II

Seqüências *ABCDE* e *FGDE*



$$((f.g) + (a.b.c)).d.e$$

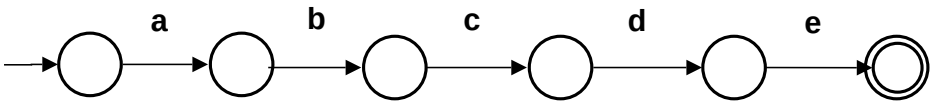
Adição da seqüência *HCDE*



$$((f.g) + (((a.b) + h).c)).d.e$$

Caso 3: Prefixos e Sufixos em Comum I

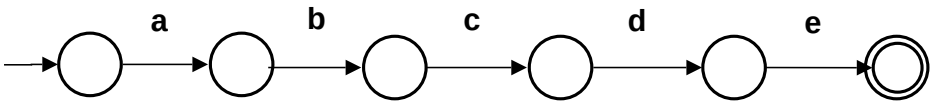
Seqüência *ABCDE*



$a.b.c.d.e$

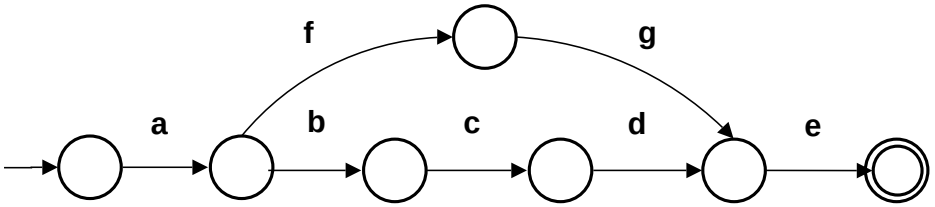
Caso 3: Prefixos e Sufixos em Comum I

Seqüência *ABCDE*



$$a.b.c.d.e$$

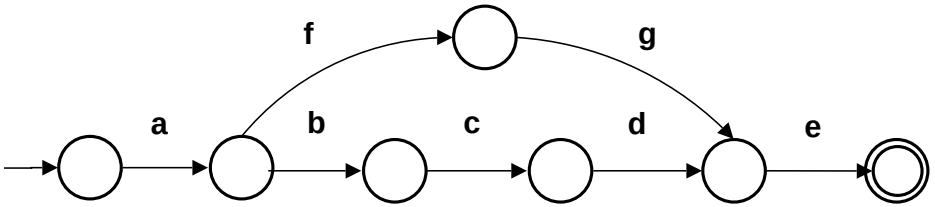
Adição da seqüência *AFGE*



$$a.((f.g)+(b.c.d)).e$$

Caso 3: Prefixos e Sufixos em Comum II

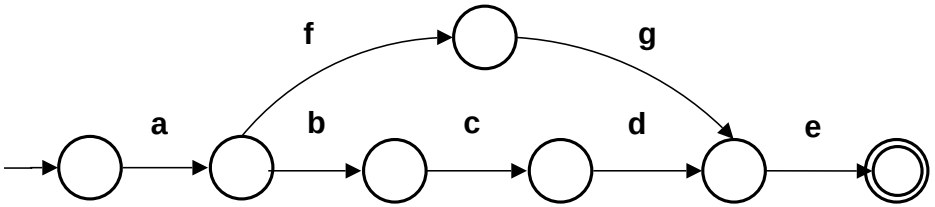
Seqüências *ABCDE* e *AFGE*



$$a.((f.g) + (b.c.d)).e$$

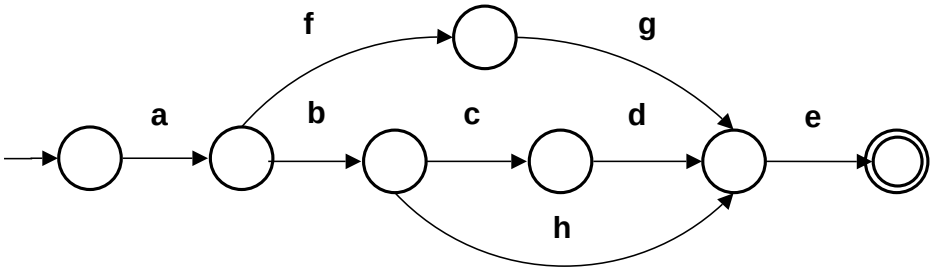
Caso 3: Prefixos e Sufixos em Comum II

Seqüências *ABCDE* e *AFGE*



$$a.((f.g)+(b.c.d)).e$$

Adição da seqüência *ABHE*



$$a.((f.g)+(b.((c.d)+h))).e$$

Atividades I

Evoluir o modelo gerado pelo algoritmo da fase 1.

Abstração do modelo:

1. substituição de seqüências periódicas por ciclos. Ex: $P \stackrel{def}{=} a.b.a.b.a.b \Rightarrow P' \stackrel{def}{=} (a.b) + (a.b.P')$;
2. substituição de ramificações de escolha por ramificações de paralelismo. Ex: $a.b + b.a \Rightarrow a||b$;
3. criação de subprocessos para o reaproveitamento de termos.

Inferência dos parâmetros estocásticos do modelo:

1. descoberta das probabilidades associadas às escolhas;
2. associação das ações às suas respectivas taxas de execução.

Atividades II

Avaliar o algoritmo implementado e os modelos gerados por ele.

Envolve as seguintes atividades:

1. coleta de dados (*logs* de eventos) gerados a partir da execução de sistemas;
2. execução dos testes do algoritmo sobre os dados coletados;
3. definição de métricas para avaliar a conformidade dos modelos gerados $\Rightarrow$ precisão e completude;
4. comparação entre os resultados obtidos e os disponíveis na literatura de mineração de processos.

Cronograma

Atividade	Período
Implementação do método para abstração do modelo	01/2009 – 06/2009
Implementação da inferência dos parâm. estocásticos	04/2009 – 12/2009
Avaliação da implementação e dos resultados obtidos	07/2009 – 03/2010
Outras atividades	
- Integração do algoritmo ao ambiente em desenvolvimento no grupo de BD do IME	01/2010 – 07/2010
- Estudo bibliográfico	01/2009 – 07/2010
- Redação de artigos e relatórios técnicos	04/2009 – 07/2010
- Redação da tese	04/2009 – 07/2010
- Defesa da tese	07/2010

Fim

Slides desta apresentação
`www.ime.usp.br/~kellyrb/quali/slides.pdf`