Análise de Desempenho Baseada em Lógica Linear e *Workflow Net* para Escalonamento de Sistemas de *Workflow*

Lorena Rodrigues Bruno

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2023

Lorena Rodrigues Bruno

Análise de Desempenho Baseada em Lógica Linear e *Workflow Net* para Escalonamento de Sistemas de *Workflow*

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software

Orientador: Stéphane Julia

Uberlândia
2023

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

B898 Bruno, Lorena Rodrigues, 1994-
2023 Análise de desempenho baseada na lógica linear e workflow net para escalonamento de sistemas de workflow [recurso eletrônico] / Lorena Rodrigues Bruno. - 2023.

Orientador: Stéphane Julia.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2023.160
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Julia, Stéphane,1969-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado 1/2023, PPGCO				
Data:	16 de fevereiro de 2023	Hora de início:	09:30	Hora de encerramento:	12:30
Matrícula do Discente:	12012CCP006				
Nome do Discente:	Lorena Rodrigues Bruno				
Título do Trabalho:	Análise de Desempenho baseada em Lógica Linear e Workflow-net para Escalonamento de Sistemas de Gerenciamento de Workflow				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Engenharia de Software				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Márcia Aparecida Fernandes - FACOM/UFU, José Reinaldo Silva - USP e Stéphane Julia - FACOM/UFU, orientador da candidata.

Os examinadores participaram desde as seguintes localidades: José Reinaldo Silva - São Paulo-SP, Márcia Aparecida Fernandes e Stéphane Julia - Uberlândia/MG. A discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Stéphane Julia, apresentou a Comissão Examinadora e a candidata, agradeceu a presença do público, e concedeu a Discente a palavra para a exposição do seu trabalho. A duração da apresentação da Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir a candidata. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando a candidata:

Aprovada

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Stéphane Julia**, **Professor(a) do Magistério Superior**, em 17/02/2023, às 10:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Márcia Aparecida Fernandes**, **Professor(a) do Magistério Superior**, em 17/02/2023, às 12:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Jose Reinaldo Silva**, **Usuário Externo**, em 17/02/2023, às 20:10, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4266927** e o código CRC **05669C0D**.

Referência: Processo nº 23117.010885/2023-48 SEI nº 4266927

Dedico este trabalho à minha mãe Maria José, ao meu pai Reginaldo, à minha irmã Renata que sempre me apoiaram.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado saúde, força e por ter me guiado durante toda minha jornada, permitindo que eu chegasse até aqui.

Aos meus pais pelo amor, incentivo e apoio que sempre me deram. Por sempre acreditarem em mim e não medirem esforços para que eu tivesse uma boa formação, tanto pessoal quanto profissional e acadêmica.

À minha irmã Renata pelo carinho, apoio e incentivo.

Aos meus amigos, pelo auxilio e compreensão durante toda essa jornada.

Ao meu orientador Stéphane Julia, pela paciência, prontidão e confiança. Por me auxiliar e guiar durante a execução deste trabalho, seu apoio foi essencial.

"A única maneira de fazer algo excelente é amar o que você faz. Se você ainda não a encontrou, continue procurando. Não se acomode."

(Steve Jobs)

Resumo

O modelo de um Sistema de Gerenciamento de Workflow (SGW) deve descrever as restrições de tempo dos recursos sobre as atividades do processo de negócio correspondente. Em geral, fenômenos temporais incluem atrasos na execução das atividades, limites para a ocorrência de intervalos válidos relacionados às atividades, limites para intervalos válidos relacionados as recursos (limites para o ciclo de vida de um recurso), limites na duração da execução de processos, intervalo de distância entre duas atividades, entre outros. Nestre trabalho, um modelo de *Workflow net* com adição de intervalos de tempo associados à duração das atividades e tempos de espera são apresentados. Para definir os intervalos mínimo e máximo para a execução das atividades, um mecanismo de propagação com restrição de tempo baseado no cálculo de sequentes da Lógica Linear e em datas simbólicas é proposto. Utilizando Colored Petri Nets (CPNs)(Rede de Petri Colorida) hierárquicas é possível separar o modelo do proceso do modelo de recurso e definir formalmente os mecanismos de comunicação entre esses dois modelos. Cada ficha da rede de Petri Colorida será capaz de carregar as informações de tempo para cada caso, assim como o intervalo de tempo que cada atividade terá que respeitar para que o processo seja finalizado dentro do prazo proposto. Tal marcação temporal pode ser representada por um conjunto de cores do modelo CPN e será utilizado para monitorar a execução do processo a fim de encontrar a quantidade certa de recursos e a melhor técnica de escalonamento envolvidos na execução das atividades. A simulação da CPN com informações de tempo permite então estimar a quantidade de casos que respeitam as restrições de tempo do processo (prazo de entrega), além de melhorar o planejamento de recursos em um SGW.

Palavras-chave: Rede de Petri. *Workflow net*. Lógica Linear. Propagação para frente. Propagação para trás. CPN Tools.

Abstract

The model of a Workflow Management System should describe the time constraints of resources over the activities of the corresponding business process. In general, typical temporal phenomena include activity execution delays, limits to the occurrence of valid intervals over the activities, limits to valid intervals over resources (limit to resources life cycle), limits to duration of process execution, time distance between two activities, etc. In this study, a Workflow net model incremented with time intervals to describe the duration of activities and waiting times is presented. To define the execution of activities minimum and maximum intervals, a time constraint propagation mechanism based on the sequent calculus of Linear Logic and on symbolic dates is proposed. Using Colored Petri nets, it is possible to separate the process model from the resource model and formally define the communication mechanisms between the two models. Each token from the Colored Petri net will then be able to carry the time information of each case, such as the start and end time of each activity the case will have to perform to complete the Workflow process. Such temporal data can be represented as a set of colors from the CPN model and will be used to monitor the execution of the process in order to find the right amount of resources and the best scheduling technique involved in the execution of activities. The simulation of the CPN with time information then allows estimating the percentage of cases that respect the time constraints of the process (deadline delivery dates), in addition with improving resource planning in Workflow Management System.

Keywords: Petri Net. Workflow Net. Linear Logic. Forward Propagation. Backward Propagation. CPN Tools.

Lista de ilustrações

Lista de tabelas

Lista de siglas

CPN Colored Petri Net

FIFO First in First Out

SGW Sistema de Gerenciamento de Workflow

WF-net Workflow net

Sumário

Capítulo 1

Introdução

Workflow Management Systems são sistemas que gerenciam a execução dos processos de casos a serem tratados dentro das organizações. Um processo consiste em um conjunto de tarefas a serem realizadas de acordo com um conjunto de condições que determinam a ordem de execução dessas tarefas. A tradução do termo *workflow* em inglês é fluxo de trabalho, porém o termo também pode ser definido como qualquer conjunto de atividades executadas de forma coordenada, em série ou em paralelo, por membros de um mesmo grupo de trabalho, visando um objetivo comum. O termo *workflow* é utilizado em (AALST; HEE; HEE, 2004) como sinônimo de fluxo de trabalho.

Em sistemas de gerenciamento de *workflow*, os processos são modelados através de fluxos de trabalho que representam as sequências de atividades que precisam ser executadas em uma organização para tratar casos específicos e atingir uma meta previamente estabelecida. Além disso, é importante definir também os responsáveis por cada atividade e os recursos envolvidos, de acordo com as necessidades descritas em cada processo (AALST; HEE; HEE, 2004). Os sistemas de gerenciamento de *workflow* são ferramentas importantes dentro das organizações pois quanto mais rápido elas se adaptam às mudanças, maiores são suas chances de se destacar no mercado e melhorar sua proditividade, e consequentemente, sua lucratividade; são ferramentas que ajudam a reestruturar os processos internos, sejam eles administrativos ou técnicos.

Várias linguagens de modelagem podem ser utilizadas no contexto de modelagem de processos de negócio porém, dependendo do problema a ser resolvido, pode ser necessário um modelo de processos com uma semântica formal, que é o caso quando se pretende aplicar uma técnica de verificação ou validação dos requisitos expressos pelos processos modelados (AALST; HEE; HEE, 2004). Neste caso, uma solução mais adequada seria a modelagem utilizando as redes de Petri, que possuem uma representação gráfica, funcionam como linguagem de comunicação entre especialistas de diversas áreas, permitem a descrição de aspectos estáticos e dinâmicos do sistema a ser representado, e ainda possuem um formalismo matemático que possibilita a utilização de importantes métodos de análise (MURATA, 1989). As redes de Petri que modelam os processos de negócio foram

definidas por (AALST; HEE; HEE, 2004) e são chamadas *Workflow nets*. Vários trabalhos passaram a considerar esse modelo formal e específico para a modelagem dos processos de negócio e suas abordagens.

As tarefas de um processo de negócio geralmente dependem de recursos para serem executadas. Portanto, é necessário quantificar os recuros necessários para a execução de um *workflow* a fim de evitar que muitas tarefas sejam atribuídas ao mesmo recurso e, como resultado, o recurso se torne um gargalo para o sistema.

A maioria dos modelos existentes foca na representação do processo e não considera as características importantes dos sistemas de gerenciamento de *workflow*, como os mecanismos de alocação de recursos, que muitas vezes são representados apenas de maneira informal. Também, a representação de recursos de um processo por fichas discretas pode não ser suficiente para representar, por exemplo, recursos como funcionários humanos que são capazes de tratar simultaneamente vários casos em um único dia. Em particular, o comportamento dinâmico de alguns sistemas exige o tratamento do problema de escalonamento dos recursos compartilhados para um melhor controle dos fluxos de trabalho (uma simples política de tratamento o quanto antes nas atividades, a medida que os recursos envolvidos se tornam disponíveis, respeitando um funcionamento baseado em simples filas de espera, pode comprometer a qualidade de serviço em relação ao respeito dos prazos de conclusão dos casos do processo).

O problema do escalonamento (ESQUIROL; HUGUET; LOPEZ, 1995) consiste em organizar no tempo a realização de tarefas, considerando restrições temporais e restrições de utilização de recursos compartilhados, necessários à execução das tarefas. O problema do escalonamento pode ser comparado à atividade de execução de um cenário, ou seja, simulação do comportamento de um sistema em tempo real. Nestes casos, vários cenários podem ser executados de forma simultânea e podem ocorrer conflitos se um mesmo recurso não-preemptivo for solicitado por duas atividades de diferentes cenários.

A principal diferença entre o problema do escalonamento em um sistema de produção e em um sistema de gerenciamento de *workflow* é a natureza dos recursos empregados nas atividades. No caso de sistemas de produção, os recuros representam equipamentos físicos, o que permite que sejam representados por fichas simples de uma rede de Petri. Já nos sistema de gerenciamento de *workflow*, os recursos podem representar tanto equipamentos físicos quanto funcionários humanos. Funcionários humanos podem tratar diversas atividades simultaneamente e por isso não podem ser representados por simples fichas nos lugares de uma rede de Petri ordinária (JULIA; OLIVEIRA; VALETTE, 2008).

Em casos nos quais os recursos não podem ser representados por fichas puramente discretas (que representariam a execução de uma única atividade), a solução é utilizar a representação do recurso por um número real que represente sua disponibilidade, de forma que será igual a 100% quando o recurso estiver completamente disponível (JULIA; OLIVEIRA; VALETTE, 2008). Utilizaremos neste trabalho representações discretas e

híbridas dos recursos (recursos discretos e contínuos) que podem ser modeladas utilizando redes de Petri discretas e redes de Petri Contínuas (DAVID; ALLA, 2010).

Muitas técnicas de análise podem ser consideradas para um processo de *workflow*; em particular, duas grandes abordagens são apresentadas em (AALST; HEE; HEE, 2004): qualitativa e quantitativa. Análise qualitativa está associada à corretude lógica de um processo de *workflow*, enquanto a análise quantitativa está associada aos requisitos de performance e capacidade. A propriedade *Soundness* é um critério importante a ser satisfeito quando se refere à processos de *workflow*, e sua prova está relacionada a uma análise qualitativa.

A abordagem proposta é baseada no cálculo do sequente da Lógica Linear para verificar propriedades quantitativas e qualitativas de *Workflow nets*. A análise qualitativa é baseada na prova de corretude lógica (*Soundness*) definida para *Workflow nets*. A análise quantitativa é associada ao planejamento de recursos de cada atividade mapeada para o processo utilizando *Workflow nets* temporizadas.

Será apresentada uma abordagem baseada na Lógica Linear para calcular fórmulas simbólicas baseadas nos operadores $(max, +)$ e $(min, -)$ que expressam os possíveis intervalos de visibilidade nos quais a ficha (que representa um caso em uma *Workflow net*) deve iniciar e finalizar cada atividade. Tais expressões simbólicas são utilizadas para definir as restrições numéricas de tempo associadas a cada caso em específico, tratado pelo processo de *workflow* e serão considerados no processo de planejamento e escalonamento de recursos associados a cada atividade do processo.

Os resutlados obtidos são considerados para modelar, simular e monitorar um processo de *workflow* utilizando redes de Petri coloridas hierárquicas na ferramenta CPN Tools. O objetivo é estimar o número de casos que respeitam os limites de tempo determinados pelos intervalos de visibilidade especificados para cada atividade.

1.1 Motivação

O problema principal a ser tratado neste trabalho é o cálculo dos intervalos de visibilidades para cada atividade de um processo e, a partir dos dados obtidos, encontrar a melhor forma de alocação e escalonamento de recursos de forma que os casos (representados pelas fichas nas *Workflow nets*) respeitem esses intervalos de tempo. Processos de negócio distintos podem compartilhar recursos comuns. Por exemplo, um funcionário pode atuar em diversas atividades que se encontram em diversos processos não diretamente relacionados entre si. Além disso, na execução das atividades de processos de negócio num ambiente real, dependendo da urgência, os recursos podem ser sobrecarregados ou os prazos de entrega podem ser alterados (atrasados). Para obter a melhor utilização dos recursos, as cargas de trabalho devem ser gerenciadas para evitar a sobrecarga em alguns recursos, e/ou taxas de ocupação baixa de certos funcionários.

Dessa forma, deseja-se modelar um processo utilizando uma *Workflow net* respeitando os intervalos de visibilidade calculados para este modelos e em seguida, simular o comportamento dos casos utilizando diferentes formas de alocação de recursos e de escalonamento.

1.2 Objetivos e Desafios da Pesquisa

O objetivo deste trabalho é propor um mecanismo de propagação com restrição de tempo que permita encontrar os intervalos de visibilidade para as atividades de um processo, de forma que o prazo limite para execução do processo como um todo seja respeitado. Após calculados os intervalos de visibilidade simbólicos, deve-se calcular os intervalos numéricos para um processo específico escolhido como objeto de estudo. A partir desses dados, será realizada uma análise de atraso no disparo das atividades para verificar como isso afeta o processo como um todo e quais as melhores opções para dimuir esses atrasos utilizando mecanismos de alocação de recursos e políticas de escalonamento.

Além disso, deve-se propor um novo modelo de alocação de recurso para dimensionamento e gestão de recursos em processos de negócios. Baseado em tais mecanismos, pretende-se aplicar heurísticas de distribuição de recursos baseadas no cálculo dos sequentes da Lógica Linear para tratar o problema de escalonamento em tempo de execução (sem mecanismo de retrocesso) das atividades dos SGW.

Deverão ser realizados testes através de simulações no CPN Tools de cenários de validação de diversas propostas de políticas de escalonamento, cada uma aprensentando um mecanismos de alocação de recursos diferente (discreto e/ou contínuo, com disparo das atividades o mais cedo possível ou respeitando as restrições de prazos indicados pelos resultados do cálculo dos sequentes da Lógica Linear).

Para que esses objetivos sejam alcançados, é preciso:

a) propor um mecanismo de propagação com restrição de tempo que possibilite o cálculo dos intervalos de visibilidade simbólicos para cada atividade do processo escolhido como objeto de estudo ;

b) calcular os intervalos de visibilidade simbólicos com base no cálculo de sequentes da Lógica Linear ;

c) calcular os intervalos de visibilidade numéricos para o processo escolhido como objeto de estudo;

d) implementar o processo escolhido na ferramenta CPN Tools;

e) analisar os resultados relcionados aos prazos determinados pelos intervalos de visibilidade tanto para o sistema como um todo quanto para as atividades;

f) criar modelos de alocação de recursos baseados em Redes de Petri discetas e contínuas que permitem a implementação de heurísticas diversas para teste e validação

de solução de escalonamento em tempo de execução de atividades em processos de negócios;

g) propor heurística de escalonamento em tempo de execução (sem mecanismo de retrocesso) baseado no cálculo dos sequentes da Lógica Linear para obtenção de solução de escalonamento que respeite as restrições de prazo de entrega dos casos tratados pelos processos de negócio;

h) implementar as soluções propostas na ferramenta de modelagem, análise e simulação de Redes de Petri Coloridas *CPN Tools* para realizar um estudo comparativo entre políticas de escalonamento baseado em sistemas de filas de espera tradicionais (com disparo ao mais cedo das atividades tão logo os recursos estiverem disponíveis) e políticas de escalonamento inteligentes (não necessariamente com disparos ao mais cedo das atividades) que consideram heurísticas baseadas no cálculo dos sequentes da Lógica Linear.

1.3 Hipótese

As hipóteses desta dissertação consistem em:

a) o cálculo do sequente da Lógica Linear aplicado aos roteiros das *Workflow nets* permite expressar de forma simbólica as datas de início ao mais cedo e ao mais tarde das atividades a serem executadas e fornece informações fundamentais para estabelecer heurísticas de escalonamento que permitem o respeito dos prazos dos casos tratados nos processos de negócios;

b) a modelagem de um processo com mecanismos de alocação de recursos torna o modelo mais próximo à realidade da gestão de recursos dos sistemas de gerenciamento de processos de negócios e melhora o respeito dos prazos de execução das atividades de um processo e do processo como um todo;

c) a modelagem de um processo utilizando políticas de escalonamento pode melhorar o respeito dos prazos de execução das atividades de um processo e do processo como um todo;

d) a ferramenta de modelagem, análise e simulação de Redes de Petri Coloridas *CPN Tools* permite implementar, simular e validar políticas de escalonamento quando o modelo de especificação do sistema são *Workflow nets* com mecanismo de recursos discretos e contínuos.

1.4 Contribuições

As contribuições desta pesquisa serão:

a) propor um mecanismo de propagação com restrição de tempo para encontrar os intervalos de visibilidade simbólicos para as atividades de um processo por meio do cálculo de sequente da Lógica Linear;

b) uma proposta de modelagem de mecanismo de alocação de recursos que simula o comportamento flexível dos funcionários humanos envolvidos na execução das atividades em processos de negócios;

c) a partir dos intervalos simbólicos, calcular os intervalos de visibilidade numéricos para cada atividade do processo escolhido para ser modelado;

d) uma proposta de validação de soluções para o planejamento, dimensionamento e escalonamento de recursos em sistemas de gerenciamento de processos de negócios.

Pretende-se com a realização desta pesquisa contribuir para a área de processos de *Workflow*, trazendo uma nova proposta de modelagem de processos que possuem recursos compartilhados. A ideia é produzir modelos de alocação de recursos próximos da realidade da gestão de recursos em sistemas administrativos.

1.5 Organização da Dissertação

Este trabalho está dividido em seis capítulos, organizados no formato a seguir.

No Capítulo 2 é apresentada a fundamentação teórica necessária para melhor compreensão deste trabalho. Na Seção 2.1 são apresentados os fundamentos teóricos relacionados à definições básicas sobre redes de Petri e algumas de suas extensões, *Workflow net* e Lógica Linear. Na Seção 2.2 são apresentados os trabalhos relacionados.

No Capítulo 3 são apresentados dois mecanismos de propagação com restrição de tempo diferentes para um processo,; um com evolução para frente, e outro com evolução para trás. Cada um deles é utilizado para o cálculo de um dos limites do intervalos de visibilidade das atividades (intervalos de datas), sendo o mecanismo para frente responsável pelo cáculo do limite mínimo e o mecanismo para trás, do limite máximo dos intervalos.

O Capítulo 4 trata de análise e experimentos para o dimensionamento de recursos, de forma que na Seção 4.1 são apresentados os mecanismos de alocação de recursos considerados para este trabalho. Na Seção 4.2 é apresentada as implementação de uma *Workflow net* com recursos na ferramenta CPN Tools utilizando os diferentes recursos apresentados na Seção 4.1. Na Seção 4.3 é apresentada uma análise de atraso a partir dos dados obtidos pela simulação do modelo implementado.

O Capítulo 5 trata de análise e experimentos para políticas de escalonamento. Na Seção 5.1 é apresentada a definição básica do problema de escalonamento. Na Seção 5.2 é apresentada a implementação de uma *Workflow net* utilizando filas, e na Seção 5.3, os resultados das simulações para diferentes políticas de filas.

Finalmente, no Capítulo 6 são apresentadas as considerações finais, bem como as principais contribuições (em particular as produções bibliográficas que foram publicadas e apresentadas durante o trabalho de dissertação) e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Nesse capítulo é apresentada a fundamentação teórica necessária para a compreensão deste trabalho bem como os trabalhos relacionados. Este capítulo está dividido da seguinte forma: na seção 2.1 são apresentadas as principais definições e propriedades relacionadas às Redes de Petri, Redes de Petri Coloridas, CPN Tools, *Workflow net* e Lógica Linear. Os Trabalhos Relacionados são apresentados na seção 2.2.

2.1 Fundamentos Teóricos

Esta seção apresenta os conceitos teóricos relacionados às redes de Petri (2.1.1), às Redes de Petri com Tempo (2.1.2), às Rede de Petri Coloridas (2.1.3), às *Workflow nets* (2.1.4) e à Lógica Linear (2.1.5).

2.1.1 Redes de Petri

A teoria das redes de Petri foi proposta inicialmente por Carl Adam Petri em sua tese (PETRI, 1962) apresentada em 1962 na Universidade de Darmstadt na Alemanha. Apesar de ser considerada um teoria relativamente jovem, ela se adapta bem a um grande número de aplicações em que as noções de eventos e evolução simultânea são importantes (CARDOSO; VALETTE, 1997).

As redes de Petri são consideradas como uma ferramenta gráfica e matemática de representação formal que permite a modelagem, análise e o controle de sistemas orientados a eventos discretos (sistemas nos quais as mudanças de estado ocorrem em instantes precisos de tempo) que comportam atividades paralelas, concorrentes e assíncronas (MURATA, 1989). Entre os pontos mais importantes das redes de Petri estão seu formalismo matemático, característica que permite realizar uma análise precisa dos modelos para verificar propriedades estruturais e comportamentais.

Os elementos básicos que fomam as redes de Petri são três: lugar, transição e ficha.

a) lugar: representado por um círculo, pode ser interpretado, entre outras abordagens, como uma condição, um procedimento, um estado de espera, um conjunto de recursos. É considerado o elemento passivo do modelo;

b) transição: representada por uma barra ou retângulo, está associada aos eventos que ocorrem no sistema. É o elemento ativo do modelo;

c) ficha: representada por um ponto em um lugar, funciona como um indicador para representar que a condição associada ao lugar é verificada, ou verdadeira.

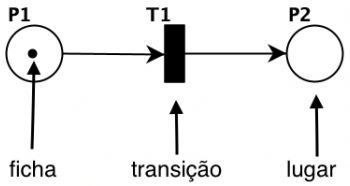


Figura 1 – Rede de Petri.

A Figura 1 ilustra um exemplo de uma rede de Petri e seus elementos básicos. Formalmente, as redes de Petri são definidas da seguinte forma (MURATA, 1989; CARDOSO; VALETTE, 1997):

Definicão 1 (Rede de Petri)**.** *Uma Rede de Petri é formalmente definida como uma quádrupla*

$$R =< P, T, Pre, Pos > \tag{1}$$

onde:

a) P é um conjunto finito de lugares de dimensão n;

b) T é um conjunto finito de transições de dimensão n;

c) $Pre : P \times T \rightarrow \mathbb{N}$ é uma relação de entrada que define arcos que ligam os lugares as transições;

d) $Pos : T \times P \rightarrow \mathbb{N}$ é uma relação de saída que define arcos que ligam as transições aos lugares;

No caso das redes de Petri marcadas, a definição formal é a seguinte:

Definicão 2 (Rede de Petri Marcada)**.** *Uma rede marcada N é uma dupla*

$$N =< R, M > \tag{2}$$

onde:

a) R é uma rede de Petri,

b) M é a marcação inicial dada pela aplicação $M : P \to \mathbb{N}$

$M(p)$ representa o número de fichas contidas no lugar p. A marcação M é a distribuição das fichas nos lugares, sendo representada por um vetor coluna cuja dimensão é o número de lugares e os elementos são $M(p)$.

A Figura 2 mostra um exemplo de rede de Petri marcada onde a marcação é dada por $M^T = [1 \;\; 0 \;\; 3 \;\; 0 \;\; 1]$ (M^T é o vetor transposto).

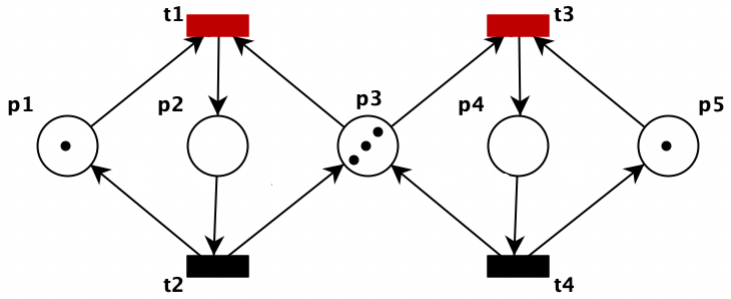


Figura 2 – Exemplo de Rede de Petri Marcada.

Formalmente, a evolução dinâmica de uma rede de Petri é dada pelas seguintes definições:

Definição 3 (Transição habilitada)**.** *Uma transição t está habilitada ou sensibilizada se, e somente se:*

$$\forall p \in P, \quad M(p) \geq Pre(p,t) \tag{3}$$

Isto é, uma transição está habilitada se o número de fichas em cada um de seus lugares de entrada for maior ou igual ao peso do arco que liga este lugar à transição.

Definição 4 (Disparo de transição)**.** *Se t está habilitada por uma marcação M, uma nova marcação M′ é obtida através do disparo de t de maneira que:*

$$\forall p \in P, \quad M'(p) = M(p) - Pre(p,t) + Post(p,t) \tag{4}$$

Isto é, o disparo de uma transição consiste no consumo de fichas de cada um de seus lugares de entrada e produção de fichas em cada um dos seus lugares de saída. Um exemplo de disparo de transição é dado na Figura 3, no qual a transição $T1$ está habilitada, já que existe pelo menos uma ficha em cada um de seus lugares de entrada. Após o disparo de $T1$, é retirada 1 ficha de cada um de seus lugares de entrada ($P1$ e $P2$) e produzida uma ficha em cada um de seus lugares de saída ($P3$ e $P4$).

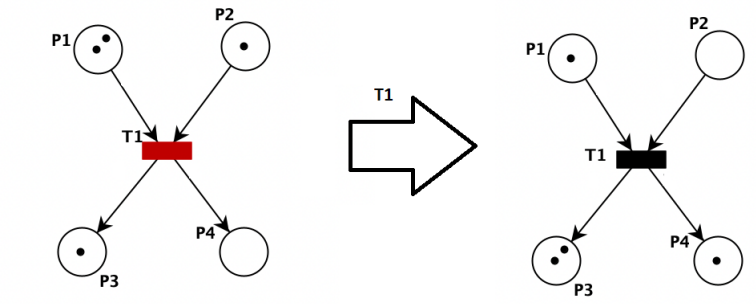


Figura 3 – Exemplo de Disparo de uma Transição.

2.1.2 Redes de Petri e a Representação do Tempo

As redes de Petri clássicas não permitem a modelagem do tempo de forma explícita, apresentando as datas de ocorrência de início e fim das operações. Porém essas redes podem ser estendidas em relação ao tempo, já que informações temporais são úteis para sequenciar eventos, comparar suas durações, bem como determinar o intervalo existente entre elas. Dessa forma é possível representar e analisar problemas cujas atividades relacionadas dependem da relação de tempo produzida entre elas. Vários trabalhos foram realizados no sentido de utilizar explicitamente o tempo como parâmetro contínuo e quantificável em redes de Petri.

Com a adição de tempo, o indeterminismo com relação ao disparo de uma transição é, de certa forma, reuzido já que a informação temporal acrescenta uma nova relação de ordem entre os disparos das transições. Além disso, outro aspecto importante é a forma como o tempo é interpretado . Este aspecto é denominado semântica do tempo e, para uma rede de Petri, existem basicamente duas semânticas distintas:

a) tempo de sensibilização: determina o tempo que deve se passar entre o momento no qual a transição é habilitada (ou sensibilizada) até o momento de seu disparo;

b) tempo de disparo: o de disparo de uma transição ocorre em três fases: retirada da marca, disparo anatômico e colocação da marca.

Outro aspecto importante é quanto a efetivação do disparo de uma transição habilitada em relação ao tempo. Para alguns sistemas, o disparo de uma transição habilitada deve ocorrer em determinada data. Nestes casos, a semântica do tempo é denominada semântica forte, em oposição à semântica fraca, na qual o disparo de uma transição habilitada não é obrigatório, porém, se ocorrer, deve acontecer também em uma determinada data.

As primeiras redes de Petri com interferência do tempo foram apresentadas por Ramchandani (1974) e Merlin (1974), em suas teses de doutorado. Desde então, um grande número de diferentes redes de Petri com tempo tem sido proposto na literatura. Geralmente, os modelos se diferenciam em aspectos, tais como: tipo de temporização, localização

da restrição temporal e propriedade da restrição (CERONE; MAGGIOLO-SCHETTINI, 1999).

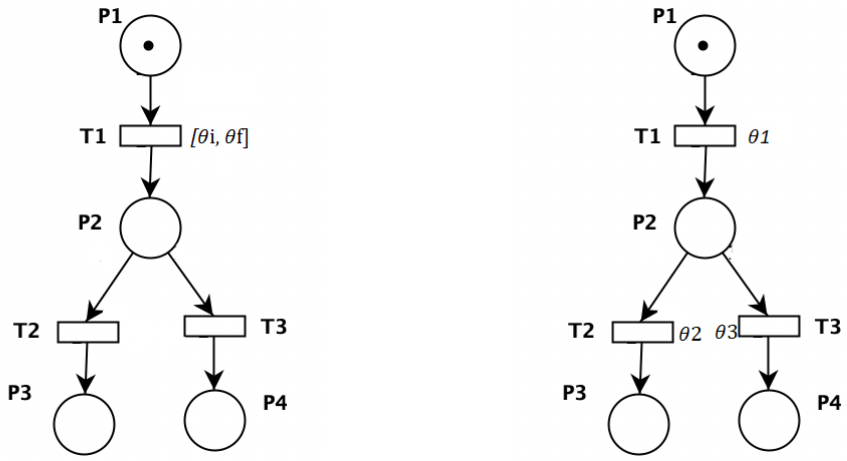
Entre outras abordagens, as extensões temporais da rede de Petri podem ser agrupadas nas seguintes categorias:

a) Redes de Petri Temporais: A restrição temporal é um intervalo de tempo associado a cada transição. Foram inicialmente usadas na descrição de protocolos de comunicação, entretanto, seu campo de aplicação tem se ampliado para áreas tais como: manufatura, sistemas em tempo real, validação e verificação de sistemas (BERTHOMIEU; MENASCHE, 1983; CERONE; MAGGIOLO-SCHETTINI, 1999; BOWDEN, 2000; BERARD et al., 2013).

b) Redes de Petri Temporizadas: Permitem que transições ou lugares retenham marcas durante um intervalo de tempo. São comumente usadas na análise de sistemas de manufatura (CELKO, 1982; TSAI; YANG; CHANG, 1995).

A Figura 4 mostra dois exemplos de extensões temporais de redes de Petri. Na Figura 4a é possível observar que existe um intervalo de tempo $[\theta_i, \theta_f]$ associado à transição $T1$. θ_i indica a duração mínima de sensibilização de uma transição antes do disparo, enquanto que θ_f permite calcular a duração máxima de sensibilização. Dessa forma, a transição deve ser disparada dentro desse intervalo de tempo.

Na Figura 4b, exitem durações de tempo θ_1, θ_2 e θ_3 associado a cada uma das 3 transições da rede, $T1$, $T2$ e $T3$ respectivamente. Portanto, o disparo não é instantâneo, mas possui uma duração, que só tem sentido se a transição é interpretada como uma atividade e não como um evento instantâneo.



(a) Rede de Petri Temporal. (b) Rede de Petri t-Temporizada.

Figura 4 – Exemplos de extensões temporais Redes de Petri.

2.1.3 Redes de Petri Colorida

As redes de Petri clássicas permitem a representação do paralelismo e sincronização; dessa forma, são apropriadas para modelagem de sistemas distribuídos. Em contrapartida, quando as redes de Petri clássicas são utilizadas para a modelagem de processos, os sistemas grandes e complexos se tornam um problema complicado. Dessa forma, muitas extensões dos modelos básicos de redes de Petri surgiram da necessidade de representar esses sistemas complexos.

Em um sistema modelado, as fichas em redes de Petri geralmente representam objetos ou recursos (AALST, 1992). Esses objetos podem ter atributos que nem sempre são facilmente representados por uma simples ficha de uma rede de Petri. Jensen (1981) definiu o que hoje se conhece por rede de Petri Colorida, ou CPN para permitir a modelagem e análise de sistemas complexos.

Uma CPN pertence à classe de redes de Petri de alto nível, que são caracterizadas pela combinação de redes de Petri com linguagens de programação apresentadas como um modelo teórico para representação de sistemas concorrentes. A rede de Petri fornece a base formal para modelar concorrência e sincronização enquanto que a linguagem de programação funcional pode ser usada para modelar a manipulação de dados e criação de modelos compactos e parametrizáveis.

Numa CPN, um lugar comporta fichas que devem respeitar a cor (domínio) associada a ele. Essas cores não significam apenas cores ou padrões, representando tipos de dados complexos. Tais fichas são representada por estruturas de dados que podem conter informações, permitindo aos arcos realizar operações sobre elas (JENSEN; KRISTENSEN, 2009). As expressões dos arcos podem conter constantes, variáveis, funções e operações definidas nas declarações com o intuito de manipular as informações contidas nas fichas. Dessa forma, diferente processos ou recursos podem ser representados em uma mesma estrutura gráfica.

A CPN é composta por três partes: estrutura, inscrições e declarações. A estrutura é uma rede de Petri; as inscrições são associadas aos lugares, transições e arcos; e as declarações são tipos, funções, operações e variáveis (GORGÔNIO et al., 2001). A definição formal de uma CPN é dada a seguir (JENSEN, 1981; CARDOSO; VALETTE, 1997):

Definicão 5 (Rede de Petri Colorida)**.** *Uma rede de Petri Colorida associada a uma marcação inicial é uma sêxtupla*

$$N_{CPN} =< R, C_{or}, C_{SC}, G, E, M_0 > \tag{5}$$

onde:

a) R é uma rede de Petri;

b) C_{or} é um conjunto de tipos não-vazios chamados cores;

c) C_{SC}: $P \rightarrow C_{or}$ *é uma função de cor que associa a cada lugar um subconjunto de* C_{or} *(as cores possíveis para este lugar);*

d) G: $T \rightarrow exp$ *é uma função de guarda que associa a cada transição uma expressão de tipo booleano tal que* $\forall t \in T | Type(G(t)) = Boolean \wedge Type(Var(G(t))) \subseteq C_{or}$;

e) E: $Pre \cup Post \rightarrow \{expressões\}$ *é uma função de expressões de arco, que associa a cada arco uma expressão com domínio em* C_{or}. *A imagem de cada expressão de arco deve ser um multiconjunto com elementos da mesma cor associada ao lugar que está vinculado ao arco. Em outras palavras:* $\forall a \in (Pre \cup Post) | Type(E(a)) = C_{or} \wedge Type(Var(E(a))) \subseteq C_{or}$;

f) M_0 *é a marcação inicial, associando a cada lugar* $p \in P$ *uma expressão cujo resultado é um multiconjunto sobre o conjunto de cores dos lugares.*

A dinâmica de uma CPN é determinada por suas transições, que podem apresentar expressões de guarda associadas a elas. Expressões de guarda são expressões booleanas que indicam qual tipo de ficha possibilita a ativação uma transição, restringindo assim o disparo das transições. As fichas em uma CPN não se deslocam livremete, sendo as funções de guarda e as inscrições em arcos encarregadas por definir as condições para que uma ficha possa se deslocar ou não de um lugar para outro da rede.

A Figura 5 mostra um exemplo de uma CPN cuja estrutura é composta por dois lugares ($p1$ e $p2$) e uma transição ($T1$). No lugar $p1$ existe uma ficha, associada à cor (ou domínio) *TOKEN*. *TOKEN* é o produto de uma variável do tipo *Inteiro* e uma do tipo *String*. Sendo assim, o lugar inicial tem uma ficha no formato (1,"$first\_token$"), respeitando a cor associada a ela. Os arcos tem a variável x associada a eles, de forma que x é do tipo *TOKEN*.

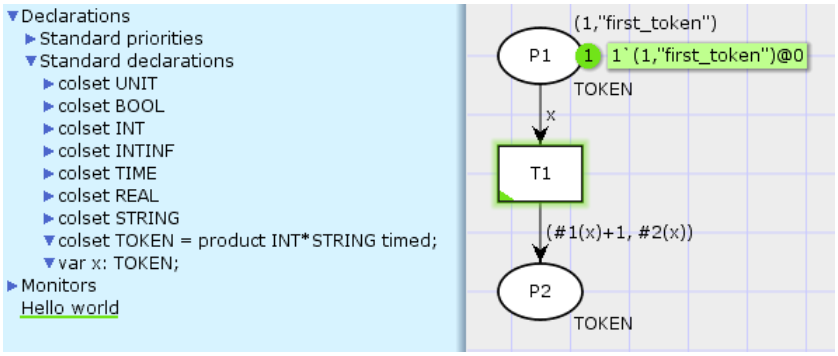
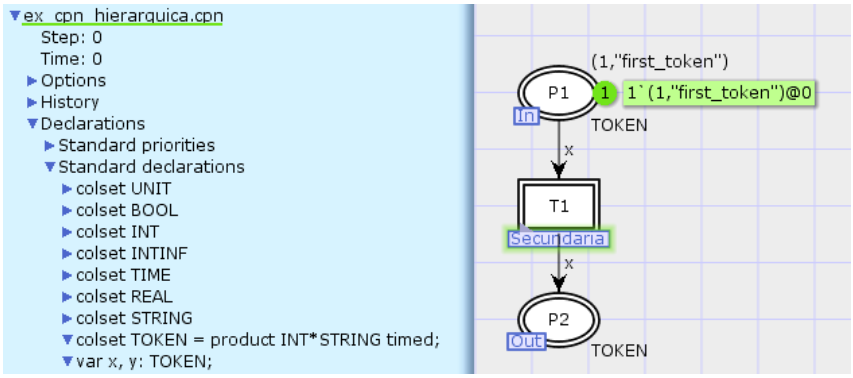


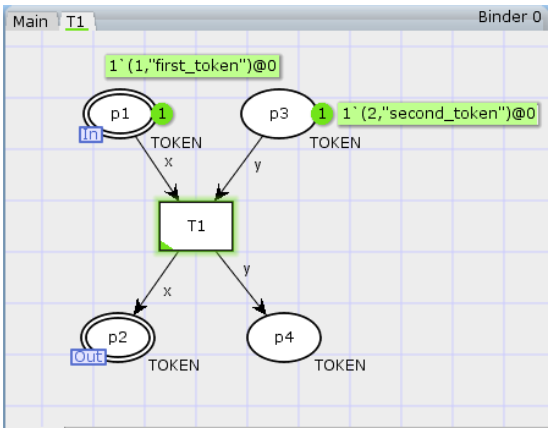
Figura 5 – Rede de Petri Colorida.

2.1.3.1 CPN Tools

A aplicação prática da modelagem e análise das CPNs depende fortemente da existência de ferramentas computacionais que ofereçam suporte à criação e manipulação de



(a) Página Principal.



(b) Página Secundária.

Figura 6 – CPN Hierárquica.

tais modelos. CPN Tools é uma ferramenta adequada para edição, simulação e análise dos modelos de redes de Petri Coloridas (JENSEN; KRISTENSEN, 2009).

Ao utilizar a ferramenta CPN Tools é possível, através da simulação do modelo do processo, investigar o comportamento e analisar o desempenho do sistema modelado. E além disso, por meio da análise do espaço de estado (*state space*), verificar suas propriedades (JENSEN; KRISTENSEN; WELLS, 2007). A ferramenta fornece condições para modelar processos distribuídos, considerando os vários tipos de processos envolvidos. Além de informações do ponto de vista qualitativo, é possível gerar simulações e investigar o comportamento de um sistema modelado do ponto de vista quantitativo.

O *software* permite também estruturar os modelos de rede de Petri em módulos. Baseado em um mecanismo de estruturação hierárquica, a ideia básica da hierarquia em redes de Petri é permitir a construção de um amplo modelo a partir da combinação de pequenos modelos (JENSEN; KRISTENSEN, 2009). A Figura 6 mostra um exemplo de uma CPN hierárquica modelada no CPN Tools. Na Figura 6a está representada a página principal, com a rede de Petri principal do modelo, e na página secundária, apresentada na Figura 6b está o modelo que representa a transição $T1$ do modelo principal.

Esta ferramenta permite também modelar redes de Petri Temporais e Temporizadas. É possível adicionar o conceito de tempo nas fichas, nas transições e/ou nos arcos, sendo

permitido especificar um tempo fixo ou um intervalo de tempo de simulação nesses componentes. Com essas características é possível realizar análises quantitativas e verificar a performance dos modelos construídos no CPN Tools.

O CPN Tools usa a linguagem CPN ML para declarações e atribuições de estruturas de dados complexas, sendo que se trata de uma linguagem baseada no padrão Standard ML (RATZER et al., 2003). O CPN ML faz com que o CPN Tools possibilite uma certa capacidade de programação, permitindo declaração de variáveis, criação de funções e estruturas de controle que poderão ser utilizadas nos lugares, arcos e transições, possibilitando a manipulação de dados abstratos complexos e a criação de condições de guardas associadas às transições do modelo.

Um dos conectores de linguagens de CPN hierárquica é o lugar de fusão (JENSEN; KRISTENSEN, 2009), que tem como objetivo permitir a especificação de um conjunto de lugares como sendo idênticos, ou seja, embora desenhados como lugares individuais, conceitualmente todos representam um único lugar. Dessa forma, quando uma ficha é adicionada/removida de um desses lugares, uma ficha idêntica é adicionada/removida de todos os outros lugares. Na ferramenta CPN Tools, ao definir um lugar de fusão, uma etiqueta é adicionada a ele, definindo o nome do conjunto de fusão ao qual ele pertence.

No CPN Tools, cada elemento da rede de Petri possui atributos descritos na linguagem CPN ML. Os atributos dos respectivos elementos de um modelo são representados a seguir.

Inscrições nos Lugares

Existem três tipos de inscrições que podem ser associadas a um lugar, de forma que somente uma delas é obrigatória e as outras duas são opcionais, como mostra a Figura 7.

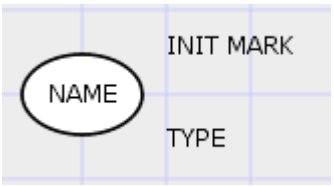


Figura 7 – Inscrições Associadas a um Lugar.

a) PLACE TYPE (obrigatória) - inscrição do conjunto de cores associado ao lugar;
b) INIT MARK (opcional) - inscrição da marcação inicial do lugar;
c) NAME (opcional) - nome do lugar, que não tem significado semântico, mas é importante para a compreensão do modelo.

Inscrições nos Arcos

Os arcos têm apenas uma única inscrição *expr*, que é uma expressão que deve coincidir com o conjunto de cores associado ao lugar ligado ao arco; caso contrário, será apresentada uma mensagem de erro próxima ao arco durante a verificação de sintáxe.

Existe uma diferença básica entre as incrições dos arcos de entrada e as dos arcos de saída de uma transição. A expressão do arco de entrada da transição é constituída pela escolha de fichas dos lugares de entrada da transição. A expressão do arco de saída da transição é um construtor para a criação de novas fichas. Esse construtor geralmente usa variáveis das inscrições dos arcos de entrada da transição, como pode ser visto na Figura 8.

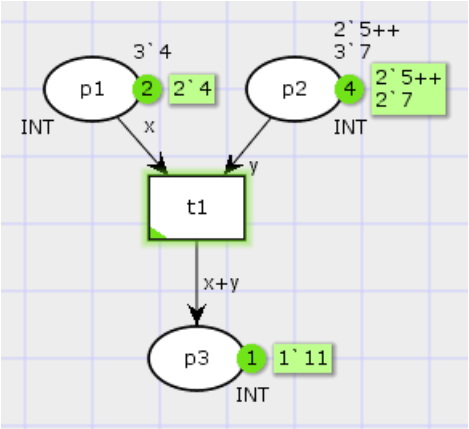


Figura 8 – Expressões nos Arcos de Entrada e de Saída de uma Transição.

Na rede apresentada na Figura 8, a marcação inicial do lugar $p1$ consiste em três fichas de valor 4 ($3'4$). A marcação inicial em $p2$ é composta por duas fichas de valor 5 ($2'5$) e três fichas de valor 7 ($3'7$). Então, $p2$, inicialmente, possui um total de cinco fichas. As variáveis x e y escolhem aleatoriamente uma ficha de cada um de seus lugares de entrada, retirando uma ficha de cada lugar $p1$ e $p2$. A expressão do arco de saída da transição é o construtor de uma nova ficha criada a partir da soma dos valores que as variáveis x e y assumem no momento do disparo da transição. No exemplo, $x = 4$ e $y = 7$.

As expressões dos arcos saída das transições podem ser temporizadas. O atraso de tempo, indicado pela expressão @+ $< TEMPO >$ associado ao arco de saída, garante que a nova ficha produzida seja utilizada apenas quando o tempo corrente alcançar o tempo associado à ficha.

Na Figura 9, a inscrição do arco ($t1$,$p2$) contém um atraso de tempo no valor de 5 unidades. Dessa forma, a cada 5 unidades de tempo, uma ficha é produzida e adicionada ao lugar $p2$.

Inscrições nas Transições

Existem cinco tipos de inscrições que podem ser associadas a uma transição, sendo todas elas opcionais, como mostra a Figura 10.

a) NAME - nome;

b) P_NORMAL - prioridade de disparo;

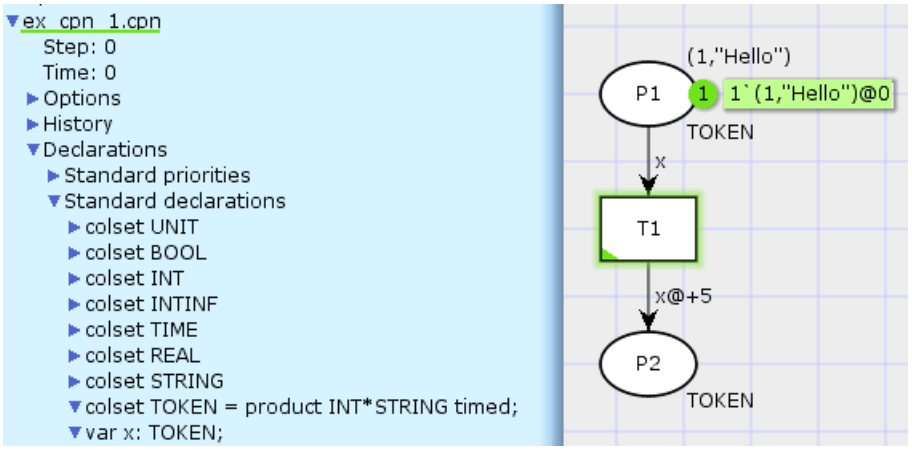


Figura 9 – Expressão Temporizada em um Arco de Saída de uma Transição.

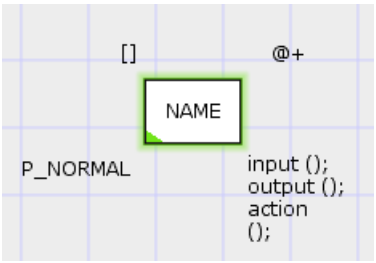


Figura 10 – Inscrições Associadas a uma Transição.

c) [ ] - guarda;

d) @+ - atraso de tempo;

e) $input(); output(); action();$ - segmento de código.

Cada transição tem uma prioridade de disparo associada a ela. Três prioridades com graus diferentes são definidas no CPN Tools, são elas: P_HIGH, P_NORMAL e P_LOW. Caso nenhuma prioridade seja definida, a padrão para o CPN Tools é P_NORMAL.

A inscrição referente ao atraso de tempo deve ser uma expressão positiva. Além disso, o atraso é sempre relativo ao tempo atual, ou seja, se, por exemplo, o tempo atual é 20 e o atraso associado à transição é igual a $@+15$, então o tempo associado às fichas colocadas nos lugares de saída da transição, após seu disparo, será igual a $20 + 15$. Quando uma transição não possui um tempo associado, o atraso relacionado ao disparo é igual a zero.

A inscrição de guarda é uma expressão booleana da linguagem CPN ML avaliada como verdadeira ou falsa, podendo ser composta por zero, uma ou várias expressões booleanas. Caso nenhuma expressão de guarda seja associada a uma dada transição, o valor padrão para essa inscrição será sempre verdadeiro. Por outro lado, caso uma expressão de guarda seja associada a determinada transição, o disparo da mesma só poderá ocorrer caso a expressão seja avaliada como verdadeira.

Com relação ao segmento de código, caso seja definido para uma determinada transição, é executado assim que a mesma é disparada. Cada segmento de código pode utilizar

variáveis CPN e associá-las aos arcos de saída de uma transição, mesmo que elas não tenham sido associadas a nenhuma ficha dos lugares de entrada da transição. Um segmento de código pode conter:

a) padrão de entrada (*input pattern*) - opcional: conjunto de variáveis CPN usadas no *code action*, que não podem ter seus valores modificados. Caso não seja declarado, nenhuma variável CPN pode ser usada no *code action*;

b) padrão de saída (*output pattern*) - opcional: conjunto de variáveis CPN modificadas de acordo com o resultado da execução do *code action*. Caso não seja declarado, nenhuma variável CPN pode ser calculada;

c) código (*code action*) - obrigatório: expressão ML executada como uma declaração local em um ambiente contendo variáveis CPN especificadas no padrão de entrada. Nessa expressão, nenhuma declaração de conjunto de cores, variáveis CPN ou de referência pode ser utilizada. No entanto, ela pode utilizar constantes pré-declaradas ou declaradas pelo usuário, operações e funções. Além disso, novas funções e constantes podem ser definidas localmente por meio da cláusula $let - in - end$;

2.1.4 *Workflow Nets*

Workflow é um termo que em português significa fluxo de trabalho. Contextualizando para a área tecnologica, *workflow* consiste na utilização de ferramentas computacionais para auxiliar nos passos necessários para a execução de um processo de negócio (REZENDE et al., 2013).

Um processo de *workflow* define quais tarefas de um processo de negócio devem ser executadas e em qual ordem essa execução deve ocorrer. Além disso, de acordo com (AALST, 1998), processos de *workflow* são baseados em casos , isto é, cada parte do trabalho é executada para um caso específico. Modelar um proceso de *workflow* em termos de uma rede de Petri é bem direto: transições são componentes ativos e modelam as tarefas, lugares são componentes passivos e modelam as condições (pré e pós), e as fichas modelam os casos (AALST, 1998; AALST; HEE; HEE, 2004).

Assim, uma Workflow net (WF-net) é uma extensão de uma rede de Petri que modela um processo de *workflow* (AALST, 1998) e satisfaz os seguintes requisitos:

a) tem apenas um lugar de início i (denominado *Start*) e apenas um lugar de saída o (denominado *End*), de forma que o lugar i possui somente arcos de saída e o lugar o possui somente arcos de entrada;

b) a rede de Petri é fortemente conexa, o que significa que todo nó da rede, lugar e transição, deve estar no caminho entre o lugar de início i e o lugar de fim o.

Definição 6 (Workflow net)**.** *Uma rede de Petri*

$$WF = (P, T, Pre, Post) \tag{6}$$

é uma WF-net se, e somente se:

1. *existe um único lugar de início* $i \in P$ *tal que* $\bullet i = \phi$*;*

2. *existe um único lugar de saída* $o \in P$ *tal que* $o\bullet = \phi$*;*

3. *todo nó* $x \in P \cup T$ *está em um caminho entre os lugares* i *e* o*;*

Para ilustrar o mapeamento de um processo por uma *Workflow net*, considera-se o processo de tratamento de reclamações ("Handle Complaint Process") apresentado em (AALST; HEE; HEE, 2004):

> An incoming complaint first is recorded. Then the client who has complained and the department affected by the complaint are contacted. The client is approached for more information. The department is informed of the complaint and may be asked for its initial reaction. These two tasks may be performed in parallel – that is, simultaneously or in any order. After this, the data are gathered and a decision is taken. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed (AALST; HEE; HEE, 2004).

A Figura 11 representa o processo de negócio descrito. De acordo com (AALST; HEE; HEE, 2004), as seguintes construções básicas para o roteamento de tarefas devem ser consideradas:

a) sequencial: a forma mais simples de execução de tarefas, na qual uma tarefa é executada após a outra, havendo claramente, dependência entre elas;

b) paralela: mais de uma tarefa pode ser executada simultaneamente, ou em qualquer ordem. Neste caso, as tarefas podem ser executadas sem que o resultado de uma interfira no resultado das outras;

c) condicional (ou rota seletiva): quando há uma escolha entre duas ou mais tarefas;

d) iterativa: quando é necessário executar uma mesma tarefa múltiplas vezes.

Considerando o processo de tratamento de reclamações mostrado na Figura 11, as tarefas *Contact_Client* e *Contact_Department* são exemplos de roteamento paralelo. As tarefas *Collect* e *Assess* são um exemplo de roteamento sequencial. Já as tarefas *Pay* e *Send_Letter* são um exemplo de roteamento condicional. Conforme apresentado por (PASSOS; JULIA, 2009) um roteamento iterativo pode ser substituido por uma tarefa global, que considera a definição dos blocos bem formados, apresentada por (VALETTE, 1979). Assim, uma rota iterativa será substituída por uma tarefa global, como mostra a Figura 12.

Além dos roteamentos, (AALST; HEE; HEE, 2004) também define a noção de acionamento, sendo o acionamento uma condição externa que guia a execução de uma tarefa habilitada. De acordo com Aalst e Hee (2004), existem quatro tipos distintos de acionamento:

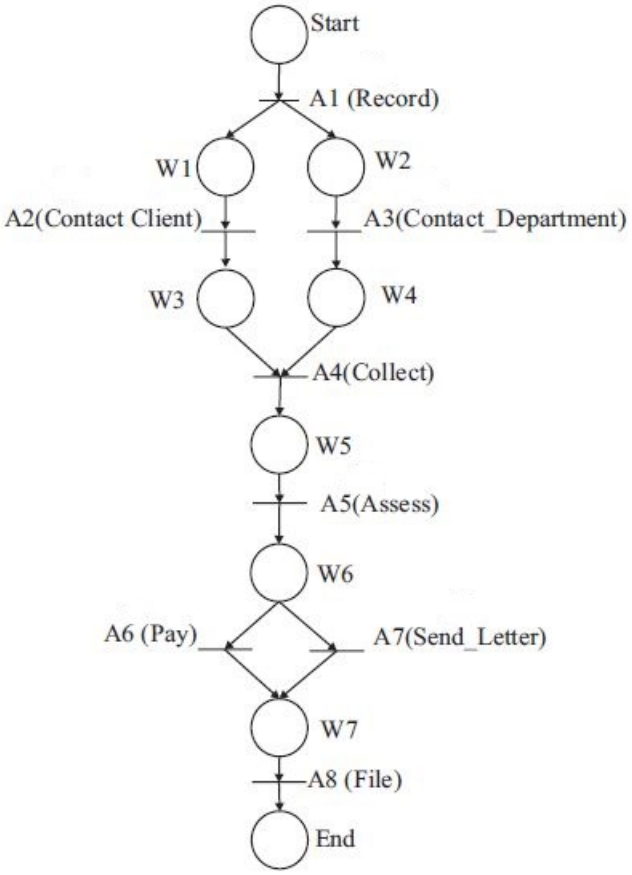


Figura 11 – *Workflow net* para o "Processo de Tratamento de Reclamações" (AALST; HEE; HEE, 2004)

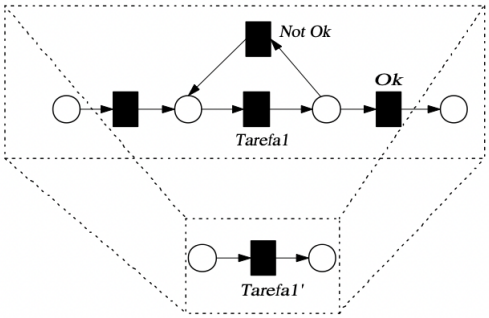


Figura 12 – Substituição de um roteiro iterativo por uma tarefa global (PASSOS; JULIA, 2009)

a) usuário: uma tarefa é acionada por um recurso humano;

b) mensagem: um evento externo aciona uma tarefa habilitada;

c) tempo: uma tarefa habilitada é acionada por um relógio, isto é, uma tarefa é executada em um tempo pré-definido;

d) automático: uma tarefa é acionada no momento em que é habilitada e não requer interação humana.

Somente nos acionamentos do tipo usuário, onde uma tarefa é acionada por um recurso

(humano, máquina, etc.) existe a necessidade de alocação de recurso para o tratamento da tarefa. Nos demais tipos de acionamento não é obrigatória a alocação de recursos associada. Na Figura 11 as tarefas *Record*, *Contact_Client*, *Contact_Department*, *Assess*, *Pay* e *Send_Letter* são tarefas do tipo usuário e as demais tarefas (*Collect* e *File*) são tratadas automaticamente.

2.1.4.1 *Soundness*

A propriedade *Soundness* é o principal critério de correção definido para as WF-nets. Uma WF-net é *Sound* se, e somente se, os três requisitos a seguir são satisfeitos (AALST; HEE; HEE, 2004):

a) para cada ficha colocada no lugar de início i, uma (e apenas uma) ficha aparecerá no lugar de término o;

b) quando uma ficha aparecer no lugar o, todos os outros lugares estarão vazios, considerando o caso em questão;

c) considerando uma tarefa associada a uma transição t, é possível evoluir a marcação inicial M_0 até uma marcação M' que sensibiliza tal transição, ou seja, não deve haver nenhuma transição morta na WF-net.

Em outras palavras, um modelo de processo é considerado correto (*Sound*) se ele não contém nenhuma transição morta, ou seja, uma transição que nunca pode ser alcançada, independente do estado da rede, e se todo caso iniciado é totalmente concluído em algum momento, não restando nenhuma referência a ele (nenhuma ficha remanescente) no sistema. A definição formal do critério de correção *Soundness* no contexto da WF-net, proposto por Aalst em (AALST, 1998), é apresentada na sequência:

Definição 7 (Soundness)**.** *Um processo de workflow modelado por uma WF-net*

$$WF = (P, T, Pre, Post) \tag{7}$$

é sound se, e somente se:

a) para cada marcação M alcançável a partir da marcação inicial M_0, existe uma sequência de disparo que leva da marcação M até a marcação final M_f. Formalmente:

$$\forall_M (M_0 \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} M_f); \tag{8}$$

b) a marcação final M_f é a única alcançável a partir da marcação inicial M_0 com pelo menos uma ficha no lugar de término o e somente neste lugar. Formalmente:

$$\forall_M ((M_0 \xrightarrow{*} M) \wedge (M \geq M_f)) \Rightarrow (M = M_f); \tag{9}$$

c) não existe nenhuma transição morta em (R, M_0). Formalmente:

$$\forall_{t \in T} \exists_{M,M'} (M_0 \xrightarrow{*} M \xrightarrow{t} M'). \tag{10}$$

A *Workflow net* mostrada na Figura 11 é *Sound*, pois quando uma ficha atinge o lugar de término, não existe nenhuma ficha remanescente, considerando qualquer sequência de disparo, para cada ficha colocada no lugar de início, apenas uma ficha aparecerá no lugar de término e, não existe nenhuma transição morta na rede.

2.1.5 Lógica Linear

A Lógica Linear foi proposta por Jean-Yves Girard em 1987 (GIRARD, 1987). Na Lógica Linear, as proposições são consideradas como recursos que são consumidos e produzidos em cada mudança de estado, o que diferencia da Lógica Clássica na qual as proposições assumem um valor *booleano* (*verdadeiro* ou $falso$) (PRADIN-CHÉZALVIEL; VALETTE; KUNZLE, 1999).

A Lógica Linear introduz sete novos conectivos divididos em três grupos:

a) conectivos multiplicativos: "times"($\otimes$), "par"($\parr$) e "implicação linear"($\multimap$), que são versões bilineares do "e", "ou" e "implica" respectivamente;

b) conectivos aditivos: "plus"($\oplus$) e "with"(&), que são versõs lineares do "ou" e "e" respectivamente;

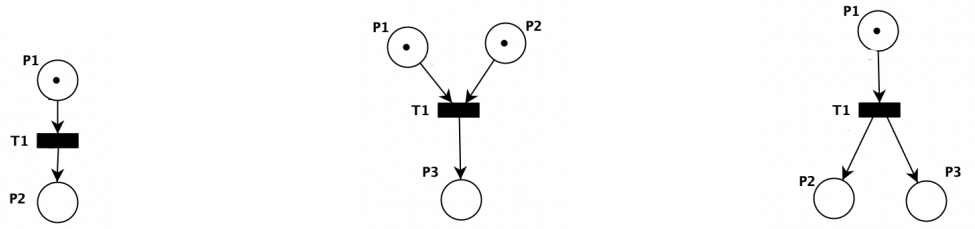
c) conectivos exponenciais: "of course"(!) and "why not"(?), que possuem alguma similaridade com os modais $\Box$ (necessariamente) e $\Diamond$ (possivelmente) e são essenciais para preservar a força da lógica.

A tradução de uma rede de Petri em fórmulas da Lógica Linear é apresentada em (PRADIN-CHÉZALVIEL; VALETTE; KUNZLE, 1999). Para representar uma rede de Petri através da Lógica Linear, apenas dois conectivos da Lógica Linear são necessários e serão considerados neste trabalho, e são eles:

a) conectivo *times*, denotado pelo símbolo $\otimes$, que representa a disponibilidade simultânea de recursos. Por exemplo, $A \otimes B$ representa a disponibilidade simultânea dos recursos A e B;

b) conectivo *implicação linear*, denotado pelo símbolo $\multimap$, que representa uma mudança de estado. Por exemplo, $A \multimap B$ significa que o recurso B é produzido quando o recurso A é consumido.

A Lógica Linear possui características como a noção de mudança de estado (PRADIN-CHÉZALVIEL; VALETTE; KUNZLE, 1999) e a manipulação de recursos (PRADIN-CHÉZALVIEL et al., 1999) que possibilitam uma aproximação com a teoria das rede de Petri. Apesar dos modelos em redes de Petri poderem ser reescritos utilizando a Lógica Linear, as duas teorias não são equivalentes e cada uma delas possuem suas vantagens específicas (CHAMPAGNAT; PRADIN-CHÉZALVIEL; VALETTE, 2000), como o fato das redes de Petri permitirem o cálculo de invariantes de lugar e de transição, enquanto a Lógica Linear trata claramente cenários que envolvem a produção e consumo de recursos (SOARES, 2004).



(a) (b) (c)

Figura 13 – Exemplo da Tradução de Redes de Petri em Fórmulas da Lógica Linear.

A transformação de uma rede de Petri em fórmulas da Lógica Linear é realizada conforme apresentado em (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001). Uma marcação M é um monômio em $\otimes$ que é representado por $M = A1 \otimes A2 \otimes ... \otimes Ak$, onde A_i são nomes de lugares. Por exemplo, considerando as redes das Figuras 13a e 13c, tem-se a marcação incial $M = p1$. Já a marcação inicial da rede de Petri da Figura 13b é dada por $M = p1 \otimes p2$.

Uma transição é uma fórmula $M_1 \multimap M_2$, onde M_1 e M_2 são marcações. Esta expressão representa o disparo da transição: ela aparecerá em um sequente o número de vezes em que esta transição for disparada (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001). Se considerarmos como exemplo a transição $t1$ da rede de Petri apresentada na Figura 13a, tem-se que $t1 = p1 \multimap p2$. Para a rede da Figura 13b, tem-se que $t1 = p1 \otimes p2 \multimap p3$. Já para a rede da Figura 13c, tem-se $t1 = p1 \multimap p2 \otimes p3$.

Um sequente representa o disparo de uma transição (ou uma sequência de possíveis transições). O sequente $M, t_i \vdash M'$ representa um cenário onde M é a marcação inical, M' a marcação final e t_i é uma lista de transições não ordenadas (JULIA; SOARES, 2003). Pegando como exemplo a Figura 13a e considerando sua marcação final $p2$, tem-se o seguinte sequente linear $p1, p1 \multimap p2 \vdash p2$.

Para provar a corretude de um sequente, deve ser construida uma árvore de prova a partir da aplicação de regras do cálculo de sequente. De acordo com (GIRAULT; PRADIER-CHEZALVIEL; VALETTE, 1997) a prova de um sequente da Lógica Linear é equivalente ao problema de alcançabilidade correspondente na teoria das redes de Petri.

O presente trabalho considera apenas algumas das regras da Lógica Linear. As regras serão utilizadas para representar uma rede de Petri através da Lógica Linear. Para este fim, três regras serão explicadas e terão suas aplicações exemplificadas, as demais regras do cálculo de sequente da Lógica Linear podem ser encontradas em (GIRARD, 1987) e em (GIRARD, 1995). F, G e H são consideradas fórmulas e, Γ e Δ são blocos de fórmulas da Lógica Linear. As regras a seguir, apresentadas em (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001) são as regras utilizadas nos métodos propostos nesta pesquisa:

a) a regra $\multimap_L$ expressa o disparo de uma transição e gera dois sequentes. O sequente da esquerda representa as fichas consumidas pelo disparo da transição enquanto

que o sequente da direita representa o subsequente que ainda precisa ser provado. A regra $\multimap_L$ é dada por:

$$\frac{\Gamma \vdash F \;\; \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H} \multimap_L$$

b) a regra $\otimes_L$ é usada para transformar uma marcação em uma lista de átomos e é dada por:

$$\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L$$

c) a regra $\otimes R$ transforma um sequente na forma $A, B \vdash A \otimes B$ em dois sequentes identidade $A \vdash A$ e $B \vdash B$. A regra $\otimes R$ é dada por:

$$\frac{\Gamma \vdash F \;\; \Delta \vdash G}{\Delta, \Gamma \vdash F \otimes G} \otimes_R$$

Para exemplificar as regras apresentadas anteriormente, considerando a transição $A1(Record) = Start \multimap W1 \otimes W2$ da WF-net apresentada na Figura 11, pela aplicação da regra $\multimap_L$, dois sequentes são gerados: $Start \vdash Start$, representando a ficha consumida pelo disparo da transição, e $W1 \otimes W2$, o subsequente remanescente, que ainda precisa ser provado:

$$\frac{Start \vdash Start \;\; W1 \otimes W2, ... \vdash End}{Start, Start \multimap W1 \otimes W2, ... \vdash End} \multimap L$$

Utilizando a marcação $W1 \otimes W2$ produzida a partir da aplicação da regra $\multimap L$ na transição $A1(Record)$, é possível aplicar a regra $\otimes L$ para transformá-la em uma lista de átomos $W1$, $W2$, temos:

$$\frac{W1, W2, ... \vdash End}{W1 \otimes W2, ... \vdash End} \otimes L$$

Um exemplo da aplicação da regra $\otimes R$ seria, considerando a transição $A4(Collect) = W3 \otimes W4 \multimap W5$ da *WF-net* apresentada na Figura 11, o sequente que representa a ficha consumida pelo disparo da transição, $W3, W4 \vdash W3 \otimes W4$ também precisa ser provado. Utilizando a regra $\otimes R$, temos:

$$\frac{W3 \vdash W3 \;\; W4 \vdash W4}{W3, W4 \vdash W3 \otimes W4} \otimes R$$

Para exemplificar a construção de uma árvore de prova canônica da Lógica Linear, considere a rede de Petri da Figura 14. Transformando as transições da rede de Petri em fórmulas da Lógica Linear, temos:

$t_1 = P1 \multimap P2 \otimes P3$,
$t_2 = P2 \multimap P4$,
$t_3 = P3 \multimap P5$,

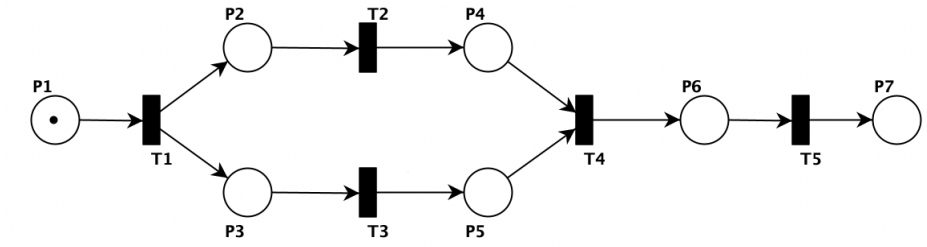


Figura 14 – Exemplo de rede de Petri para construção de uma árvore de prova canônica da Lógica Linear.

$t_4 = P4 \otimes P5 \multimap P6$,
$t_5 = P6 \multimap P7$.

A marcação inicial desta rede de Petri é $M = P1$. Assim, o sequente linear a ser provado é dado por $P1, t_1, t_2, t_3, t_4, t_5 \vdash P7$ onde $P1$ e $P7$ são, respectivamente, a marcação inicial e final da rede de Petri da Figura 14. Aplicando as regras do cálculo de sequente a este sequente linear, é possível provar se o mesmo é ou não um sequente sistematicamente válido. A árvore de prova é apresentada na sequencia.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P6 \vdash P6} \quad \dfrac{}{P7 \vdash P7}\,\multimap_L}{P4 \vdash P4 \quad P5 \vdash P5}\,\otimes_R \quad P6, P6 \multimap P7 \vdash P7}\,\multimap_L}{P3 \vdash P3 \quad P4, P5, P4 \otimes P5 \multimap P6, t_5 \vdash P7}\,\multimap_L}{P2 \vdash P2 \quad P3, P4, P3 \multimap P5, t_4, t_5 \vdash P7}\,\multimap_L}{P2, P3, P2 \multimap P4, t_3, t_4, t_5 \vdash P7}\,\otimes_L}{P1 \vdash P1 \quad P2 \otimes P3, P2 \multimap P4, t_3, t_4, t_5 \vdash P7}\,\multimap_L}{P1, P1 \multimap P2 \otimes P3, t_2, t_3, t_4, t_5 \vdash P7} \tag{11}$$

Uma vez que a árvore de prova da Lógica Linear é construída de forma *bottom-up*, a raíz da árvore de prova corresponde à primeira linha da árvore, neste caso $P1 \vdash P1 \quad P2 \otimes P3, P2 \multimap P4, t_3, t_4, t_5 \vdash P7$, que é o sequente linear a ser provado. A primeira regra a ser aplicada é a regra $\multimap_L$. Como mencionado anteriormente, a regra $\multimap_L$ expressa o disparo de uma transição, neste caso, a transição t_1, e gera dois sequentes, tais que o sequente à esquerda, i.e. $P1 \vdash P1$, representa as fichas consumidas pelo disparo da transição t_1 e o sequente à direita, i.e. $P2 \otimes P3, P2 \multimap P4, t_3, t_4, t_5 \vdash P7$ representa o subsequente restante a ser provado. O sequente identidade $P1 \vdash P1$, à esquerda é um nó folha da árvore de prova. Deve-se notar que os átomos $P2 \otimes P3$ no sequente à direita representam os átomos produzidos pelo disparo da transição t_1. A regra $\otimes_L$ deve então ser aplicada para transformar tal marcação em uma lista de átomos. Assim, após a aplicação da regra $\otimes_L$, tem-se o sequente $P2, P3, P2 \multimap P4, t_3, t_4, t_5 \vdash P7$. Para provar tal sequente, mais uma vez a regra $\multimap_L$ deve ser aplicada. A aplicação desta regra

representa o disparo da transição t_2. Novamente são gerados dois sequentes, tais que o sequente à esquerda é o nó folha $P2 \vdash P2$, que representa o átomo consumido pelo disparo de t_2 e o sequente à direita é dado por $P3, P4, P3 \multimap P5, t_4, t_5 \vdash P7$, onde: $P3$ é o átomo disponível no sequente anterior e não utilizado no disparo t_2; $P4$ é o átomo produzido pelo disparo de t_2; t_3, t_4 e t_5 representam as fórmulas para as referidas transições e, finalmente, $P7$ representa a marcação a ser atingida. Para a prova de $P3, P4, P3 \multimap P5, t_4, t_5 \vdash P7$, mais uma vez a regra $\multimap_L$ deve ser aplicada. A aplicação desta regra, desta vez, representa o disparo da transição t_3. Novamente, dois sequentes são gerados: $P3 \vdash P3$ e $P4, P5, P4 \otimes P5 \multimap P6, t_5 \vdash P7$. O primeiro sequente corresponde ao átomo consumido pelo disparo da transição t_3 e o segundo corresponde ao subsequente restante a ser provado, onde $P5$ representa o átomo produzido pelo disparo de t_3. Mais uma vez a regra $\multimap_L$ deve ser aplicada. Desta vez a aplicação desta regra representa o disparo de t_4. Note que a fórmula para a transição t_4 é $P4 \otimes P5 \multimap P6$, ou seja, deve-se consumir os átomos $P4$ e $P5$ para produzir o átomo $P6$. Assim, após a aplicação da regra $\multimap_L$, tem-se mais dois sequentes a serem provados: $P4, P5 \vdash P4 \otimes P5$ e $P6, P6 \multimap P7 \vdash P7$. O sequente $P4, P5 \vdash P4 \otimes P5$ representa os átomos consumidos no disparo de t_4 e para provar tal sequente, deve-se utilizar a regra $\otimes_R$. A regra $\otimes_R$ transforma o sequente $P4, P5 \vdash P4 \otimes P5$ em dois sequentes identidade e, consequentemente, nós folha da árvore de prova: $P4 \vdash P4$ e $P5 \vdash P5$. Deve-se então provar o sequente $P6, P6 \multimap P7 \vdash P7$. Para provar tal sequente, deve-se aplicar a regra $\multimap_L$ que, neste caso, representa o disparo da transição t_5. A aplicação da regra $\multimap_L$ gera dois sequentes identidade e nós folha da árvore de prova: $P6 \vdash P6$ e $P7 \vdash P7$. como todos os nós folha da árvore de prova apresentada e detalhada são sequentes identidade, tem-se que o sequente linear $P1, t_1, t_2, t_3, t_4, t_5 \vdash P7$ é sintaticamente correto.

No contexto das redes de Petri t-temporais, em uma árvore de prova da Lógica Linear, cada disparo de transição pode gerar uma data simbólica associada a cada átomo (ficha), como apresentado em (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001).

Neste trabalho, D_i denota uma data e d_i a duração da sensibilização associada a um disparo de transição (t_i). Um par (D_p, D_c) será associado a cada átomo da árvore de prova, de forma que D_p representa a data de produção e D_c, a data de consumo de um átomo. O cálculo das datas em árvores de prova canônica é dado pelos seguintes passos, mostrados em (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001):

a) determinar a data de produção D_i para todas as fichas da marcação inicial;

b) para cada aplicação da regra $\multimap L$, calcule a data de disparo da transição correspondente, isto é igual ao valor máximo da data de produção entre os átomos consumidos por esta transição, somado à duração d_i de sensibilização da transição considerada;

c) atualizar as datas de todas os átomos produzidos e consumidos.

A Figura 15 mostra uma rede de Petri semelhante à apresentada na Figura 14, porém com representação de tempo associada às suas transições (rede de Petri t-temporal). Considerando a Figura 15, as fórmulas da Lógica Linear para cada uma das transições definidas anteriormente, sua marcação inicial $P1$ e seja $Seq = D_1 + d_1 + max(d_2, d_3) + d_4$, temos a seguinte prova com cálculo de datas:

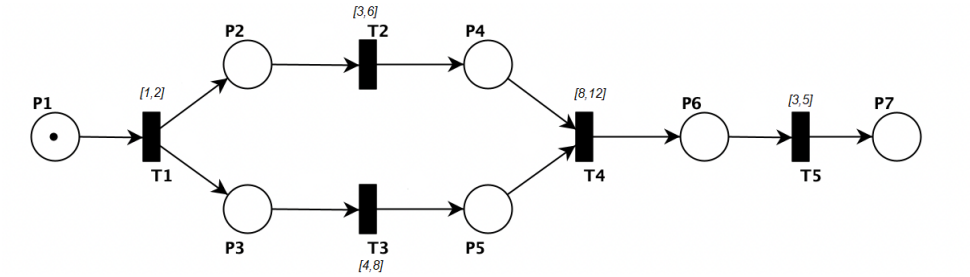


Figura 15 – Exemplo de rede de Petri t-temporal.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P6(Seq,Seq+d_5)\vdash P6} \quad \dfrac{}{P7(Seq+d_5,.)\vdash P7}\,{\multimap_L}}{P6(Seq,.),P6\multimap P7\vdash P7}\,{\multimap_L}}{P3(D_1+d_1,D_1+d_1+d_3)\vdash P3} \quad \dfrac{\dfrac{}{P4(D_1+d_1+d_2,Seq)\vdash P4} \quad \dfrac{}{P5(D_1+d_1+d_3,Seq)\vdash P5}\,{\otimes_R}}{P4(D_1+d_1+d_2,Seq),P5(D_1+d_1+d_3,Seq)\vdash P4\otimes P5}}{P4(D_1+d_1+d_2,.),P5(D_1+d_1+d_3,.),P4\otimes P5\multimap P6,t_5\vdash P7}\,{\multimap_L}}{P2(D_1+d_1,D_1+d_1+d_2)\vdash P2} \quad P3(D_1+d_1,.),P4(D_1+d_1+d_2,.),P3\multimap P5,t_4,t_5\vdash P7}\,{\multimap_L}}{P2(D_1+d_1,.),P3(D_1+d_1,.),P2\multimap P4,t_3,t_4,t_5\vdash P7}\,{\otimes_L}}{P1(D_1,D_1+d_1)\vdash P1} \quad P2(D_1+d_1,.)\otimes P3(D_1+d_1,.),P2\multimap P4,t_3,t_4,t_5\vdash P7}\,{\multimap_L}}{P1(D_1,.),P1\multimap P2\otimes P3,t_2,t_3,t_4,t_5\vdash P7}\,{\multimap_L} \tag{12}
$$

A Tabela 1 apresenta as datas simbólicas de produção e de consumo de cada átomo da rede de Pètri da Figura 15.

Tabela 1 – Datas simbólicas de produção e consumo dos átomos.

Átomo	**Data de Produção**	**Data de Consumo**
$P1$	D_1	$D_1 + d_1$
$P2$	$D_1 + d_1$	$D_1 + d_1 + d_2$
$P3$	$D_1 + d_1$	$D_1 + d_1 + d_3$
$P4$	$D_1 + d_1 + d_2$	$D_1 + d_1 + max(d_2, d_3) + d_4$
$P5$	$D_1 + d_1 + d_3$	$D_1 + d_1 + max(d_2, d_3) + d_4$
$P6$	$D_1 + d_1 + max(d_2, d_3) + d_4$	$D_1 + d_1 + max(d_2, d_3) + d_4 + d_5$
$P7$	$D_1 + d_1 + max(d_2, d_3) + d_4 + d_5$	-

Em um modelo de rede de Petri t-temporal, toda duração de sensibilização d_i de uma transição t_i tem um valor que pertence a um intervalo de tempo $\Delta_i = [\delta_{imin}, \delta_{imax}]$. Dessa forma, como todas as datas simbólicas calculadas dependem de d_i, seus domínios também serão em função de intervalos de tempo. A Tabela 2 mostra os intervalos de visibilidade de datas simbólicas de produção e consumo de cada átomo da rede de Petri t-temporal da Figura 15.

Tabela 2 – Intervalo simbólico de datas de produção e consumo dos átomos.

Átomo	**Data de Produção**	**Data de Consumo**
$P1$	D_1	$[D_1 + d_{1min}, D_1 + d_{1max}]$
$P2$	$[D_1 + d_{1min}, D_1 + d_{1max}]$	$[D_1 + d_{1min} + d_{2min}, D_1 + d_{1max} + d_{2max}]$
$P3$	$[D_1 + d_{1min}, D_1 + d_{1max}]$	$[D_1 + d_{1min} + d_{3min}, D_1 + d_{1max} + d_{3max}]$
$P4$	$[D_1 + d_{1min} + d_{2min}, D_1 + d_{1max} + d_{2max}]$	$[D_1 + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min}, D_1 + d_{1max} + max(d_{2max}, d_{3max}) + d_{4max}]$
$P5$	$[D_1 + d_{1min} + d_{3min}, D_1 + d_{1max} + d_{3max}]$	$[D_1 + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min}, D_1 + d_{1max} + max(d_{2max}, d_{3max}) + d_{4max}]$
$P6$	$[D_1 + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min}, D_1 + d_{1max} + max(d_{2max}, d_{3max}) + d_{4max}]$	$[D_1 + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min} + d_{5min}, D_1 + d_{1max} + max(d_{2max}, d_{3max}) + d_{4max} + d_{5max}]$
$P7$	$[D_1 + d_{1min} + max(d_{2min}, d_{3min}) + d_{4min} + d_{5min}, D_1 + d_{1max} + max(d_{2max}, d_{3max}) + d_{4max} + d_{5max}]$	-

Considerando os intervalos de datas numéricas apresentadas na Figura 15 e os intervalos de datas simbólicas apresentados na Tabela 2 é possível encontrar os intervalos de datas numéricas para produção e consumo de cada átomo da rede de Petri da Figura 15. Considerando $D_1 = 0$, o resultado dos intervalos numéricos é apresentado na Tabela

Tabela 3 – Intervalo numérido de datas de produção e consumo dos átomos.

Átomo	**Data de Produção**	**Data de Consumo**
$P1$	0	$[1, 2]$
$P2$	$[1, 2]$	$[4, 8]$
$P3$	$[1, 2]$	$[5, 10]$
$P4$	$[4, 8]$	$[13, 22]$
$P5$	$[5, 10]$	$[13, 22]$
$P6$	$[13, 22]$	$[16, 27]$
$P7$	$[16, 27]$	-

2.2 Trabalhos Relacionados

As *Workflow nets* tem sido muito utilizadas como um modelo para processos de *workflow*, por exemplo em (WANG; LI, 2013; MARTOS-SALGADO; ROSA-VELARDO, 2011; HEE; SIDOROVA; VOORHOEVE, 2006; KOTB; BADREDDIN, 2005; AALST et al., 1997) . A maioria dos modelos existentes não contempla alguns aspectos importantes do Sistema de Gerenciamento de *Workflow*, com por exemplo a alocação de recursos e a gestão de tempo.

Em (HEE; SIDOROVA; VOORHOEVE, 2006; AALST et al., 1997) o mecanismo de alocação de recursos é apresentado através de uma representação informal. Em (WANG; LI, 2013; KOTB; BADREDDIN, 2005) o mecanimso de alocação de recursos é representado por fichas simples como normalmente acontece em sistemas de produção. Porém, uma ficha simples não é capaz de representar de forma realística funcionários que podem

tratar diversos casos de forma simultânea, que é o que acontece com muita frequência hoje no setor administratio das empresas.

O tempo em processos de negócio é um ponto de grande importância, já que no caso de prazos não cumpridos, as empresas podem sofrer graves prejuizos. Um estudo sobre a gestão do tempo em processos de *workflow* concentram-se principalmente no planejamento de tempo de execução do processo como um todo, em estimativa de duração das atividades, evitando que os prazos estabelecidos sejam desrespeitados, seja no âmbito geral (do processo como um todo) ou no âmbito das atividades que o compõe (LIU, 2016; JESKE; JULIA; VALETTE, 2009; JULIA; OLIVEIRA; VALETTE, 2008) . Além disso, vários trabalhos já trataram de mecanismos de propagação com restrição de tempo em *Workflow nets* para planejamento/escalonamento de recursos compartilhados em SGW.

Em (JULIA; OLIVEIRA; VALETTE, 2008), são apresentados o raciocínio de propagação para frente e para trás para produzir intervalos de visibilidade de casos em um processo de *workflow* modelado por uma rede de Petri, porém tal abordagem considera apenas informações numéricas. Consequentemente, o mecanismo de propagação para frente e para trás devem ser aplicados toda vez que um novo caso aparece no ponto inicial de uma *Workflow net*.

Em (MEDEIROS; JULIA, 2017), um raciocínio energético é utilizado para aplicar uma espécie de filtro nas restrições de tempo de um processo de *workflow*, dado pelos intervalos de visibilidade. Os mecanismos para frente e para trás são implementados em uma linguagem de programação de restrições; estes, no entanto, precisa ser aplicados a cada novo caso que chega no processo devido aos valores numéricos dos intervalos associados às durações das atividades.

Em (FREITAS; JULIA, 2015), técnicas de propagação *fuzzy* para frente e para trás são apresentadas. A vantagem é a possibilidade de considerar restrições de tempo mais flexiveis dadas por intervalos *fuzzy*, porém, a duração das atividades ainda é dada por valores numéricos fuzzy e a técnica de propagação deve ser aplicada toda vez que um novo caso inicia o processo de *workflow*.

Em (PASSOS; JULIA, 2016), o cálculo do sequente da Lógica Linear foi utilizado em uma *Workflow net* temporal que produziu restrições de tempo sombólicas, de uma forma muito parecida com a apresentada neste trabalho. Porém, apenas a propagação para frente foi apresentada. Nenhum mecanismo de propagação para trás foi aplicado ao modelo temporizado invertido foi considerado. A restrição de tempo dada pelos intervalos de visibilidade dos casos processados não era tão preciso e, como tal, não estavam adequadamente adaptados para uso no problema prático de planejamento/escalonamento de recursos, usado na implementação de atividades em um processo de *workflow* de tempo real.

Capítulo 3

Mecanismo de Propagação com Restrição de Tempo

Este capítulo apresenta uma nova abordagem para se calcular intervalos de visibilidade de datas relacionados a atividades de uma *Workflow Net*.

3.1 Mecanismo de Propagação

Em geral, o tempo necessário para se executar uma atividade em um processo de *workflow* é não determinístico. De acordo com (RIVIERE; PRADIN-CHEZALVIEL; VALETTE, 2001) restrições de tempo explícitas, que existem em sistemas com características de tempo real, podem ser formalmente especificados usando um intervalo de tempo estático associado a cada atividade do modelo. O comportamento dinâmico da rede de Petri temporal, portanto, depende das marcações da rede, assim como do contexto temporal relacionado à ficha, que é dado pelo intervalo de visibilidade (SOARES; JULIA; VRANCKEN, 2008). Um intervalo de visibilidade $[(\delta_p)_{min}, (\delta_p)_{max}]$ associado a uma ficha em um lugar p de uma rede de Petri temporal especifica a data mínima na qual uma ficha está disponível em p para disparar uma transição de saída de p (data de início ao mais cedo de uma atividade - $(\delta_p)_{min}$) e a data máxima depois da qual toda ficha se torna indisponível (morta) e não pode ser utilizada para disparar nenhuma transição (data de início de uma atividade ao mais tarde - $(\delta_p)_{max}$).

Em um Sistema de Gerenciamento de *Workflow*, o intervalo de visibilidade depende de um contador global conectado a toda a rede e que calcula a passagem do tempo a partir do momento $\delta = 0$, o que corresponde ao início das operações do sistema. Em particular, o tempo de espera entre atividades sequenciais será representado por um intervalo de visibilidade, no qual os limites mínimo e máximo dependem da entrega ao mais cedo e ao mais tarde de um caso por uma *Workflow net*. A partir de uma data de início conhecida e do conhecimento da duração de um caso, é possível calcular os intervalos de visibilidade associados às fichas nos lugares de espera.

O cálculo dos intervalos de visibilidade associados às fichas nos lugares de espera será realizado nessa proposta considerando os diferentes tipos de rotas que podem existir em um processo de *workflow*, os quais podem ser expressos por sequentes da Lógica Linear.

A técnica de propagação com restrição de tempo é baseada em duas abordagens distintas. A primeira, um raciocínio para frente, para produzir os limites mínimos de um intervalo de visibilidade (datas ao mais cedo para iniciar uma atividade do processo). A segunda abordagem, de raciocínio para trás, proposta pioneiramente neste trabalho, para se produzir os limites máximos dos intervalos de visibilidade (datas ao mais tarde para iniciar as atividades do processo). Os intervalos de visibilidade produzidos serão dados por expressões de data simbólicas ao invés de numéricas. A principal vantagem de se trabalhar com datas simbólicas é que uma vez calculadas para um processo de *workflow* específico, essas datas podem ser utilizadas para qualquer caso a ser tratado por esse mesmo processo, sem necessidade de recálculo.

Para este trabalho, D_i representa uma data e d_i uma duração associada ao disparo de uma transição t_i. Um par (D_p, D_c) será associado a cada ficha da árvore de prova e representa respectivamente a data de produção e de consumo da ficha. Como os disparos de transição são instantâneos em rede t-temporal (rede com intervalo de tempo associado às transições), a seguinte definição pode ser produzida:

Definição 8 (Data de produção e data de consumo de uma ficha)**.** *A data de produção D_p de uma ficha é igual à data de disparo da transição que a produziu e a data de consumo D_c é igual a data de disparo da transição que a consumiu.*

Para cada atividade disparada numa *Workflow net* t-temporal, devem ser extraídas as datas de produção (D_p) e de consumo (D_c) dos átomos que representam as pré-condições da transição correspondente a esta tarefa. Quando há mais de uma pré-condição associada à transição, considera-se a data máxima das produções dos átomos correspondentes a estas pré-condições. A data de produção de uma ficha (D_p) corresponde ao início da execução da atividade associada à transição e a data de consumo (D_c) corresponde à sua conclusão. Dessa forma, um intervalo $[D_p, D_c]$ será gerado e o recurso destinado a determinada atividade poderá ser requisitado dentro do intervalo calculado.

Como as datas de produção e consumo dependem de d_i (duração associada ao disparo de uma transição) e obtem seus valores respeitando um intervalo de tempo $\Delta_i = [\delta_{imin}, \delta_{imax}]$, muitos intervalos de execução de tarefas podem ser considerados de acordo com um planejamento estratégico. Em outras palavras, um intervalo de execução $I_{Exec} = [D_{Pmin}, D_{Cmax}]$ considera que a alocação de recursos para lidar com uma tarefa pode ocorrer entre o início ao mais cedo de uma atividade e sua conclusão ao mais tarde. Para ilustrar essa proposta de trabalho, será considerada a *Workflow net* apresentada na Figura 11.

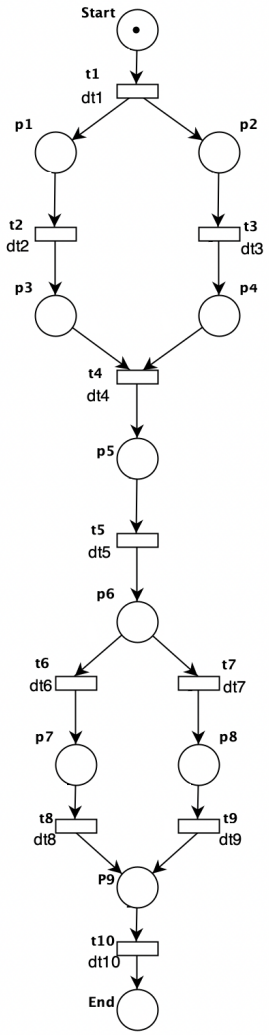


Figura 16 – *Workflow net* t-temporal - Propagação para frente.

3.1.1 Mecanismo de Propagação Para Frente

O mecanismo de propagação para frente considera uma *Workflow net* em seu fluxo normal, que é um caso representado por uma ficha que começa no lugar *Start* e termina no lugar *End*. Entre o lugar de início e termino do processo, a ficha percorre uma sequência ordenada de lugares e transições.

O cálculo das datas derivadas da prova canônica do sequente da Lógica Linear para uma *Workflow net* é a seguinte:

a) atribuir uma data de produção D_i para a ficha inicial no lugar *Start* (marcação inicial do modelo de rede de Petri t-temporal);

b) para cada aplicação da regra $\multimap L$, calcule a data de disparo da transição correspondente, isto é igual ao valor máximo da data de produção entre os átomos consumidos por esta transição, somado à duração d_i de sensibilização da transição considerada;

c) atualizar as datas de todas os átomos produzidos e consumidos.

Considerando a Figura 16, para verificar o critério de correção *Soundness* (verificar a acessibilidade de todos os caminhos que levam um caso do lugar *Start* de início até o lugar *End* de fim do processo), dois sequentes lineares devem ser provados, os quais representam cenários diferentes. Estes sequentes lineares podem ser obtidos automaticamente encontrando os invariantes de transição (t-invariante) resolvendo um sistema de equações lineares como proposto em (OLIVEIRA; JULIA, 2020).

Em determinado ponto da *Workflow net* representada pela Figura 11, existe uma rota condicional (ou rota seletiva), onde o caso em execução deve optar por executar uma de duas possíveis atividades: $A6(Pay)$ ou $A7(Send\_Letter)$. Na Figura 16 essa mesma rota condicional é representada pelas tansições $t6$ e $t7$. Dessa forma, cada uma dessas atividades será contemplada em um cenário distinto, sendo o cenário 1 (S_{c1}) o que abrange a transição $t6$ e o cenário 2 (S_{c2}) o que abrange a transição $t7$.

Para o cenário S_{c1}, o seguinte sequente deve ser provado:

$$Start, t1, t2, t3, t4, t5, t6, t8, t10 \vdash End \tag{13}$$

Para o cenário S_{c2}, o seguinte sequente deve ser provado:

$$Start, t1, t2, t3, t4, t5, t7, t9, t10 \vdash End \tag{14}$$

As transições da *Workflow net* são representadas pelas seguintes formas da Lógica Linear:
$t_1 = Start \multimap p1 \otimes p2$,
$t_2 = p1 \multimap p3$,
$t_3 = p2 \multimap p4$,
$t_4 = p3 \otimes p4 \multimap p5$,
$t_5 = p5 \multimap p6$,
$t_6 = p6 \multimap p7$,
$t_7 = p6 \multimap p8$,
$t_8 = p7 \multimap p9$,
$t_9 = p8 \multimap p9$,
$t_{10} = p9 \multimap End$.

Considerando $Seq = D_S + d_{t1} + max\{d_{t2}, d_{t3}\} + d_{t4}$, e aplicando as régras do cálculo de sequente linear no primeiro cenário, é possível provar se o mesmo é ou não um sequente sintaticamente válido (*Sound*). A árvore de prova correspondente ao cenário S_{c1} é a seguinte:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p9 \vdash p9} \quad \dfrac{}{End \vdash End}\,{\multimap_L}}{p7 \vdash p7} \quad \dfrac{}{p9, p9 \multimap End \vdash End}\,{\multimap_L}}{p6 \vdash p6} \quad \dfrac{}{p7, p7 \multimap p9, t10 \vdash End}\,{\multimap_L}}{p5 \vdash p5} \quad \dfrac{}{p6, p6 \multimap p7, t8, t10 \vdash End}\,{\multimap_L}}{\dfrac{p3 \vdash p3 \quad p4 \vdash p4}{p3, p4 \vdash p3 \otimes p4}\,{\otimes_R}} \quad \dfrac{}{p5, p5 \multimap p6, t6, t8, t10 \vdash End}\,{\multimap_L}}{p2 \vdash p2} \quad \dfrac{}{p3, p4, p3 \otimes p4 \multimap p5, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{p1 \vdash p1} \quad \dfrac{}{p2, p3, p2 \multimap p4, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{} \quad \dfrac{}{p1, p2, p1 \multimap p3, t3, t4, t5, t6, t8, t10 \vdash End}\,{\otimes_L}}{Start \vdash Start} \quad \dfrac{}{p1 \otimes p2, p1 \multimap p3, t3, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{Start, Start \multimap p1 \otimes p2, t2, t3, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L} \tag{15}$$

A árvore de prova com datas correspondente ao cenário S_{c1} é a seguinte:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p9(Seq+d_{t5}+d_{t6}+d_{t8}, Seq+d_{t5}+d_{t6}+d_{t8}+d_{t10}) \vdash p9} \quad \dfrac{}{End(Seq+d_{t5}+d_{t6}+d_{t8}+d_{t10},.) \vdash End}\,{\multimap_L}}{p7(Seq+d_{t5}+d_{t6}, Seq+d_{t5}+d_{t6}+d_{t8}) \vdash p7} \quad \dfrac{}{p9(Seq+d_{t5}+d_{t6}+d_{t8},.), p9 \multimap End \vdash End}\,{\multimap_L}}{p6(Seq+d_{t5}, Seq+d_{t5}+d_{t6}) \vdash p6} \quad \dfrac{}{p7(Seq+d_{t5}+d_{t6},.), p7 \multimap p9, t10 \vdash End}\,{\multimap_L}}{p5(Seq, Seq+d_{t5}) \vdash p5} \quad \dfrac{}{p6(Seq+d_{t5},.), p6 \multimap p7, t8, t10 \vdash End}\,{\multimap_L}}{\dfrac{p3(D_S+d_{t1}+d_{t2}, Seq) \vdash p3 \quad p4(D_S+d_{t1}+d_{t3}, Seq) \vdash p4}{p3(D_S+d_{t1}+d_{t2}, Seq), p4(D_S+d_{t1}+d_{t3}, Seq) \vdash p3 \otimes p4}\,{\otimes_R}} \quad \dfrac{}{p5(Seq,.), p5 \multimap p6, t6, t8, t10 \vdash End}\,{\multimap_L}}{p2(D_S+d_{t1}, D_S+d_{t1}+d_{t3}) \vdash p2} \quad \dfrac{}{p3(D_S+d_{t1}, +d_{t2},.), p4(D_S+d_{t1}, +d_{t3,.}), p3 \otimes p4 \multimap p5, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{p1(D_S+d_{t1}, D_S+d_{t1}+d_{t2}) \vdash p1} \quad \dfrac{}{p2(D_S+d_{t1},.), p3(D_S+d_{t1}+d_{t2},.), p2 \multimap p4, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{} \quad \dfrac{}{p1(D_S+d_{t1},.), p2(D_S+d_{t1},.), p1 \multimap p3, t3, t4, t5, t6, t8, t10 \vdash End}\,{\otimes_L}}{Start(D_S, D_S+d_{t1}) \vdash Start} \quad \dfrac{}{p1(D_S+d_{t1},.) \otimes p2(D_S+d_{t1},.), p1 \multimap p3, t3, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L}}{Start(Ds,.), Start \multimap p1 \otimes p2, t2, t3, t4, t5, t6, t8, t10 \vdash End}\,{\multimap_L} \tag{16}$$

Todos os nós folha da árvore de prova acima são sequentes identidade, ou seja, sequentes do tipo $A \vdash A$. Dessa forma, tem-se que o sequente linear $Start, t1, t2, t3, t4, t5, t6, t8, t10 \vdash End$, que representa o primeiro cenário da *Workflow net* apresentada na Figura 16 é sintaticamente válido (*Sound*).

As datas simbólicas obtidas a partir da árvore de prova do mecanismo de propagação para frente são apresentadas na Tabela 4.

3.1.2 Mecanismo de Propagação para Trás

A partir da árvore de prova obtida pelo modelo de propagação para frente e considerando a data de início D_S, é possível determinar parte do intervalo de visibilidade para disparar atividades (representado por uma ficha em p_i). Assim como em (PASSOS;

Tabela 4 – Intervalo simbólico de datas na propagação em avanço para o cenário S_{c1}.

Átomo	**Data de Produção**	**Data de Consumo**
Start	D_s	$D_s + d_{t1}$
$p1$	$D_s + d_{t1}$	$D_s + d_{t1} + d_{t2}$
$p2$	$D_s + d_{t1}$	$D_s + d_{t1} + d_{t3}$
$p3$	$D_s + d_{t1} + d_{t2}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$
$p4$	$D_s + d_{t1} + d_{t3}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$
$p5$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5}$
$p6$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6}$
$p7$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8}$
$p9$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8}$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8} + d_{t10}$
End	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8} + d_{t10}$	D_{End}

JULIA, 2016), o mecanismo de propagação para frente nos permite encontrar os limites mínimos dos intervalos de visibilidade.

A partir dos cálculos e resultados obtidos com o modelo de propagação para frente, é proposta então um novo modelo de propagação para calcular os limites máximos do mesmo intervalo de visibilidade, o modelo de propagação para trás. Essa abordagem proposta tem como justificativa o reaproveitamento dos cálculos já proposto para os limites mínimos do intervalo com algumas modificações que permitam que agora sejam calculados os limites máximos; além do fato de que a partir desses cálculos é possível obter datas simbólicas para os casos e não mais numéricas como já apresentado na literatura.

Para atingir esse objetivo, a abordagem é baseada em um modelo parecido com o apresentado em (KHALFHOUI et al., 2002; OLIVEIRA; JULIA, 2020). Em (KHALFHOUI et al., 2002) um raciocínio por retrocesso (para trás) foi aplicado em um modelo de rede de Petri com todos os arcos invertidos para identificar cenários temidos em sistemas mecatrônicos. O trabalho (OLIVEIRA; JULIA, 2020) também propõe um raciocínio por retrocesso aplicado a uma *Workflow net* com todos os arcos invertidos, a fim de diagnosticar as causas de situações de bloqueio (*deadlock*) em Arquiteturas Orientadas a Serviço.

Considerando a nova técnica de propagação para trás apresentada neste trabalho, todos os arcos da *Workflow net* são então invertidos de forma que a marcação inicial é apresentada no lugar *End* e atinge o lugar *Start* ao final da execução ou do processamento do caso da *Workflow net* invertida, como mostra a Figura 17.

A prova do sequente na *Workflow net* invertida é obtida utilizando o mesmo algoritmo de prova apresentado em (PASSOS; JULIA, 2016) para as regras aplicadas da Lógica Linear, porém com uma diferença no cálculo das marcas de tempo associadas às fichas produzidas.

Consideramos D_i utilizado para representar datas e d_i para durações. Quando a data de consumo não é completamente calculada, alguns pares como $(D_p, .)$ aparecerão na árvore de prova. Sequentes identidade gerados no lado esquerdo da árvore de prova produzem pares de marcas (D_p, D_c) completamente definidas. As marcas das folhas fi-

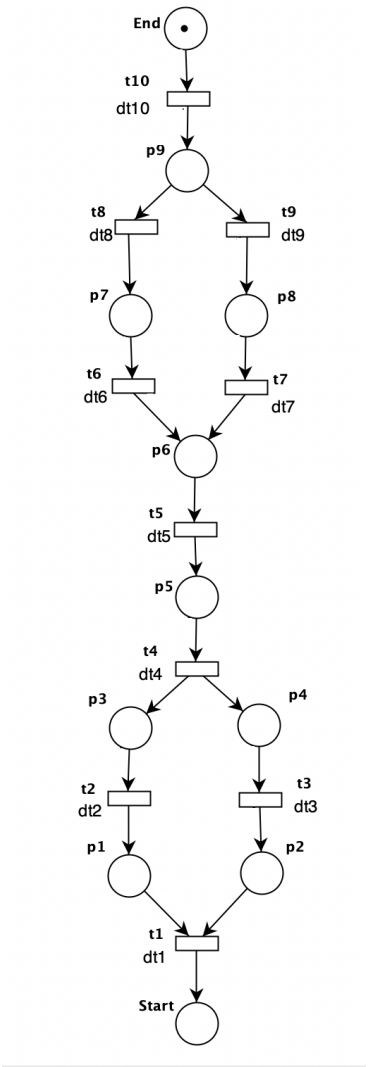


Figura 17 – *Workflow net* t-temporal - Propagação em retrocesso.

nais correspondem aos principais resultados temporais (limite máximo dos intervalos de visibilidade) para o mecanismo de propagação em retrocesso.

O cálculo das datas relativas à prova canônica do sequente da Lógica Linear para uma *Workflow net* é a seguinte:

a) atribuir uma data de produção D_i para a ficha inicial no lugar *End* (marcação inicial do modelo de rede invertida);

b) para cada aplicação da regra $\multimap L$, calcule a data de disparo da transição correspondente, isto é igual ao valor mínimo da data de produção entre os átomos consumidos por esta transição, menos a duração d_i de sensibilização da transição considerada;

c) atualizar as datas de todas os átomos produzidos e consumidos.

O restante do processo para provar o sequente é o mesmo que o apresentado para o mecanismo de propagação para frente.

Considerando a *Workflow net* da Figura 17, a prova do critério de validade sintática

(*Sound*) no modelo invertido corresponde a dois diferentes cenários dados pelos seguintes sequentes:

Para o cenário S_{c1}:

$$End, t10, t8, t6, t5, t4, t3, t2, t1 \vdash Start \tag{17}$$

Para o cenário S_{c2}:

$$End, t10, t9, t7, t5, t4, t3, t2, t1 \vdash Start \tag{18}$$

As transições da *Workflow net* são representadas pelas seguintes formas da Lógica Linear:
$t_{10} = End \multimap p9$,
$t_9 = p9 \multimap p8$,
$t_8 = p9 \multimap p7$,
$t_7 = p8 \multimap p6$,
$t_6 = p7 \multimap p6$,
$t_5 = p6 \multimap p5$,
$t_4 = p5 \multimap p3 \otimes p4$,
$t_3 = p4 \multimap p2$,
$t_2 = p3 \multimap p1$,
$t_1 = p1 \otimes p2 \multimap Start$.

Considerando $Seq_1 = D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$, a árvore de provas para o cenário S_{c1} é a seguinte:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\frac{p1\vdash p1 \quad p2\vdash p2}{p1,p2\vdash p1\otimes p2}\otimes_R \quad Start\vdash Start}{p3\vdash p3 \quad p1,p2,p1\otimes p2\multimap Start\vdash Start}\multimap_L}{p4\vdash p4 \quad p3,p2,p3\multimap p1,t1\vdash Start}\multimap_L}{p4,p3,p4\multimap p2,t2,t1\vdash Start}\otimes_L}{p5\vdash p5 \quad p3\otimes p4,p4\multimap p2,t2,t1\vdash Start}\multimap_L}{p6\vdash p6 \quad p5,p5\multimap p3\otimes p4,t3,t2,t1\vdash Start}\multimap_L}{p7\vdash p7 \quad p6,p6\multimap p5,t4,t3,t2,t1\vdash Start}\multimap_L}{p9\vdash p9 \quad p7,p7\multimap p6,t5,t4,t3,t2,t1\vdash Start}\multimap_L}{End\vdash End \quad p9,p9\multimap p7,t6,t5,t4,t3,t2,t1\vdash Start}\multimap_L}{End,End\multimap p9,t8,t6,t5,t4,t3,t2,t1\vdash Start}\multimap_L \tag{19}$$

A árvore de prova com datas correspondente ao cenário S_{c1} é a seguinte:

$$\dfrac{\dfrac{\dfrac{p1(Seq_1-d_{t2},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1 \quad p2(Seq_1-d_{t3},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p2}{p1(Seq_1-d_{t2},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1}),p2(Seq_1-d_{t3},Seq_1-min\{d_{t2},d_{t3}\}-d_{t1})\vdash p1\otimes p2}\otimes_R \quad Star(Seq_1-min\{d_{t2},d_{t3}\}-d_{t1},.)t\vdash Start}{p3(Seq_1,Seq_1-d_{t2})\vdash p3 \quad p1(Seq_1-d_{t2},.),p2(Seq_1-d_{t3},.),p1\otimes p2\multimap Start\vdash Start}\multimap_L$$

$$\dfrac{p4(Seq_1,Seq_1-d_{t3})\vdash p4 \quad p3(Seq_1,.),p2(Seq_1-d_{t3},.),p3\multimap p1,t1\vdash Start}{p4(Seq_1,.),p3(Seq_1,.),p4\multimap p2,t2,t1\vdash Start}\multimap_L$$

$$\dfrac{p5(D_{End}-d_{t10}-d_{t8}-d_{t6}-d_{t5},Seq_1)\vdash p5 \quad p3\otimes p4,p4\multimap p2,t2,t1\vdash Start}{p6(D_{End}-d_{t10}-d_{t8}-d_{t6},D_{End}-d_{t10}-d_{t8}-d_{t6}-d_{t5})\vdash p6 \quad p5(D_{End}-d_{t10}-d_{t8}-d_{t6}-d_{t5},.),p5\multimap p3\otimes p4,t3,t2,t1\vdash Start}\multimap_L$$

$$\dfrac{p7(D_{End}-d_{t10}-d_{t8},D_{End}-d_{t10}-d_{t8}-d_{t6})\vdash p7 \quad p6(D_{End}-d_{t10}-d_{t8}-d_{t6},.),p6\multimap p5,t4,t3,t2,t1\vdash Start}{p9(D_{End}-d_{t10},D_{End}-d_{t10}-d_{t8})\vdash p9 \quad p7(D_{End}-d_{t10}-d_{t8},.),p7\multimap p6,t5,t4,t3,t2,t1\vdash Start}\multimap_L$$

$$\dfrac{\otimes_L}{\multimap_L}\multimap_L$$

$$\dfrac{End(D_{End},D_{End}-d_{t10})\vdash End \quad p9(D_{End}-d_{t10},.),p9\multimap p7,t6,t5,t4,t3,t2,t1\vdash Start}{End(D_{End},.),End\multimap p9,t8,t6,t5,t4,t3,t2,t1\vdash Start}\multimap_L \tag{20}$$

Todos os nós folha da árvore de prova acima são sequentes identidade, ou seja, sequentes do tipo $A \vdash A$. Dessa forma, tem-se que o sequente linear $t10, t8, t6, t5, t4, t3, t2, t1 \vdash Start$, que representa o primeiro cenário da *Workflow net* apresentada na Figura 17 é sintaticamente válido (*Sound*).

As datas simbólicas obtidas a partir da árvore de prova do mecanismo de propagação em retrocesso são apresentadas na Tabela 5.

Tabela 5 – Intervalo simbólico de datas na propagação para trás.

Átomo	**Data de Produção**	**Data de Consumo**
End	D_{End}	$D_{End} - d_{t10}$
$p9$	$D_{End} - d_{t10}$	$D_{End} - d_{t10} - d_{t8}$
$p7$	$D_{End} - d_{t10} - d_{t8}$	$D_{End} - d_{t10} - d_{t8} - d_{t6}$
$p6$	$D_{End} - d_{t10} - d_{t8} - d_{t6}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5}$
$p5$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$
$p4$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t3}$
$p3$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t2}$
$p2$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t3}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$
$p1$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t2}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$
$Start$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$	D_s

Após os resultados obtidos pelos cálculos dos sequentes da Lógica Linear para os mecanismos de propagação para frente e para trás, os intervalos de visibilidade simbólicos podem ser produzidos para cada lugar da *Workflow net*. Os limites mínimos correspondem à data de produção da Tabela 4 e os limites máximos correspondem à data de produção da Tabela 5. Os resultados são apresentados na Tabela 6.

Considerando que um caso do processo representado na Figura 11 inicia na data 0 ($D_S = 0$), o tempo limite para que ele chegue ao lugar final e complete o processo é 105,

Tabela 6 – Intervalos de visibilidade simbólicos.

Átomo	**Limite Mínimo do Intervalo de Visibilidade**	**Limite Máximo do Intervalo de Visibilidade**
Start	D_s	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - min(d_{t2}, d_{t3}) - d_{t1}$
$p1$	$D_s + d_{t1}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t2}$
$p2$	$D_s + d_{t1}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4} - d_{t3}$
$p3$	$D_s + d_{t1} + d_{t2}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$
$p4$	$D_s + d_{t1} + d_{t3}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5} - d_{t4}$
$p5$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4}$	$D_{End} - d_{t10} - d_{t8} - d_{t6} - d_{t5}$
$p6$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5}$	$D_{End} - d_{t10} - d_{t8} - d_{t6}$
$p7$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6}$	$D_{End} - d_{t10} - d_{t8}$
$p9$	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8}$	$D_{End} - d_{t10}$
End	$D_s + d_{t1} + max(d_{t2}, d_{t3}) + d_{t4} + d_{t5} + d_{t6} + d_{t8} + d_{t10}$	D_{End}

e a duração de cada operação é apresentada na Tabela 7. Os intervalos de visibilidade para execução das atividades podem ser calculados substituindo as datas simbólicas de produção apresentadas nas Tabelas 4 e 5 pelos valores numéricos apresentados na Tabela 7. As durações mínimas das operaçãos (limites mínimos dos intervalos de duração apresentados na Tabela 7) são utilizadsa para substituir as datas de produção simbólicas produzidas pelo mecanismo de propagação para frente (Tabela 4); as durações mázimas das operaçãos (limites máximos dos intervalos de duração apresentados na Tabela 7) são utilizadsa para substituir as datas de produção simbólicas produzidas pelo mecanismo de propagação para trás (Tabela 5.

Tabela 7 – Intervalo de duração das transições.

Transição	**Duração Imprecisa**
t1	[5,10]
t2	[20,30]
t3	[25,35]
t4	[0,0]
t5	[0,0]
t6	[15,25]
t7	[15,25]
t8	[5,15]
t9	[20,30]
t10	[0,0]

Dessa forma, é possível calcular os intervalos de visibilidade considerando os valores das datas de produção dados na Tabela 4 para o mecanismo de propagação para frente como o limite mínimo do intervalo de visibilidade, já que este simula o fluxo normal de propagação de um caso em um processo; as datas de produção dadas na Tabela 5 para o mecanimso de propagação para trás, é considerado como o limite máximo do intervalo de visibilidade, já que simula o fluxo de propagação inverso de um caso em um processo. Como os intervalos de visibilidade simbólicos foram calculados apenas para o cenário S_c1, o exemplo de resultado numérico também foi calculado somente para este cenário. O

resultado é apresentado na Tabela 8.

Tabela 8 – Intervalos de visibilidade numéricos.

Átomo	**Mínimo**	**Máximo**
Start	0	20
p1	5	35
p2	5	30
p3	25	65
p4	30	65
p5	30	65
p6	30	65
p7	45	90
p9	50	105
End	50	105

Como pode ser visto na Tabela 8, considerando o limite máximo para finalização do processo pelo caso com data de início igual a 0, $105 + 0 = 105$ e as fórmulas simbólicas produzidas pelo mecanismo de propagação para trás, é possível alcançar o lugar *Start* (a partir do lugar *End* na data 20. Sendo as datas obtidas pelo mecanismo de propagação para trás referentes ao limite máximo dos intervalos de visibilidade, os valores de datas numéricos apontam que mesmo que o processo que se inicia no tempo 0 tivesse um atraso até o tempo 20 para o disparo de $t1$, mesmo assim ele pode finalizar o processo dentro do tempo limite que é de no máximo 105.

A fim de validar esta abordagem, pode-se observar que os intervalos de visibilidade apresentados na Tabela 8 são semelhantes aos apresentados em (SOARES; JULIA; VRANCKEN, 2008) no qual os intervalos de visibilidade foram produzidos para a mesma *Workflow net* t-temporal utilizando uma abordagem puramente numérica baseada no grafo do modelo de rede de Petri correspondente.

Capítulo 4

Análise e Experimentos para o Dimensionamento de Recursos

Este capítulo apresenta alguns mecanismos de alocação de recursos(Seção 4.1), além da modelagem da *Workflow net* referente ao "Processo de Tratamento de Reclamações" (AALST; HEE; HEE, 2004), apresentado na seção 2.1.4 utilizando alocação de recursos, na ferramenta CPN Tool (Seção 4.2). Além da modelagem, serão apresentados experimentos feitos na *Workflow net* e análise dos dados obtidos (Seção 4.3).

4.1 Mecanismos de Alocação de Recursos em *Workflow net*

Existem diferentes tipos de recursos que podem ser utilizados para a realização de atividades de um processo de *workflow*. Alguns dos recursos utilizados podem ser considerados do tipo discreto, podendo ser representados por fichas simples. Um exemplo disso seria uma impressora utilizada para tratar documentos, de forma que só pode ser alocada para um único documento em determinado momento. Porém existem alguns recursos que não podem ser representados por uma ficha simples, que é o caso da maioria dos recursos do tipo humano, dado que um funcionário que trabalha em um escritório, por exemplo, pode tratar vários casos simultaneamente. Um outro exemplo seria o caso de uma enfermeira que pode ser alocada para cuidar de vários pacientes ao mesmo tempo durante seu dia de trabalho. A seguir será apresentada a definição formal para os mecanismos de alocação de recursos considerados neste trabalho.

4.1.1 Mecanismo de Alocação de Recurso Discreto

A fim de definir os mecanismos de alocação de recursos, os termos utilizados seguem as definições apresentadas em (JESKE et al., 2006). A seguir é apresentada a definição formal do mecanismo de alocação de recurso discreto.

Definição 9 (Mecanismo de Alocação de Recurso Discreto)**.** *O mecanismo de alocação de recurso discreto pode ser definido formalmente por um modelo de rede de Petri ordinária marcada* $C_{DR} =< A_{DR}, T_{DR}, Pre_{DR}, Pos_{DR}, M_{DR} >$ *com :*

a) $A_{DR} = \bigcup_{\alpha=1}^{N_{DR}} A_{(DR)_\alpha} \cup \{R_D\}$ *onde* R_D *representa o lugar do recurso discreto ,* $A_{(DR)_\alpha}$ *o lugar da atividade* α *e* N_{DR} *o número de atividades que estão conectadas com o lugar do recurso discreto* R_D*;*

b) $T_{DR} = \bigcup_{\alpha=1}^{N_{DR}} T_{in_\alpha} \cup \bigcup_{\alpha=1}^{N_{DR}} T_{out_\alpha}$ *onde* T_{in_α} *representa a transição de entrada da atividade* $A_{(DR)_\alpha}$ *e* T_{out_α} *representa a transição de saída da atividade* $A_{(DR)_\alpha}$*;*

c) $Pre_{DR} : A_{DR} \times T_{DR} \to \{0, 1\}$ *é a aplicação de incidência anterior tal que:*

$$Pre_{DR}(R_D, T_{in_\alpha}) = 1 \text{ e } Pre_{DR}(A_{(DR)_\alpha}, T_{out_\alpha}) = 1$$

(outras combinações de lugar/transição são iguais a zero);

d) $Pos_{DR} : A_{DR} \times T_{DR} \to \{0, 1\}$ *é a aplicação da incidência posterior tal que:*

$$Pos_{DR}(R_D, T_{out_\alpha}) = 1 \text{ e } Pos_{DR}(A_{(DR)_\alpha}, T_{in_\alpha}) = 1$$

(outras combinações de lugar/transição são iguais a zero);

e) $M_{DR} : A_{DR} \to N$ *é a marcação inicial tal que:*

$$M_{DR}(R_D) = m_D$$
$$e$$
$$M_{DR}(A_{(DR)_\alpha}) = 0 \text{ para } \alpha = 1 \text{ até } N_{DR}$$

onde M_{DR} *representa o número de recursos discretos do mesmo tipo.*

Supondo-se que no "Processo de Tratamento de Reclamações" apresentado anteriormente, um único funcionario trata as atividades "*Contactar Cliente*", "*Contactar Departamento*" e "*Enviar Carta*"; então a modelagem do mecanismo de alocação de recursos discretos é dado pela Figura 18.

Nesta Figura, se a ficha em R_D é utilizada para realizar uma atividade, $A2$, por exemplo, nenhuma outra atividade poderá ser realizada até que esta seja concluída e a ficha retorne para seu lugar de origem. Por exemplo, depois do início da atividade "*Contactar Cliente*", se o cliente não estiver disponível para responder as questões do funcionário, ele não pode abandonar a atividade enquanto espera e iniciar uma nova. Como o recurso é representado por uma única ficha simples, o funcionario deve concluir primeiro a tarefa de "*Contactar Cliente*" para só então iniciar uma nova atividade.

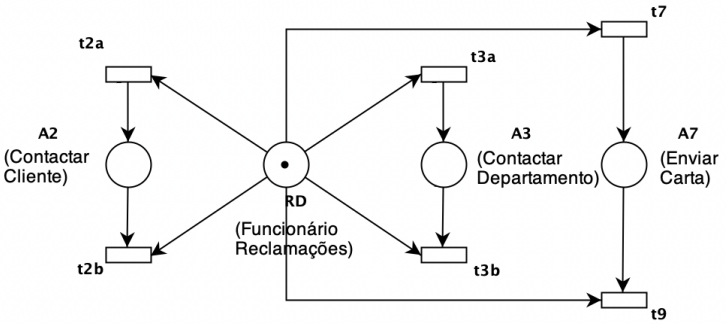


Figura 18 – Alocação de Recurso Discreto.

4.1.2 Mecanismo de Alocação de Recurso Contínuo

A fim de definir os mecanismos de alocação de recursos, os termos utilizados seguem as definições apresentadas em (JESKE et al., 2006). A seguir é apresentada a definição formal do mecanismo de alocação de recurso contínuo.

Definição 10 (Mecanismo de Alocação de Recurso Contínuo)**.** *O mecanismo de alocação de recurso contínuo pode ser definido formalmente por um modelo de rede de Petri híbrida marcada com transições discretas* $C_{CR} =< A_{CR}, T_{CR}, Pre_{CR}, Pos_{CR}, M_{CR} >$ *com:*

a) $A_{CR} = \bigcup_{\alpha=1}^{N_{CR}} A_{(CR)_\alpha} \cup \{R_C\}$ *onde* R_C *representa o lugar de recurso contínuo,* $A_{(CR)_\alpha}$ *o lugar da atividade* α *e* N_{CR} *o número de atividades que estão conectados com o lugar do recurso contínuo* R_C*;*

b) $T_{CR} = \bigcup_{\alpha=1}^{N_{CR}} T_{in_\alpha} \cup \bigcup_{\alpha=1}^{N_{CR}} T_{out_\alpha}$ *onde* T_{in_α} *representa a transição de entrada discreta da atividade* $A_{(CR)_\alpha}$ *e* T_{out_α} *representa a transição de saída discreta da atividade* $A_{(CR)_\alpha}$*;*

c) $Pre_{CR} : A_{CR} \times T_{CR} \to R^+$ *é a aplicação da incidência anterior tal que:*

$$Pre_{CR}(R_C, T_{in_\alpha}) = X_\alpha \text{ com } X_\alpha \in R^+ \text{ e } Pre_{CR}(A_{(CR)_\alpha}, T_{out_\alpha}) = 1$$

(outras combinações de lugar/transição são iguais a zero);

d) $Pos_{CR} : A_{CR} \times T_{CR} \to R^+$ *é a aplicação da incidência posterior tal que:*

$$Pos_{CR}(R_C, T_{out_\alpha}) = X_\alpha \text{ e } Pos_{CR}(A_{(CR)_\alpha}, T_{in_\alpha}) = 1$$

(outras combinações de lugar/transição são iguais a zero);

e) $M_{CR} : A_{(CR)_\alpha} \to R^+$ *é a marcação inicial tal que:*

$$M_{CR}(R_C) = m_C$$
$$e$$
$$M_{CR}(A_{(CR)_\alpha}) = 0 \text{ para } \alpha = 1 \text{ até } N_{CR}$$

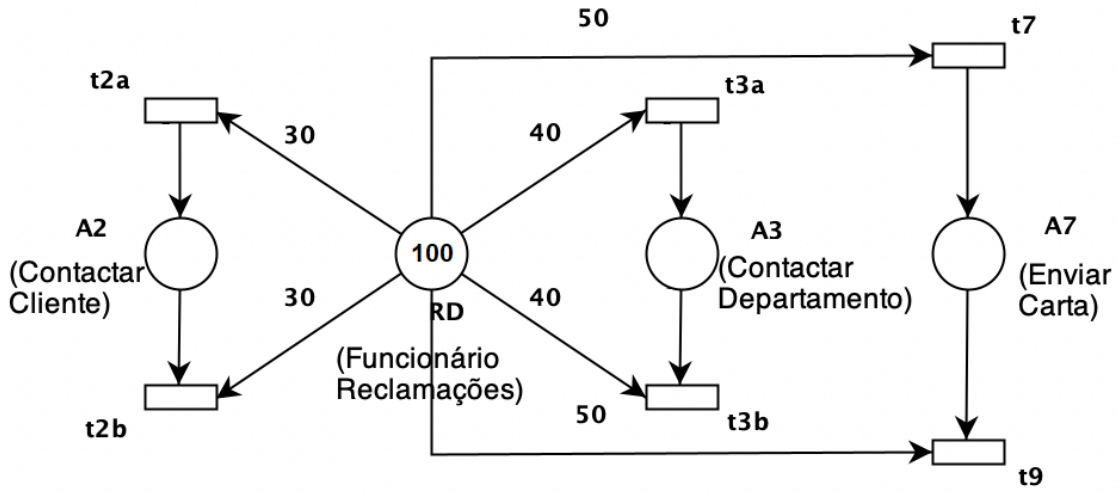


Figura 19 – Alocação de Recurso Contínuo.

onde M_{CR} representa a disponibilidade (em porcentagem) do recurso contínuo.

Supondo-se que no "Processo de Tratamento de Reclamações" apresentado anteriormente, um único funcionario trata as atividades "*Contactar Cliente*", "*Contactar Departamento*" e "*Enviar Carta*"; então a modelagem do mecanismo de alocação de recursos contínuo é dado pela Figura 19.

Nesta Figura 19, fica expecificado que somente 30% da disponibilidade do funcionário é necessária para realizar a atividade "*Contactar Cliente*", 40% para realizar a atividade "*Contactar Departamento*" e 50% para realizar a atividade "*Enviar Carta*". Dessa forma, é possivel então que um funcionário trate mais de uma atividade simultaneamente. Por exemplo, depois do início da atividade "*Contactar Cliente*", se o cliente não estiver disponível para responder as questões do funcionário, este último poderá usar sua disponibilidade de tempo (esperando a resposta do cliente) para realizar outra atividade como, por exemplo, "*Enviar Carta*", que requer 50% da disponibilidade do funcionário. Dessa forma, sabendo que o funcionário está empregando 50% de sua disponibilidade em enviar uma carta e 30% da disponibilidade em contactar o cliente, restaria ainda 20% de sua disponibilidade para ser alocada na execução de uma nova atividade.

A Figura 20 apresenta uma sequência de disparos envolvendo o recuso contínuo RD. Em 20a, as transições $t2a$, $t3a$ e $t7$, associadas ao recurso RD estão sensibilizadas e por isso estão representadas em vermelho. Dessa forma, qualquer uma delas pode ser escolhida para ser disparada. Após o disparo de $t2a$, por exemplo, temos o cenário apresentado na Figura 20b, na qual o recurso que RD que antes estava em 100, após o disparo, passou a ter valor 70, isso porque, como definido anteriormente, a ativiade representada por $t2a$ consome de forma simbólica 30 do recurso disponível. Outra observação importante é que apenas o recurso RD é contínuo, as atividades continuam sendo representadas de forma

discreta, e é por essa razão que, mesmo $t2a$ tendo consumido 30 do recurso RD, apenas uma ficha é gerada em $A2$. A Figura 20c representa um segundo disparo de transição, agora de $t3a$. A transição $t3a$ consome 40 do recurso. Como $t2a$ já tinha consumido 30, RD agora tem valor igual a 30. Assim como aconteceu em $A2$, $A3$ também é representada na forma discreta, e por isso é gerado somente uma ficha em $A2$.

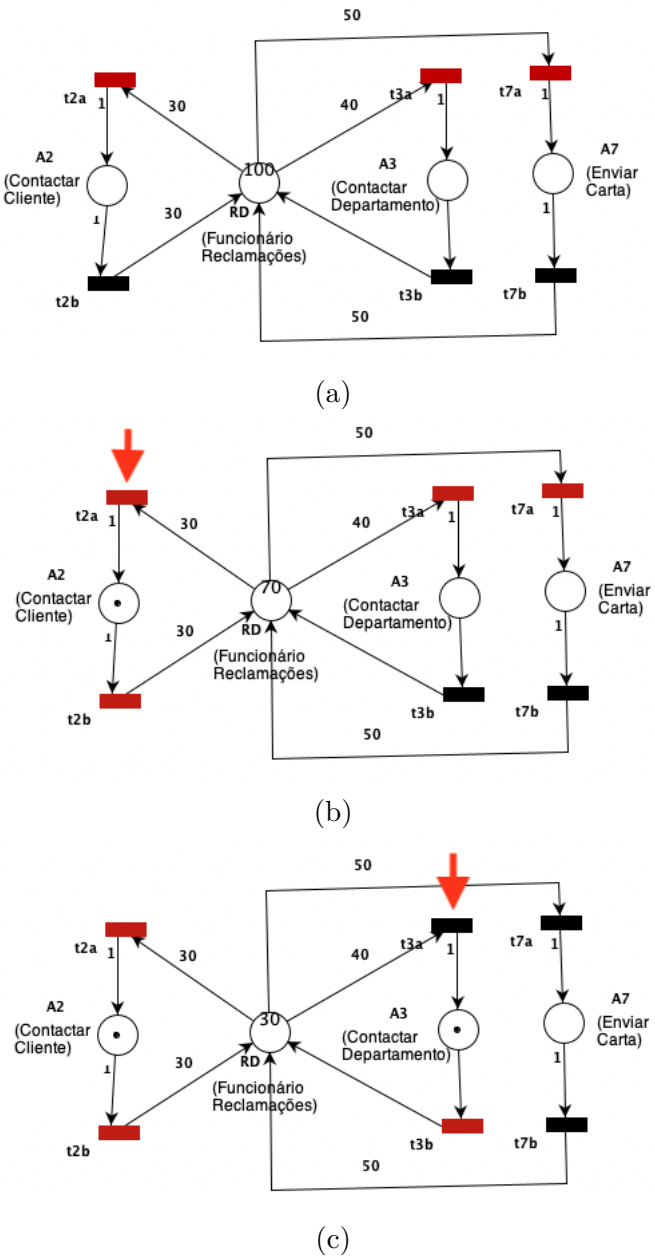


Figura 20 – Sequência de disparos envolvendo o recurso contínuo.

4.2 Implementação da *Workflow net* com Recursos no CPN Tools

O CPN Tools é uma ferramenta de uso comercial para a construção e análise de modelos CPN. Utilizando o CPN Tools é possível investigar o comportamento de um sistema modelado utilizando algoritmos para verificar certas propriedades. O CPN Tools é um *software* que oferece as funcionalidades de edição, simulação, análise de espaço dos estados e análise de um modelo CPN (JENSEN; KRISTENSEN, 2009).

Com o objetivo de mostrar a modelagem e implementação dos diversos mecanismos de alocação de recursos, será utilizada a ferramento computacional CPN Tools para o exemplo "Processo de Tratamento de Reclamações" definido em (AALST; HEE; HEE, 2004), apresentado na seção 2.1.4 e representado pela Figura 11.

A ferramenta CPN Tools permite desenvolver uma rede de Petri colorida de forma hierárquica, ou seja, é possível dividir o modelo em submodelos, sendo que cada um deles aparece em uma subpágina da área de trabalho do *software*. Cada subpágina está associada a uma transição do modelo principal.

Para definir a duração de cada atividade foi utilizada a distribuição uniforme (valor aleatório uniformemente distribuído considerando um intervalo). A duração total esperada para um caso é de 105 unidades de tempo (valor arbitrário que serve para ilustrar a abordagem proposta). A data de início do processo não é fixa, sendo que cada caso inicia o processamento em um instante definido de modo aleatório baseado na distribuição exponencial. Dessa forma, o tempo final máximo autorizado para conclusão do caso será calculado somando 105 ao tempo de início.

4.2.1 Implementação de uma *Workflow net* com Mecanismo de Alocação de Recursos Discretos

A Figura 21 apresenta o modelo para o "Processo de Tratamento de Reclamações" definido em (AALST; HEE; HEE, 2004) com inclusão explícita de mecanismos de recursos discretos e intervalos de durações de tempo (modelo p-temporal) apresentado em (JULIA; OLIVEIRA; VALETTE, 2008) associados às atividades do processo. Com base neste modelo, a implementação de uma *Workflow net* com mecanismos de alocação de recursos discretos baseada na linguagem CPN Tools será apresentada.

A implementação do modelo mostrado na Figura 21 foi construída usando CPN Hierárquica. A Figura 22 representa o modelo de processo principal e as Figuras 23 e 24 mostram os subprocessos modelados na ferramenta CPN Tools de forma hierárquica.

Cada atividade, representada por uma transição de substituição, só é habilitada caso o recurso associado a ela esteja disponível. Existe nesse processo um ponto de sincronização, de forma que o caso só pode avançar no processo caso as atividades de "*Contactar Cliente*"

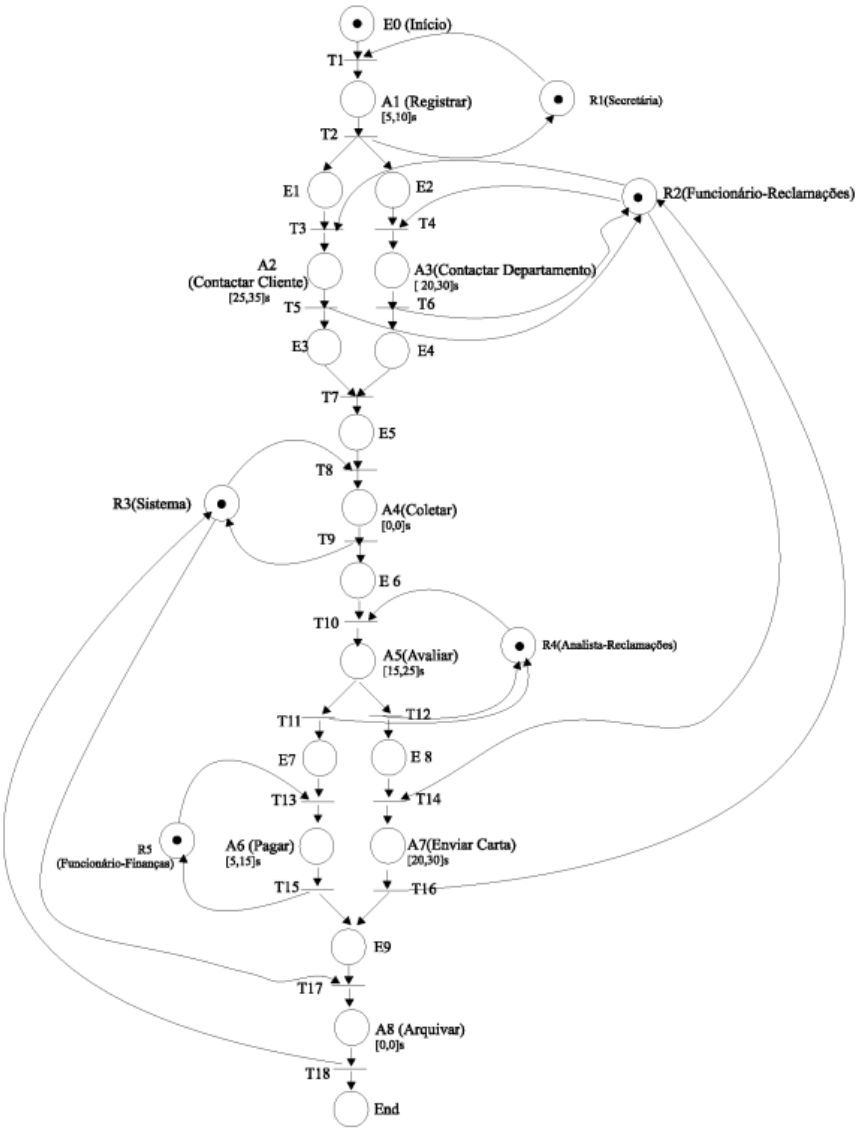


Figura 21 – Modelo para o Processo de Tratamento de Reclamações com Recursos Discretos.

e "*Contactar Departamento*" sejam concluidas. Essa situação é representada na implementação pela transição $Transition$ e a condição $if((\#1)tp1) > (\#1(tp2)\ then\ tp1\ else\ tp2$, que verifica o tempo de cada ficha e prossegue com o maior tempo do caso no momento do disparo.

A Tabela 9 apresenta algumas das principais declarações utilizadas no modelo.

De acordo com a Tabela 9 temos:

a) Tp: representa o tempo corrente;

b) t1: representa o tempo inicial do intervalo de uma atividade;

c) t2: representa o tempo final do intervalo de uma atividade;

d) transition: representa o id da transição que está relacionado à sua posição no processo, por exemplo, A1 tem id igual a 1 porque é a primeira transição ;

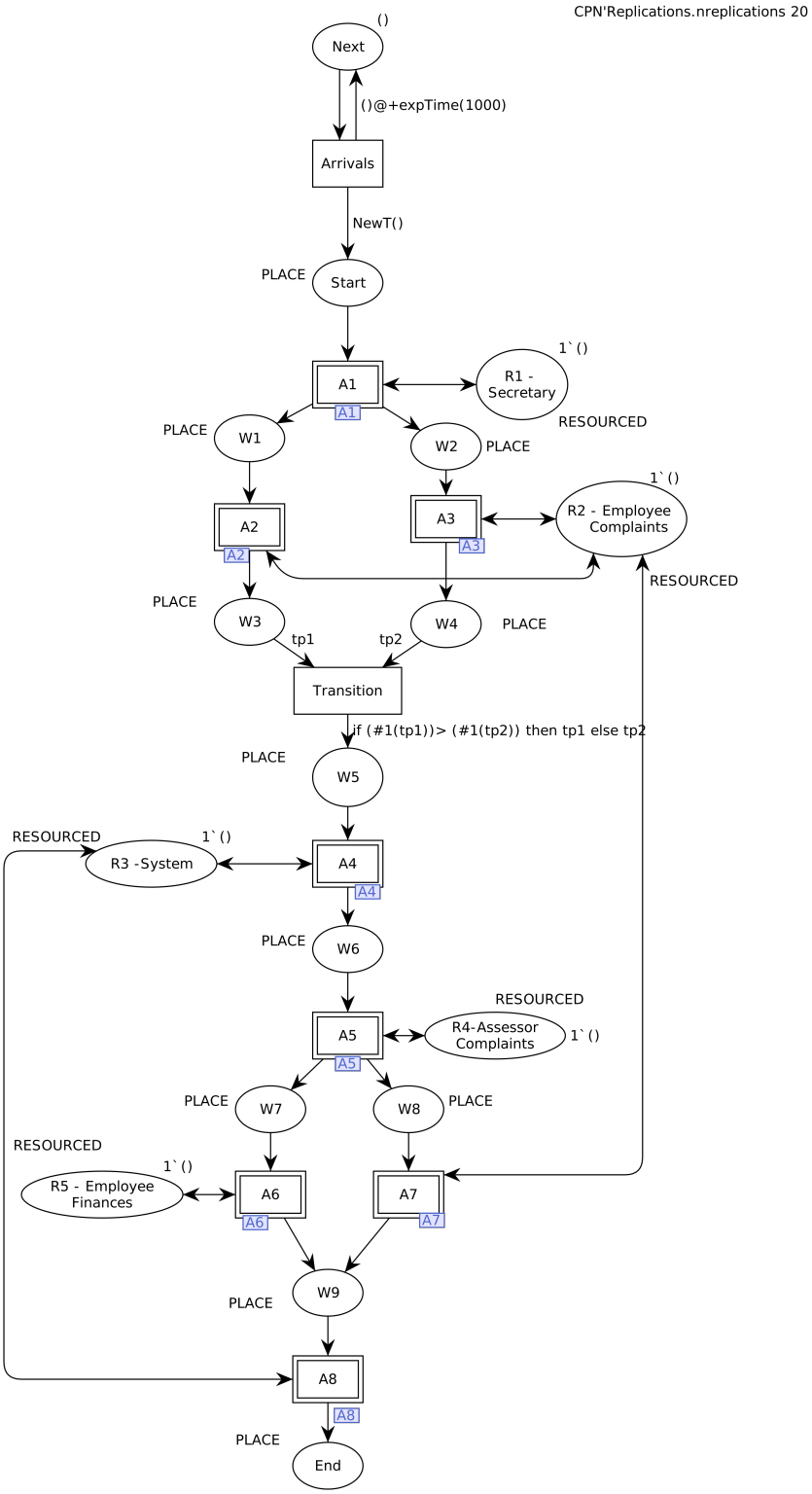


Figura 22 – Modelo para o Processo de Tratamento de Reclamações com Recursos Discretos.

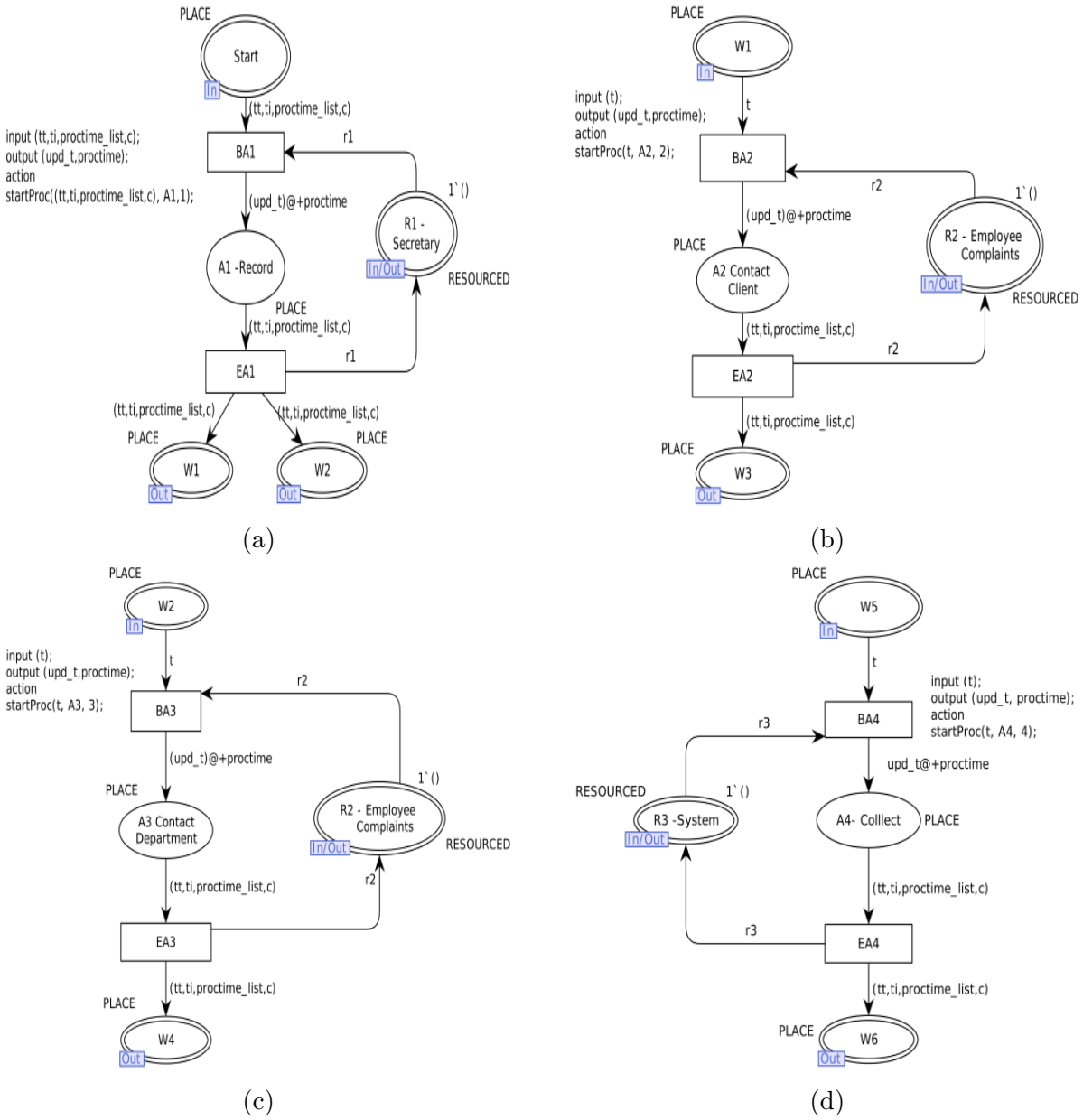


Figura 23 – Atividades A1, A2, A3 e A4 do Processo de Tratamento de Reclamações - Recursos Discretos.

e) T: representa um par de dados: id da transição e instante de disparo da transição;

f) RESOURCED: define o recurso discreto;

g) PROCTIMElist: representa uma lista de valores do tipo T;

h) CHOICE: define a escolha a ser feita - se o pagamento será efetuado ou se uma carta será enviada;

i) PLACE: representa a informação que cada ficha (caso) carrega: data de início do caso, tempo corrente, lista de tempos de processamento (data na qual uma transição t_i foi disparada, ou seja, momento no qual o caso (representado pela ficha) iniciou a execução de uma atividade e o cenário escolhido pelo caso (pagar

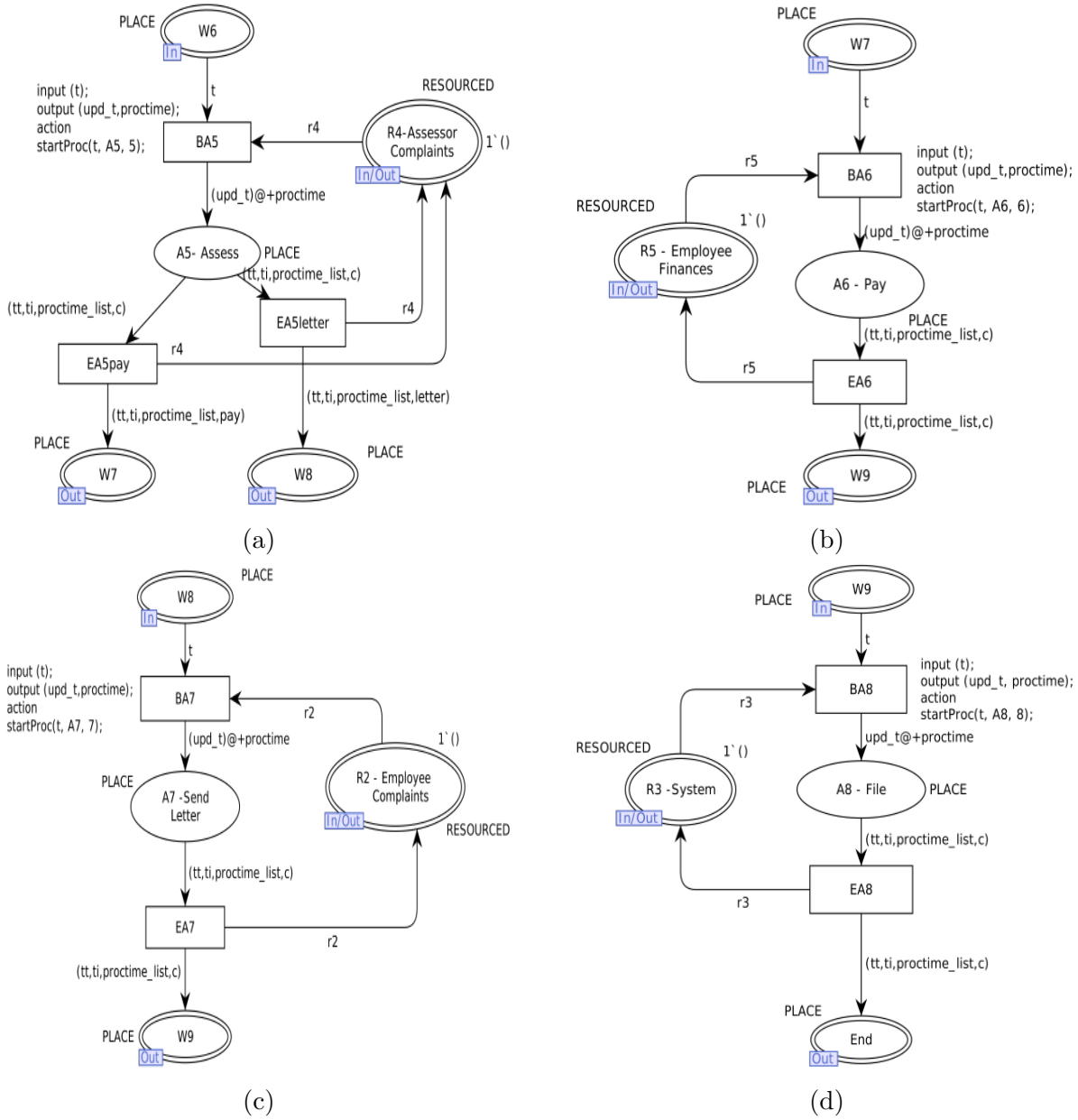


Figura 24 – Atividades A5, A6, A7 e A8 do Processo de Tratamento de Reclamações - Recursos Discretos.

ou enviar carta).

A Tabela 10 apresenta as variáveis utilizadas no modelo. Variáveis CPN têm a mesma característica de variáveis em qualquer linguagem de programação, tendo neste caso, seus tipos de dados representados por cores (*colset*).

A Tabela 11 apresenta as constantes definidas para o modelo. Neste exemplo estas constantes representam o intervalo de duração de cada atividade. As atividades $A4$ e $A8$ tem execução imediata, dessa forma, seu tempo de execução é igual a zero.

As subpáginas para as transições $A1$, $A2$, $A3$ e $A4$ são apresentadas na Figura 23. Nestas subpáginas, cada atividade é definida como um conjunto *transição-lugar-transição*. Quando o caso chega na primeira transição da atividade, é possível verificar se o recurso

Tabela 9 – Declarações para o Modelo CPN com Recursos Discretos.

Cores	**Valor**
colset Tp	INT
colset t1	REAL
colset t2	REAL
colset transition	INT
colset T	product transition*Tp
colset RESOURCED	UNIT
colset PROCTIMElist	list T
colset CHOICE	with pay | letter
colset PLACE	product Tp*Tp*PROCTIMElist*CHOICE timed

Tabela 10 – Variáveis para o Modelo CPN com Recursos Discretos.

Variáveis	**Tipo**
var t, upd_t, tp1, tp2	PLACE
var r1, r2, r3, r4, r5	RESOURCED
var c	CHOICE
var proctime_list	PROCTIMElist
var proctime, tt, ti	Tp

Tabela 11 – Constantes para o Modelo CPN com Recursos Discretos.

Constantes	**Valores**
val A1	(5.0,10.0)
val A2	(25.0,35.0)
val A3	(20.0,30.0)
val A4	(0.0,0.0)
val A5	(15.0,25.0)
val A6	(5.0,15.0)
val A7	(20.0,30.0)
val A8	(0.0,0.0)

está disponível. Se a transição é disparada, então o tempo de execução é atualizado somando o valor da duração da atividade ao tempo que está na ficha que representa o cliente ($(upd\_t)@ + proctime$). Quando a atividade termina, o recurso é devolvido.

As subpáginas para as transições $A5$, $A6$, $A7$ e $A8$ são apresentadas na Figura 24 e mostram um comportamento semelhante ao apresentado pelas atividades da Figura 23, com exceção da atividade $A5$, que apresenta um comportamento diferente por se tratar de uma transição de rota seletiva, na qual o caso deve escolher apenas um dos dois possíveis caminhos para continuar o processo, que no caso são pagar ou enviar uma carta. Se o caso segue pelo caminho $EA5pay$, o valor da variável c, que é do tipo *CHOICE*, passa a ser *pay*; caso o caminho escolhido seja pela transição $EA5letter$, o valor da variável c passa a ser $letter$.

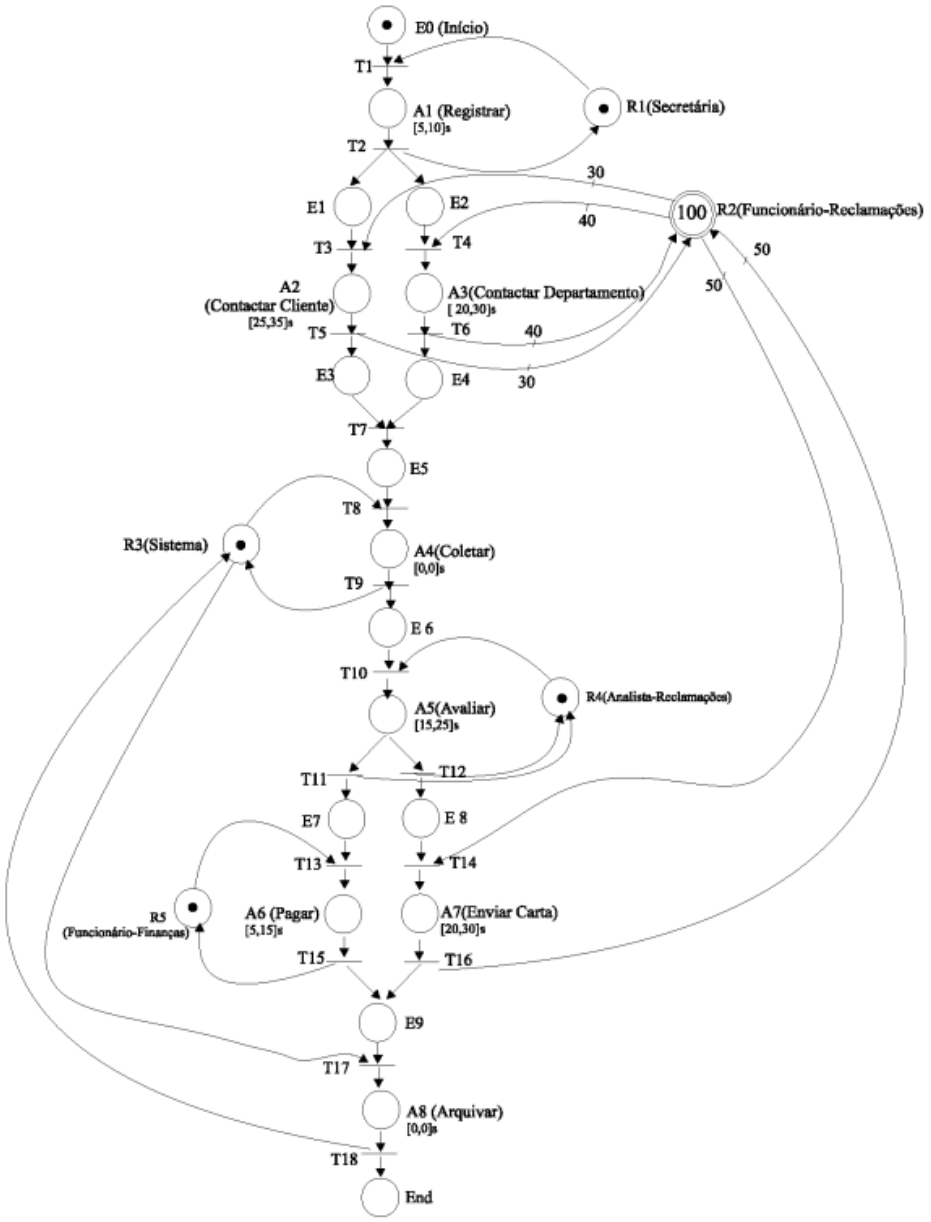


Figura 25 – Modelo para o Processo de Tratamento de Reclamações com Recursos Discretos e Contínuos.

4.2.2 Implementação de uma *Workflow net* com Mecanismo de Alocação de Recursos Discretos e Contínuos

A Figura 25 apresenta o modelo para o "Processo de Tratamento de Reclamações" definido em (AALST; HEE; HEE, 2004) com inclusão explícita de mecanismos de recursos discretos e contínuos e intervalos de durações de tempo (modelo p-temporal) apresentado em (JULIA; OLIVEIRA; VALETTE, 2008) associados às atividades do processo. Com base neste modelo, a implementação de uma *Workflow net* com mecanismos de alocação de recursos discretos e contínuos baseada na linguagem CPN Tools será apresentada.

A implementação do modelo mostrado na Figura 25 realizada na ferramenta CPN

Tools é apresentado na Figura 26, que representa o modelo de processo principal e as Figuras 27 e 28 mostram os subprocessos que utilizam um mecanismo de alocação de recursos discreto e contínuo.

Somente o recurso $R2$ é modificado para a forma contínua. Para além das definições já apresentadas na subseção 4.2.1, algumas novas declarações devem ser feitas para o recurso contínuo. A Tabela 12 apresenta as declarações adicionais deste modelo.

Tabela 12 – Declarações para o Modelo CPN com Recursos Discretos e Contínuos.

Declarações	**Valores**
colset RESOURCEC	INT
var R2	RESOURCEC
val RBA2	30
val RBA3	40
val RBA7	50

O funcionamento desse modelo é semelhante ao apresentado na subseção 4.2.1, exceto pelo mecanismo de alocação de recurso contínuo $R2$ que está associado às atividades $A2$, $A3$ e $A7$. O recurso contínuo é definido pelo *colset* $RESOURCEC$. A porcentagem do recurso necessária para a execução de cada atividade está representada nos arcos de entrada e saída do lugar recurso e especificada no modelo pelos valores constantes definidos em $RBA2$, $RBA3$ e $RBA7$.

As subpáginas para as transições relacionadas às atividades $A1$, $A2$, $A3$, $A4$, $A5$, $A6$, $A7$ e $A8$ são apresentadas nas Figuras 27 e 28 . As atividades 27b, 27c e 28c mostram as atividades $A2$, $A3$ e $A7$ respectivamente, nas quais o recurso alocado é o recurso contínuo $R2$.

Toda vez que uma transição é habilitada e o recurso $R2$ se faz necessário, é preciso validar a disponibilidade de determinada porcentagem do recurso para a execução da atividade correspondente. Para isso, é utilizada uma *Condição de Guarda* associada à transição que corresponde à atividade a ser executada, onde a disponibilidade do recurso deve ser maior ou igual ao peso do arco ($[r2 \geq RBA*$[1]$]$). Além disso, quando a transição é disparada, o valor de disponibilidade do recurso deve ser imediatamente atualizado. Se, por exemplo, a disponibilidade do recurso $R2$ (em porcentagem) for suficiente para o disparo de uma transição, o valor $RBA*$ associado ao arco de entrada da transição a ser disparada deve ser subtraído do valor de disponibilidade do recurso antes do disparo ($r2 - RBA*$). Ao final da execução da atividade que reservou parte da disponibilidade do recurso, o valor $RBA*$ associado ao arco de entrada do recurso deve ser somado ao valor de disponibilidade do recurso que fica associado com a ficha que se encontra no lugar $RESOURCEC(r2 + RBA*)$. No momento do disparo, um arco suplementar de retorno precisa ser conectado ao recurso para realizar a atualização de sua disponibilidade.

[1] * número da atividade

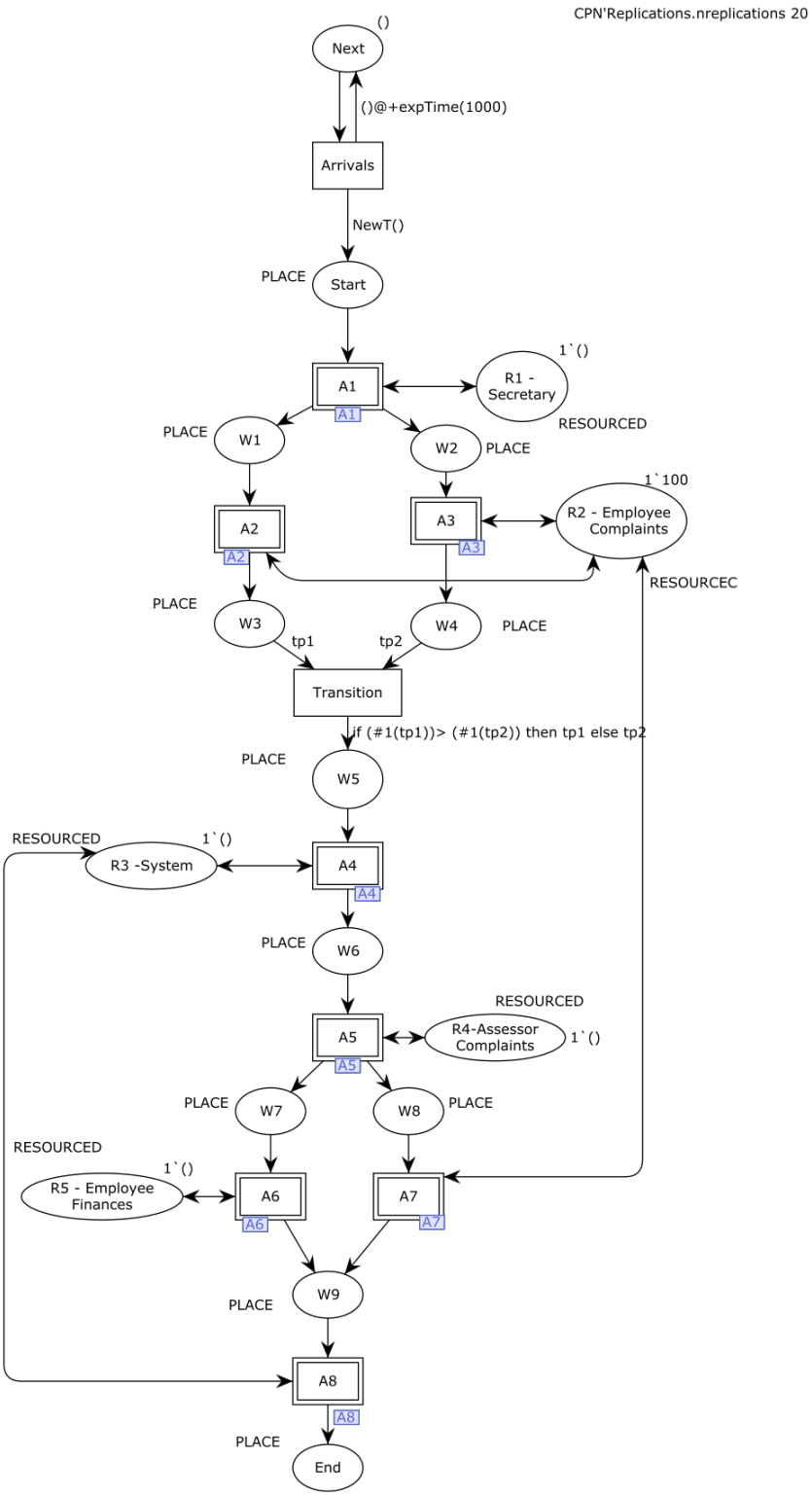


Figura 26 – Modelo para o Processo de Tratamento de Reclamações com Recursos Discretos e Contínuos.

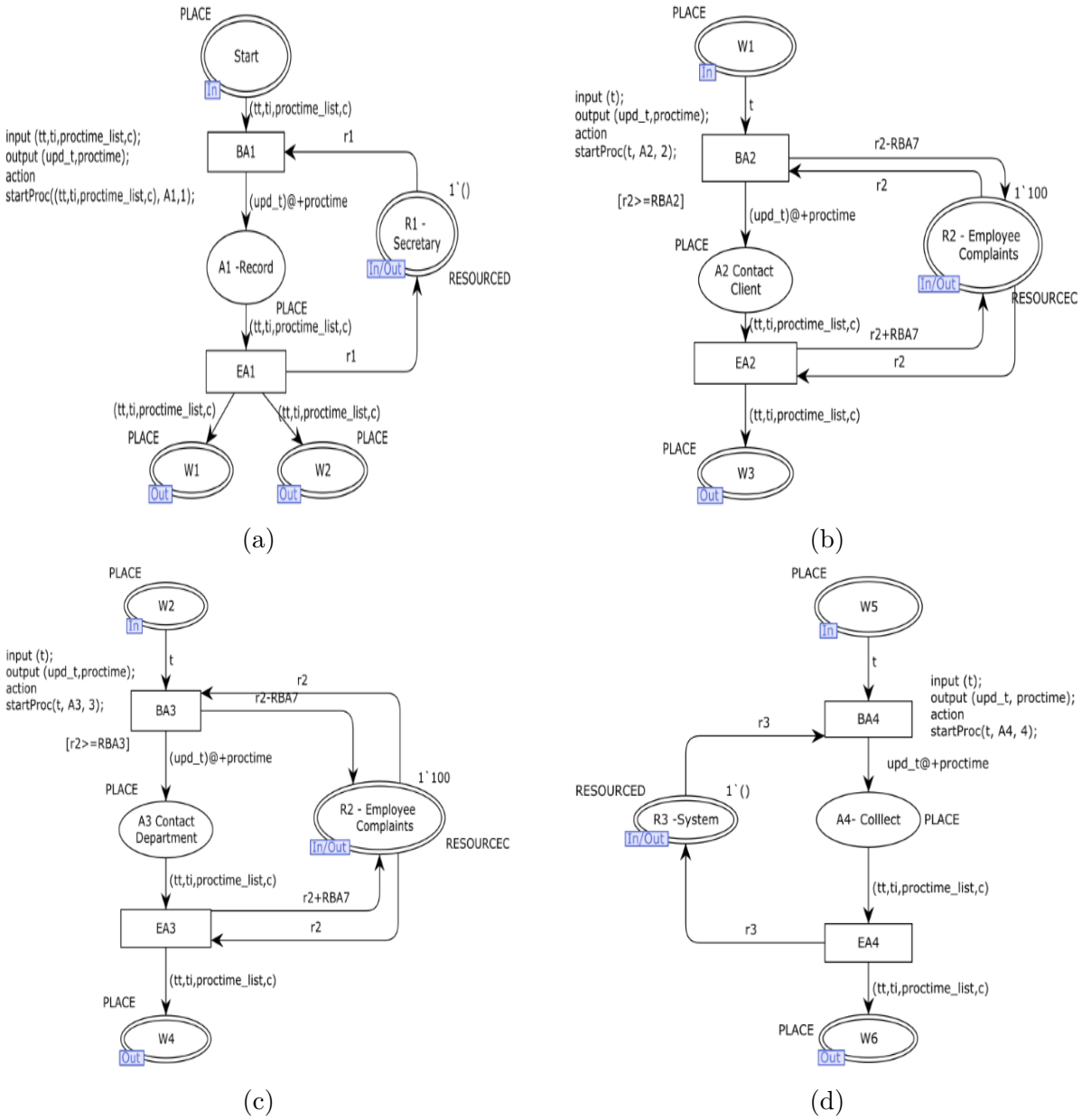


Figura 27 – Atividades A1, A2, A3 e A4 do Processo de Tratamento de Reclamações - Recursos Discretos e Contínuos.

4.3 Análise de Atraso em uma *Workflow net* com Recursos

No contexto deste capítulo, no qual foram apresentados mecanismos de alocação de recursos em *Workflow net*, será apresentada agora uma simulação de rede de Petri que contemplem centenas de casos tratados para o processo implementado no CPN Tools apresentado na subseção 4.2. Para cada simulação serão analisados dados com relação ao atraso do processo como um todo e de cada uma das atividades, a fim de entender se existe necessidade de uma redistribuição dos recursos existentes, o que deve acontecer caso seja detectado atraso nas atividades do processo.

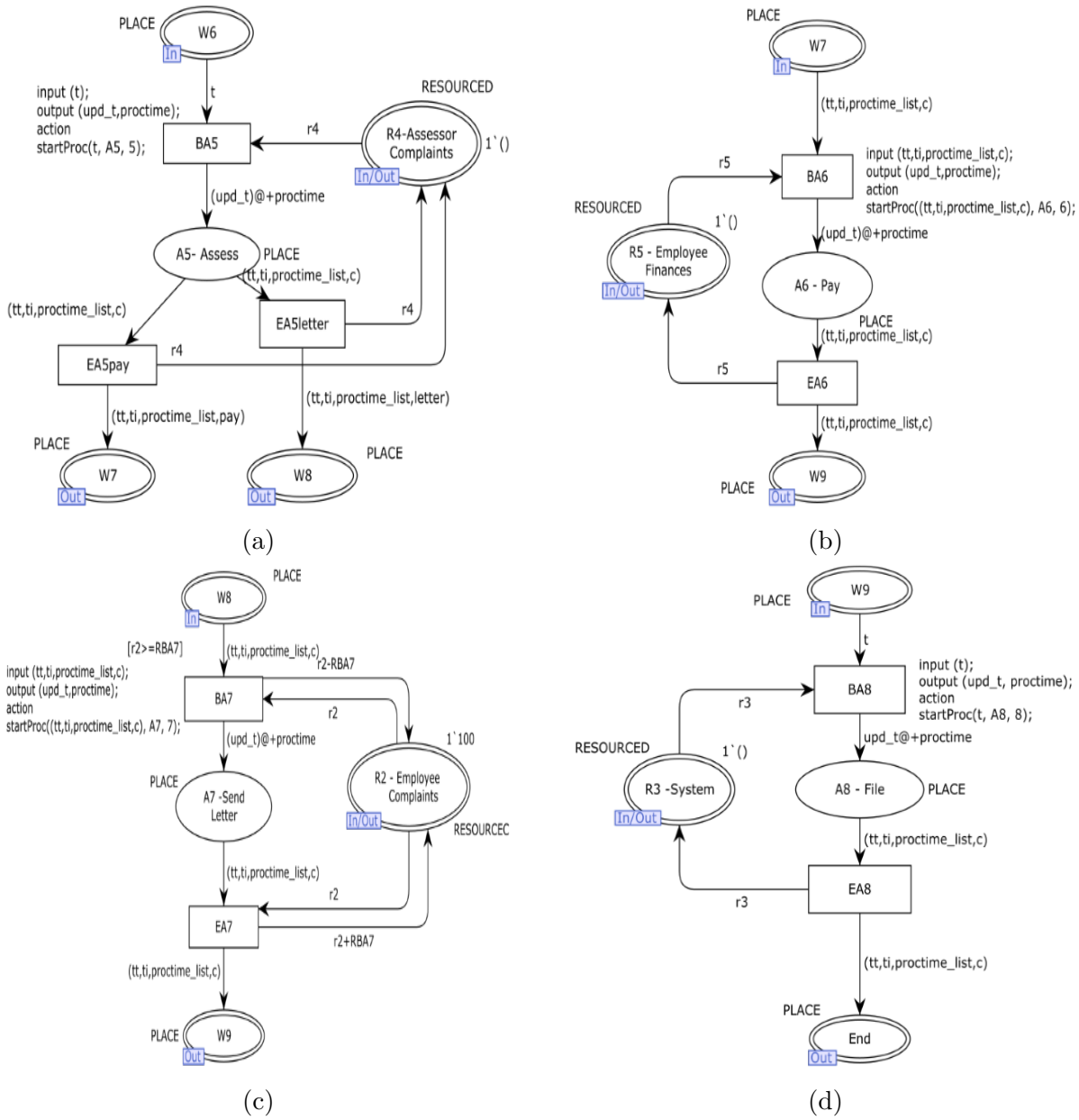


Figura 28 – Atividades A5, A6, A7 e A8 do Processo de Tratamento de Reclamações - Recursos Discretos.

Para que o modelo seja o mais completo e claro possível e visando alcançar a correta simulação do modelo proposto, foram definidas algumas funções:

a) **mda**: a cada atividade fica associada uma duração aleatória que pertence a um intervalo de tempo uniformemente distribuído. Para isto, uma função denominada *mda* foi definida para que durante a simulação a distribuição estatística seja uniforme e a probabilidade de cada instante seja a mesma.

```
1. fun mda(Z:W) = round(uniform((#1(Z)),(#2(Z))));
```

b) **intTime**: a função intTime é declarada para retornar a data de simulação corrente na forma de um valor inteiro. Esta função pertence a biblioteca de funções

predefinidas da ferramenta CPN.

```
1. fun intTime() = IntInf.toInt(time()): int;
```

c) **startProc**: a função *startProc* é responsável por atualizar o tempo de duração de cada caso. Inicialmente, a duração da atividade é calculada pela chamada da função *mda*. O tempo atual é acrescido à duração da atividade usando a função da biblioteca Base SML ModelTime.add e, então, a variável *tt* (que armazena a duração total do caso) é atualizada. Além disso, a função adiciona, em uma lista, a data em que cada atividade é disparada (com base no tempo corrente), e qual foi a escolha de ação do caso no momento que ele se depara com a rota seletiva, no qual as duas opções possíveis são *pay* e *letter*.

```
1. fun startProc ((tt,ti,proctime_list,choice):
   PLACE, Z:W, transition:TRANSITION) =
2.  let
3.     val proc_time = mda(Z)
4.     val time_stamp =
                ModelTime.add(time(),ModelTime.
                   fromInt(proc_time))
5.     val new_tt= IntInf.toInt(time_stamp)
6.     val new_proctime_list = transitionTriggerTime
   (proctime_list, new_tt, transition)
7.  in
8.     ((new_tt, ti,new_proctime_list,choice),
   proc_time)
9. end
```

d) **transitionTriggerTime**: a função *transitionTriggerTime* é responsável por adicionar à lista de tempos de processamento, o tempo no qual cada atividade foi disparada. Esse tempo pode ser obtido a partir do tempo corrente, já que esta função está ligada à função *startProc*, que é ligada ao disparo de cada transição que representa uma atividade.

```
1. fun transitionTriggerTime(proctime_list:
   PROCTIMElist, time:Tp, transition:TRANSITION) =
2.    let
3.        val new_proctime_list = proctime_list ^^ [(
   transition,time)]
4.    in
5.        new_proctime_list
6.    end
```

e) **NewT**: a função *NewT* é responsável por gerar novos casos. Cada caso tem o tempo de duração, o tempo inicial, um vetor com o tempo de processamento de cada atividade e a escolha do caminho na rota seletiva, que é inicializado como *pay* mas muda de acordo com a escolha do caso.

```
1. fun NewT() =
2.    let
3.       val time = intTime()
4.     in
5.       (time,time,[],pay)
6.    end
```

f) **expTime**: a função *expTime* é responsável por gerar as fichas iniciais a partir de uma distribuição exponencial.

```
1. fun expTime(mean:INT) =
2.    let
3.       val realMean = Real.fromInt mean
4.       val rv = exponential(1.0/realMean)
5.    in
6.       floor(rv+0.5)
7.    end
```

As fichas (ou casos) da *Workflow net* são geradas a partir de uma função de distribuição exponencial. Serão realizados 20 experimentos de simulação (replicações) e foram considerados, para cada mecanismo de alocação de recursos, um horizonte de simulação de 10000 eventos [2].

O simulador CPN Tool permite a criação de monitores para coleta de dados de simulação utilizando-se *Data Collector Monitoring Functions*, as quais apresentam um alto grau de flexibilidade, uma vez que as funções são escritas em código CPN-ML para atender necessidades específicas. Um monitor inclui uma função de predicado, que determina quando o dado deve ser coletado, e uma função de observação, que determina qual dado deve ser coletado. Sabendo que a distribuição aleatória escolhida, associada às atividades do modelo, foi a distribuição uniforme com limites mínimo e máximo, pode-se garantir que a probabilidade de cada tempo ocorrer é uniforme. Sendo assim, a duração das atividades não poderá gerar um atraso maior que o valor da borda máxima associada ao intervalo de tempo de atividade; somente a espera por um recurso poderá atrasar além do esperado (quando são considerados os valores máximos de todas as atividades envolvidas na execução do processo).

A fim de saber a duração de cada caso e de cada atividade foram definidos os seguintes moitores:

[2] valor suficiente para simular cerca de 625 casos gerados

a) **Duração Total**: a função deste monitor é calcular o tempo total de duração de cada caso. Por definição, o prazo máximo para o término do processo é de 105 unidades de tempo. Com a análise deste monitor, será possível verificar se os prazos foram cumpridos;

b) **Duração A1**: a função deste monitor é calcular o tempo de execução da atividade A1. Por definição, a duração de processamento da atividade deve estar dentro do intervalo de tempo $[5.0, 10.0]$;

c) **Duração A2**: a função deste monitor é calcular o tempo de execução da atividade A2. Por definição, a duração de processamento da atividade deve estar dentro do intervalo de tempo $[25.0, 35.0]$;

d) **Duração A3**: a função deste monitor é calcular o tempo de execução da atividade A3. Por definição, a duração de processamento da atividade deve estar dentro do intervalo de tempo $[20.0, 30.0]$;

e) **Duração A5**: a função deste monitor é calcular o tempo de execução da atividade A5. Por definição, a duração de processamento da atividade deve estar dentro do intervalo de tempo $[15.0, 25.0]$;

f) **Duração A6**: a função deste monitor é calcular o tempo de execução da atividade A6. Por definição, a duração de processamento da atividade deve estar dentro do intervalo de tempo $[5.0, 15.0]$;

g) **Quantidade de Casos**: a função deste monitor é calcular o número de casos que conclui todo o processamento, ou seja, que chegou ao lugar final da *Workflow net*;

h) **Tempo Processamento Casos**: a função deste monitor é, a partir dos dados já calculados ao longo do trajeto do caso na *Workflow net*, gerar um arquivo com as informações de tempo que iniciou e finalizou o processamento, caminho escolhido na rota seletiva e vetor com os tempos de processamento de cada atividade (transição).

Após a realização das simulações e tendo como base os dados obtidos a partir dos monitores, deseja-se analisar a ocorrência de atrasos no disparo das atividades e na conclusão do processo como um todo. Para que a análise dessas informações seja feita de forma mais rápida e precisa, foi criado o seguinte programa na linguagem Python, cujo código pode ser encontado no Apêndice A.1.

Em linhas gerais, o algoritmo lê os dados do arquivo "*Tempo_processamento_casos.txt*", gerado como resultado do monitor *Tempo Processamento Casos* e calcula os intervalos de visibilidade para cada atividade (representada na *Workflow net* por uma transição) de acordo com os resultados apresentados na Tabela 6. A partir desses dados, ele compara os intervalos de visibilidade definidos para cada atividade com o momento do disparo de cada uma delas (análise feita para todas as atividades, de todos os casos processados pela *Workflow net*). Ao final, o algoritmo apresenta os resultados referentes à porcentagem de casos que finalizam o processo com atraso (gastaram mais de 105 unidades de tempo

para finalizar o processo), a porcentagem de atividades disparadas em atraso (tempo de processamento maior que o limite máximo do intervalo de visibilidade) e, para cada atividade, quantos casos tiveram atraso nela. A informação do número de casos em atraso para cada atividade permite verificar em qual ou quais atividades existe um maior gargalo. Um detalhe importante a ser observado é que esse atraso é calculado com relação aos intervalos de visibilidade obtidos por meio da prova de sequentes da lógica linear.

4.3.1 Resultado da Simulação: Mecanismo de Alocação de Recursos Discretos

Como foi apresentado anteriormente, foram realizados 20 experimentos de simulação para obter os dados estatisticos apresentados acima. Para a análise de atraso, foram selecionadas 4 das 20 simulações e sobre elas, após a execução do programa para análise de atrasos (Apêndice A.1) foram obtidos os resultados apresentados na Figura 29. Fazendo uma média desses resultados é possível concluir que paroximadament $38,64\%$ dos casos terminam seu processo com atraso e $62,04\%$ das atividades são disparadas com atraso.

Outra informação obtida a partir da Figura 29, considerando o número de vezes que cada atividade é disparada em atraso é que o maior acúmulo de casos que não respeitam o prazo máximo, dado pelo limite máximo do intervalo de visibilidade para este caso, ocorre nas atividade $A5$, $A6$ e $A7$. Isso significa que pode estar se iniciando um gargalo de atividades no início do processo. Uma alternativa para solucionar esse problema seria realizar uma realocação de recursos, de forma que fossem alocados mais recursos para algumas atividades. No caso, escolhemos adicionar mais um recurso em $R2$, que agora passar a ter 2 fichas em sua marcação inicial. A partir dessa mudança, foram feitas 20 novos experimentos de simulação e desses, foram selecionadas 4 das 20 simulações para realizar a execução do programa para análise de atraso (Apêndice A.1) e os resultados são apresentados na Figura 30.

A partir dos resultados apresentados na Figura 30 e fazendo um comparativo com os resultados apresentados na Figura 29 é possível perceber que houve uma diminuição expressiva da porcentagem de casos que finalizam o processo com atraso, sendo agora um valor muito próximo de zero. Com relação à porcentagem de atividades que são disparadas com atraso, caiu de aproximadamente 60% no caso onde havia apenas 1 ficha em cada recurso para 30% quando o recurso $R2$ para a ter 2 fichas. Isso significa que o problema de atraso pode ser solucionado em sua grande parte pela redistribuição da quantidade de recursos na *Workflow net*.

```
====ATRASO====
De um total de 625 casos:
36.16% dos casos terminam o processo com atraso (> 105)
No geral, 61.09% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 310 casos
A3: 210 casos
A4: 406 casos
A5: 625 casos
A6/A7: 514 casos
A7: 226 casos
```

(a)

```
====ATRASO====
De um total de 625 casos:
39.36% dos casos terminam o processo com atraso (> 105)
No geral, 62.05% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 329 casos
A3: 196 casos
A4: 412 casos
A5: 623 casos
A6/A7: 521 casos
A7: 246 casos
```

(b)

```
====ATRASO====
De um total de 625 casos:
40.32% dos casos terminam o processo com atraso (> 105)
No geral, 62.75% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 311 casos
A3: 215 casos
A4: 423 casos
A5: 625 casos
A6/A7: 527 casos
A7: 252 casos
```

(c)

```
====ATRASO====
De um total de 625 casos:
38.72% dos casos terminam o processo com atraso (> 105)
No geral, 62.27% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 334 casos
A3: 191 casos
A4: 420 casos
A5: 624 casos
A6/A7: 524 casos
A7: 242 casos
```

(d)

Figura 29 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos.

4.3.2 Resultado da Simulação: Mecanismo de Alocação de Recursos Discretos e Contínuos

Os mesmos cenários de simulação realizados na seção anterior, para mecanismo de alocação de recursos discretos serão replicados agora para o mecanismo de alocação de recursos discretos e contínuos. Lembrando que foi realizado um experimento com 20 simulações, primeiramente no cenário em que todos os recursos ($R1$, $R2$, $R3$, $R4$ e $R5$) tem sua marcação inicial de valor igual a 1.

A Figura 31 mostra o resultado da aplicação do algoritmo de análise de atraso em 4 diferentes simulações. A partir dos dados obtidos é possível calcular que a porcentagem

```
====ATRASO====
De um total de 625 casos:
0.00% dos casos terminam o processo com atraso (> 105)
No geral, 30.53% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 527 casos
A3: 0 casos
A4: 5 casos
A5: 318 casos
A6/A7: 295 casos
A7: 0 casos
```

(a)

```
====ATRASO====
De um total de 625 casos:
1.12% dos casos terminam o processo com atraso (> 105)
No geral, 30.19% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 532 casos
A3: 3 casos
A4: 6 casos
A5: 308 casos
A6/A7: 276 casos
A7: 7 casos
```

(b)

```
====ATRASO====
De um total de 625 casos:
0.64% dos casos terminam o processo com atraso (> 105)
No geral, 30.24% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 518 casos
A3: 1 casos
A4: 11 casos
A5: 313 casos
A6/A7: 287 casos
A7: 4 casos
```

(c)

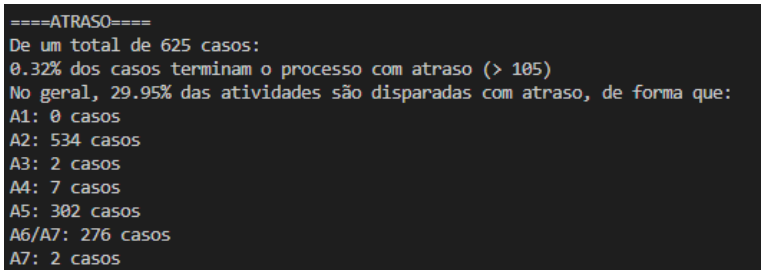
```
====ATRASO====
De um total de 624 casos:
0.32% dos casos terminam o processo com atraso (> 105)
No geral, 30.66% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 521 casos
A3: 1 casos
A4: 6 casos
A5: 316 casos
A6/A7: 302 casos
A7: 2 casos
```

(d)

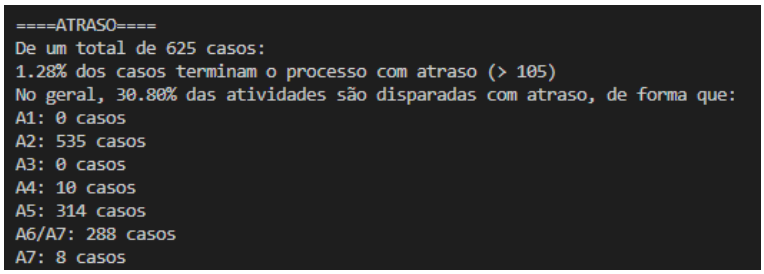
Figura 30 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com 2 recursos discretos em R2.

de casos de atraso é de $0,64\%$ (comparado com o mecanismo de alocação de recursos discretos) e a porcentagem de atraso das atividades é em média $30,58\%$, comparados com $38,64\%$ de atraso no processo e $62,04\%$ de atraso nas atividades para o mecanismo de alocação de recursos discretos nas mesmas condições. Assim, é possível perceber que o atraso é muito menor no caso contínuo.

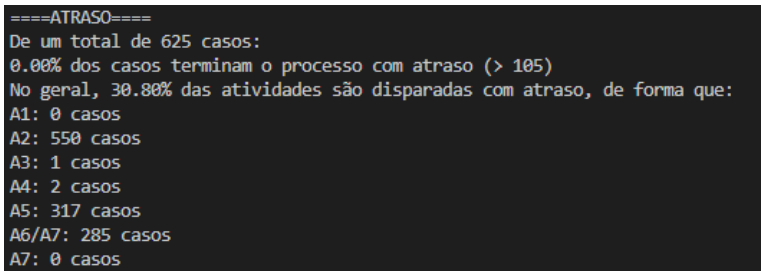
Uma nova simulação foi feita adicionando-se mais uma ficha ao recurso $R2$, de forma que agora $R1$, $R3$, $R4$ e $R5$ possuem apenas 1 recurso discreto e $R2$ possui 2 recursos contínuos. Os resultados obtidos são mostrados na Figura 32 e a partir deles é possível fazer uma comparação tanto com o caso contínuo simulado anteriormente e afirmar que



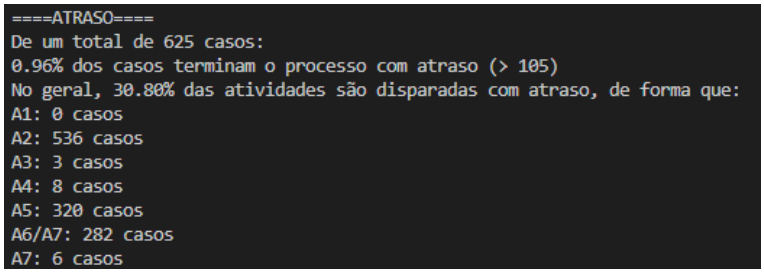
(a)



(b)



(c)



(d)

Figura 31 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e contínuos.

não houve nenhuma queda significativa nas porcentagens de atraso com a adição de um recurso a mais em $R2$.

Tabela 13 – Resultados das Simulações com Realocação de Recursos.

	Conclusão em atraso	**Atividades em atraso**
Recursos Discretos - 1 R2	38,64%	62,04%
Recursos Discretos - 2 R2	0,52%	30%
Recursos Contínuos - 1 R2	0,64%	30,58%
Recursos Contínuos - 2 R2	0,64%	30%

Comparando os resultados obtidos pelas simulações tanto com mecanismo de alocação de recursos discretos quanto pelas simulações com mecanismo de alocação discreto e

```
====ATRASO====
De um total de 625 casos:
0.00% dos casos terminam o processo com atraso (> 105)
No geral, 31.12% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 526 casos
A3: 0 casos
A4: 0 casos
A5: 337 casos
A6/A7: 304 casos
A7: 0 casos
```

(a)

```
====ATRASO====
De um total de 625 casos:
0.00% dos casos terminam o processo com atraso (> 105)
No geral, 29.89% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 517 casos
A3: 0 casos
A4: 0 casos
A5: 315 casos
A6/A7: 289 casos
A7: 0 casos
```

(b)

```
====ATRASO====
De um total de 625 casos:
0.16% dos casos terminam o processo com atraso (> 105)
No geral, 29.39% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 525 casos
A3: 0 casos
A4: 0 casos
A5: 308 casos
A6/A7: 268 casos
A7: 1 casos
```

(c)

```
====ATRASO====
De um total de 624 casos:
0.00% dos casos terminam o processo com atraso (> 105)
No geral, 30.13% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 526 casos
A3: 0 casos
A4: 0 casos
A5: 315 casos
A6/A7: 287 casos
A7: 0 casos
```

(d)

Figura 32 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com 2 recursos contínuos em R2.

contínuo (Tabela 13 é possível afirmar que no caso discreto, a adição de um novo recurso em $R2$ diminui de forma significativa as porcentagens de atraso, e as porcentagens de atraso no caso discreto com 2 recursos em $R2$ é muito semelhante às porcentagens do caso contínuo com apenas 1 recurso em $R2$.

O programa que faz a análise de atrasos, nos apresenta informações precisas sobre os atrasos nas simulações tanto de forma global quanto local (para cada atividade) e é com base nessas informações que a realocação de recursos acontece. Pelos resultados apresentados pelas Figuras 29 e 31 é possível entender em quais atividades existe maior número de casos em atraso e a partir dessa informação, entender qual recurso está ligado

a qual atividade e analisar qual recurso faz mais sentido alterar.

Capítulo 5

Análise e Experimentos para Políticas de Escalonamento

Este capítulo apresenta modelos de alocação de recursos em *Workflow net* com restrições de tempo que podem ser usados para a simulação de estratégias de escalonamento em tempo de execução (sem mecanismos de retrocesso) de SGWs. A ideia central deste capítulo está ligada ao fato de que para que se cumpram os prazos de conclusão para dos casos processados, é indispensável que a disponibilidade dos recursos utilizados pelas atividades e processos seja monitorada em função das restrições dos prazos de conclusão.

5.1 Problema de Escalonamento

O problema de escalonamento apresentado em (ESQUIROL; HUGUET; LOPEZ, 1995) consiste em organizar no tempo a realização de tarefas considerando restrições temporais (intervalos de tempo) e restrições de utilização de recursos compartilhados necessários à execução das tarefas.

Um escalonamento constitui então uma das soluções do problema e é definido pelo planejamento relacionado a execução de tarefas com restrição de tempo. No contexto dos SGWs, o escalonamento é definido pelo planejamento de execução das atividades, isto é, da ordem de execução das atividades bem como suas datas de início e fim, além da alocação dos recursos envolvidos. Uma boa solução para um problema de escalonamento de um SGWs deve considerar características importantes inerentes a cada sistema. Em particular, se muitos casos são executados simultaneamente, situações de conflito poderão ocorrer pela ultilização do mesmo recurso e deverão ser resolvidas em tempo real, ou seja, sem um mecanismo de retrocesso. Tal solução deverá em particular considerar as restrições temporais relativas aos prazos de entrega dos casos específicos, além dos diferentes roteiros que cada caso pode seguir.

Uma abordagem baseada em um jogador de rede de Petri apresentada em (SILVA; VALETTE, 1990) mostra que o disparo de transições assim que elas se tornam habilitadas

pode não ser a melhor estratégia. Dessa forma, um jogador de redes de Petri tradicional, que dispara as transições assim que elas se tornam disponíveis, dificilmente respeitará as restrições de prazos de entrega dos casos de um processo, porque o problema do escalonamento em SGWs, em vez de se resumir a um problema de otimização, trata mais de um problema de satisfação de um conjunto de restrições temporais que tem como objetivo entregar os casos dentro dos prazos determinados e não necessariamente o mais rápido possível.

Sendo assim, a proposta apresentada nesse capítulo mostra uma abordagem baseada na ideia da adição de filas em cada atividade do processo de uma *Workflow net* para que o disparo das transições ocorram de acordo com determinadas regras de escalonamento e não necessariamente assim que se tornam disponíveis.

Como a ferramenta CPN Tools permite a criação de CPN hierárquica, na qual as atividades podem ser modeladas em uma página separada da rede principal, implementando suas particularidades e ainda assim permanecer ligada à rede principal, a proposta é que seja feita uma implementação semelhante á apresentada na seção 4.2, tanto para o mecanismo de alocação de recursos discreto quanto para o mecanismo discreto e contínuo, porém com a adição de algumas particularidades no tratamento de cada atividade.

5.2 Implementação de *Workflow net* com Filas

Em (PANWALKAR; ISKANDER, 1977) são apresentados diversas regras de escalonamento que podem ser aplicadas em diferentes cenários e algumas delas serão abordadas neste trabalho. O modelo implementado será o mostrado na Figura 22, apresentado e detalhado na subseção 4.2.1 com a diferença que, em cada atividade, modelada por subprocessos, haverá uma fila responsável por receber os casos em cada atividade e determinar qual deles será executada primeiro, de acordo com determinadas regras de prioridade. O intuito dessa abordagem é tentar diminuir os atrasos tanto na execução das atividades quanto no processo como um todo. Serão aparesentados duas estruturas diferentes de modelagem de filas, uma para a fila disparada de acordo com a regra FIFO(o primeiro elemento que chega na fila é o primeiro a ser disparado) e outra para as filas disparadas de acordo com uma regra de prioridade, sendo elas: o caso que está mais distante do limite mínimo de seu intervalo de visibilidade ($rp1$), o caso que está mais próximo do limite máximo de seu intervalo de visibilidade (ou com maior atraso) ($rp2$) e o caso que está mais próximo do tempo máximo de conclusão do caso de acordo com o prazo limite para a conclusão processo ($rp3$).

As regras de prioridade foram escolhidas por serem regras de prioridade mais simples e que estão diretamente relacionadas ao atraso dos casos, que é o objeto de análise nas simulações. A primeira regra ($rp1$) dá prioriade ao caso de foi disparado primeiro, ou seja, que está na fila a mais tempo. A regra ($rp2$) dá prioridade ao caso que está mais

próximo de atingir o tempo máximo para a execução da tarefa em questão e, caso o limite máximo já tenha sido ultrapassado, a prioriadade é do caso que extrapolou mais seu limite máximo. Diferente das duas primeiras regras, que levam em consideração os limites mínimo e máximo para a conclusão de uma tarefa, a terceira regra ($rp3$) leva em consideração o tempo máximo para a conclusão do processo como um todo e dá prioridade ao caso que está mais próximo do tempo limite para conclusão do processo.

5.2.1 Implementação de *Workflow net* com Fila Utilizando Política FIFO

As subpáginas referentes a cada atividade ($A1$ até $A8$) estão representadas nas Figuras 33 e 34. As variáveis, constantes e funções desse modelo são praticamente as mesmas apresentadas na subseção 4.2.1, com adição apenas das informações apresetadas na Tabela 14, necessárias para a implementação da estrutura de filas, que é a mesma em todas as atividades.

Tabela 14 – Declarações para o Modelo CPN com Recursos Discretos e Fila FIFO.

Declarações	**Valores**
colset PLACElist	list PLACE
var L	PLACElist

É possível observar nas Figuras 33d e 34d que as atividades $A4$ e $A8$ são as únicas que não possuem a implementação de fila. Isso acontece porque ambas possuem intervalo para disparo da transição igual a zero, ou seja, a ficha chega no lugar e já é disparada, não necessitando assim de uma política de escalonamento que as controle. As atividades $A1$, $A2$, $A3$, $A5$, $A6$ e $A7$ todas possuem um fila no subprocesso de sua atividade. A fila é composta por uma transição *Create queue* e um lugar *Queue*, de forma que a ficha que chega logo dispara a transição *Create queue*, e é adicionada à lista l, que representa a fila de espera. Os elementos são retirados da fila na ordem em que entrara, por isso nenhuma função de prioridade foi implementada.

A implementação no caso de mecanismo de alocação de recurso discreto e contínuo, com relação à implementação das filas, é igual ao apresentado nas Figuras 33 e 34. O que muda é a implementação do recurso $R2$ que passa a ser contínuo como já apresentado na seção 4.2.2.

5.2.2 Implementação de *Workflow net* com Fila Utilizando Diferentes Políticas de Prioridade

As subpáginas referentes a cada atividade ($A1$ até $A8$) estão representadas nas Figuras 35 e 36. Para a implementação de filas que respeitem diferentes políticas de prioridade,

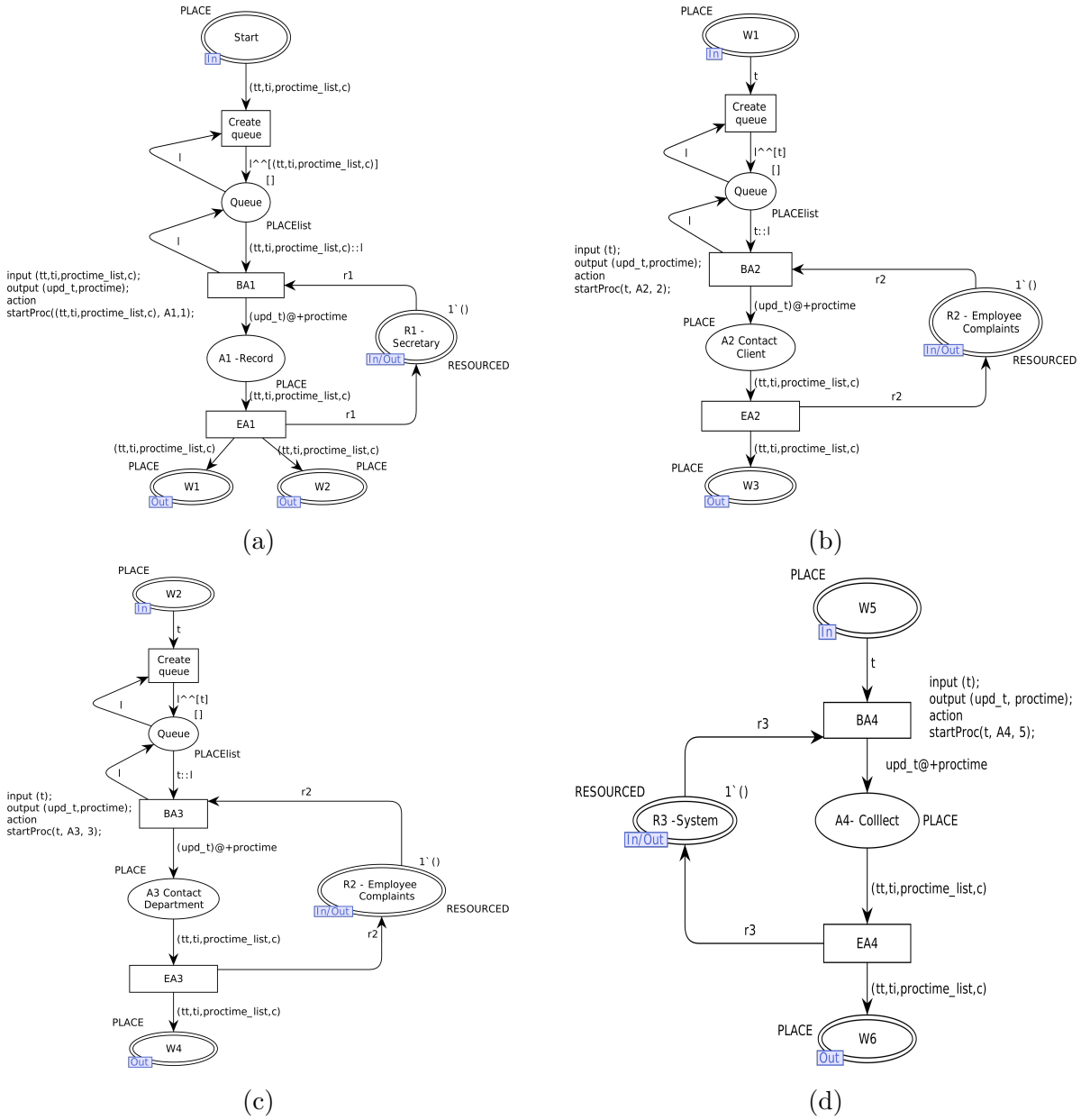


Figura 33 – Atividades A1, A2, A3 e A4 do Processo de Tratamento de Reclamações - Com Recursos Discretos e Fila FIFO.

são necessárias, além das novas declarações apresentadas na subseção 5.2.1, as seguintes novas funções:

a) **visibilityInterval**: calcula os intervalos de visibilidade relativos a cada atividade.

```
1. fun visibilityInterval(ds:INT) =
2.   let
3.       val endTime = ds + 105
4.       val dsmin = Real.fromInt ds
5.       val dp1min = dsmin + (#1(A1))
6.       val dp2min = dsmin + (#1(A1))
7.       val dp3min = dp2min + (#1(A2))
```

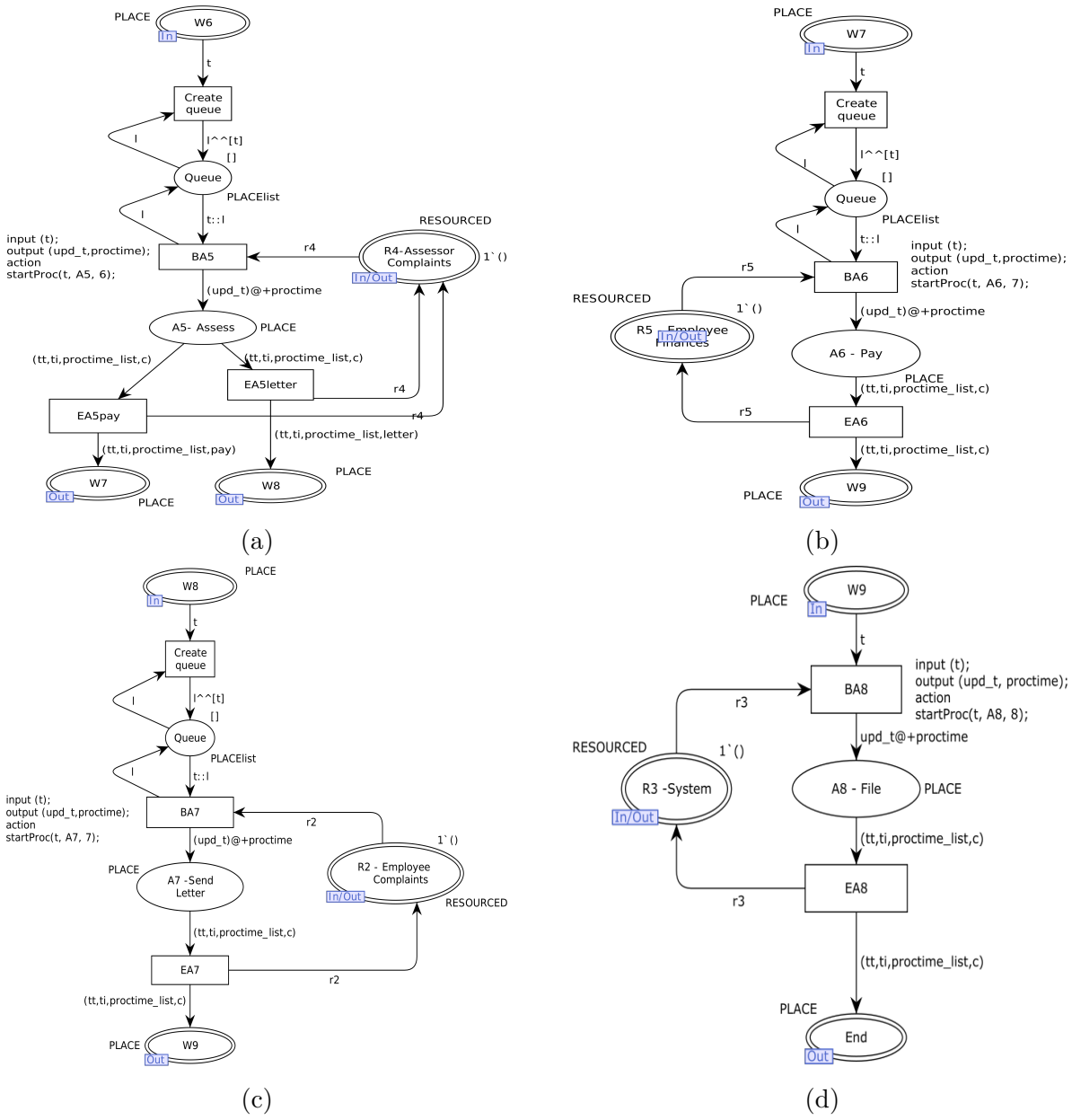


Figura 34 – Atividades A5, A6, A7 e A8 do Processo de Tratamento de Reclamações - Com Recursos Discretos e Fila FIFO.

```
8.      val dp4min = dp2min + (#1(A3))
9.      val dp5min = dp2min + Real.max((#1(A2)),(#1(
    A3)))
10.     val dp6min = dp5min + (#1(A4))
11.     val dp7min = dp6min + (#1(A5))
12.     val dp8min = dp7min + (#1(A6A7))
13.     val dendmin = dp8min + (#1(A8))
14.     val dend = Real.fromInt endTime
15.     val dendmax = dend
16.     val dp8max = dendmax  - (#2(A8))
```

```
17.     val dp7max = dp8max - (#2(A6A7))
18.     val dp6max = dp7max - (#2(A5))
19.     val dp5max = dp6max  - (#2(A4))
20.     val dp4max = dp5max
21.     val dp3max = dp5max
22.     val dp2max = dp3max - (#2(A3))
23.     val dp1max = dp3max - (#2(A2))
24.     val dsmax = dp3max -  Real.min((#2(A2)),(#2(
   A3))) - (#2(A1))
25.     val ds_interval = (dsmin, dsmax)
26.     val dp1_interval = (dp1min, dp1max)
27.     val dp2_interval = (dp2min, dp2max)
28.     val dp3_interval = (dp3min, dp3max)
29.     val dp4_interval = (dp4min, dp4max)
30.     val dp5_interval = (dp5min, dp5max)
31.     val dp6_interval = (dp6min, dp6max)
32.     val dp7_interval = (dp7min, dp7max)
33.     val dp8_interval = (dp8min, dp8max)
34.     val dend_interval = (dendmin, dendmax)
35.     val intervals_list =[ds_interval]^^[
   dp1_interval]^^[dp2_interval]^^[dp3_interval]^^[
   dp4_interval]^^[dp5_interval]^^[dp6_interval]^^[
   dp7_interval]^^[dp8_interval]^^[dend_interval]
36.   in
37.     intervals_list
38.   end
```

b) **activityDelay**: define três políticas de prioridades diferentes, $p1$, $p2$ e $p3$ de forma que $p1$ dá mais prioridade ao elemento cujo tempo está mais próximo do limite máximo do intervalo de visibilidade; $p2$ dá mais prioridade ao elemento cujo tempo está mais distante de seu limite mínimo do intervalo de visibilidade; $p3$ dá mais prioridade ao elemento cujo tempo está mais próximo do prazo final para conclusão do processo. A linha 10 define qual regra de pioridade será aplicada.

```
1. fun activityDelay((tt, ti, intervals_list,
   proctime_list,c):PLACE, transition: TRANSITION)
    =
2.   let
3.     val interval = List.nth(intervals_list,
   transition)
```

```
4.      val i_min = (#1(interval))
5.      val i_max = (#2(interval))
6.      val p1 = i_max - Real.fromInt(tt)
7.      val p2 = Real.fromInt(tt) - i_min
8.      val p3 = 105 - tt
9.    in
10.     p1
11.   end
```

c) **higherPriority**: recebe dois elementos e retorna um *booleano* que é igual a *verdadeiro* caso o valor do primeiro elemento seja menor que o do segundo e *falso* caso o segundo seja menor que o primeiro. Caso a regra de escalonamento definida pela função *activityDelay* seja $d2$, o operador da linha 6 deve ser mudado para $>$.

```
1. fun higherPriority (t1, t2, transition) =
2.  let
3.    val delay_t1 = activityDelay(t1, transition)
4.    val delay_t2 = activityDelay(t2, transition)
5.  in
6.    delay_t1 < delay_t2
7.   end
```

d) **pinsert**: insere um elemento na fila de acordo com uma ordem de prioridade definida pela função *higherPriority*.

```
1. fun pinsert elm [] transition = [elm]
2.    | pinsert elm (q::queue) transition =
3.        if higherPriority (elm,q,transition)
4.        then elm::q::queue
5.        else q::(pinsert elm queue transition);
```

Nos modelos anteriores os intervalos de visibilidade eram calculados apenas dentro do algoritmo de análise de atrasos A.1 pois essa informação não era utilizada diretamente durante a execução do modelo de simulação dentro da ferramenta CPN Tools. Porém agora, como as regras de disparo da fila são baseadas em políticas de prioridade relacionadas ao intervalo de visibilidade de cada atividade e ao prazo de conclusão do processo como um todo, o cálculo dos intervalos de visibilidade se faz necessários dentro da ferramenta. Para isso, como não sabemos se o processo, no momento que passar pela rota seletiva, vai escolher o caminho da atividade $A6$ ou da atividades $A7$, para o cálculo do intervalo de visibilidade, será utilizado um novo intervalo de habilitação que será comum para as duas transições ($valA6A7 = (10.0, 25.0) : W;$) e considera uma média entre o intervalos de habilitação de $A6$ e de $A7$.

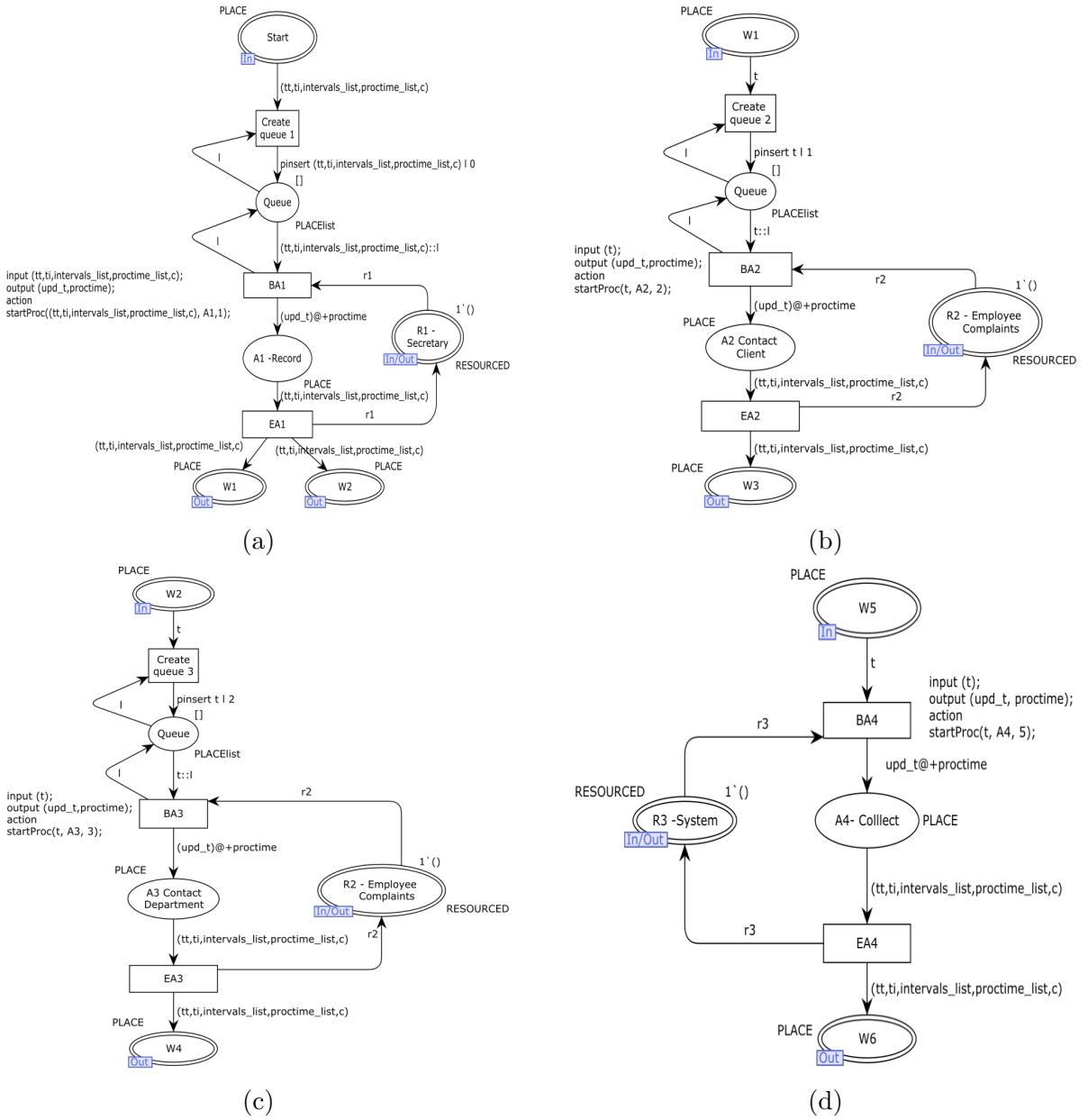


Figura 35 – Atividades A1, A2, A3 e A4 do Processo de Tratamento de Reclamações - Com Recursos Discretos e Fila Prioritária.

Além das novas funções, foi necessária também a adição de um novo monitor **Intervalos Visibilidade**, que gera um arquivo com os intervalos de visibilidade de cada atividade, para cada caso. Esse arquivo será utilizado pelo algoritmo de análise de atrasos A.2.

A implementação no caso de mecanismo de alocação de recurso discreto e contínuo, com relação à implementação das filas, é igual ao apresentado nas Figuras 35 e 36. O que muda é a implementação do recurso $R2$ que passa a ser contínuo, como já apresentado na seção 4.2.2.

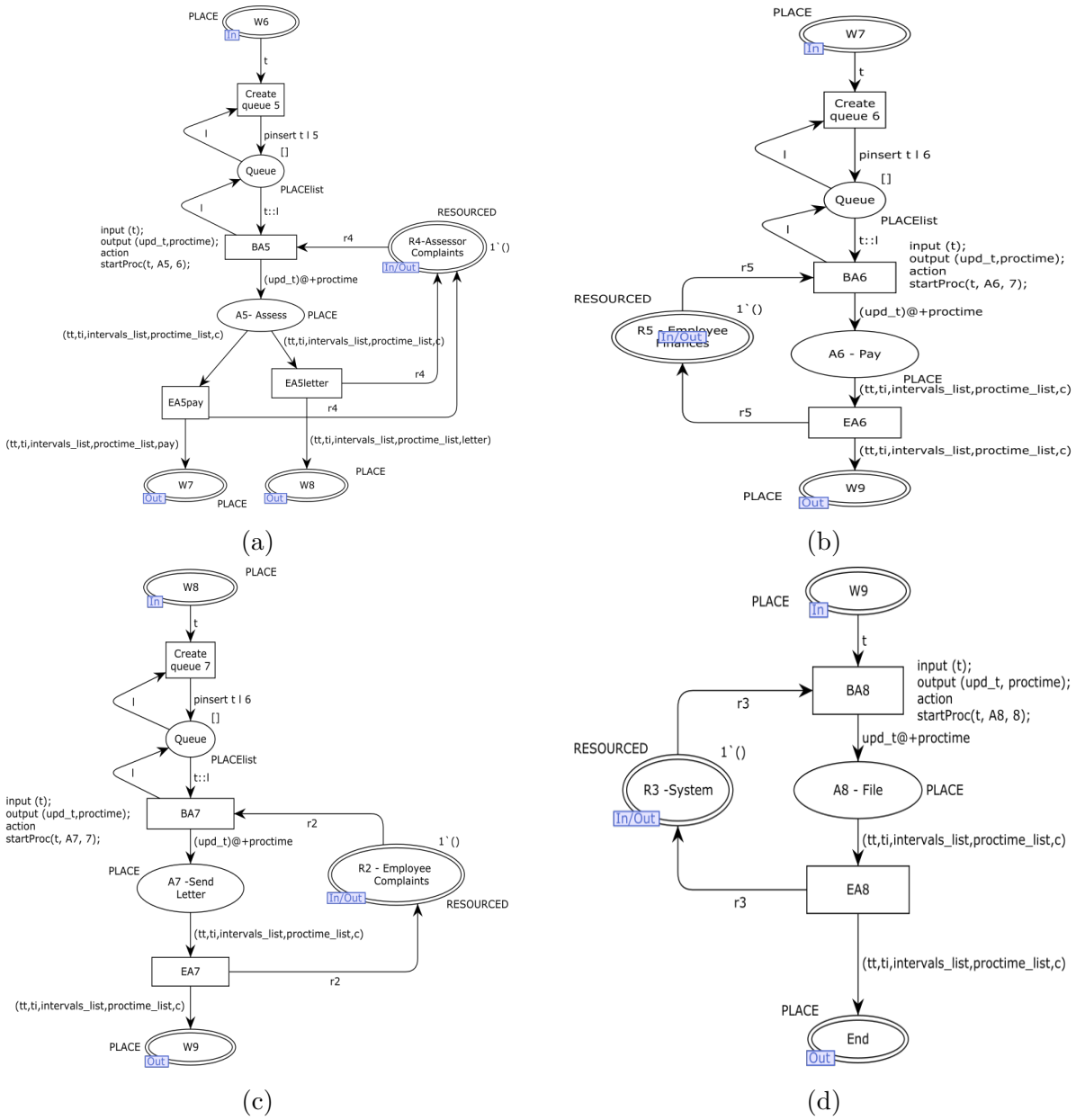


Figura 36 – Atividades A5, A6, A7 e A8 do Processo de Tratamento de Reclamações - Com Recursos Discretos e Fila Prioritária.

5.3 Resultado de Simulação de *Workflow net* com Fila

Nesta seção serão apresentados os resultados das implementações de *Workflow net* com filas que seguem diferentes políticas de escalonamento e que utilizam mecanismo de alocação de recurso discreto ou discreto e contínuo.

5.3.1 Resultado da Simulação: Fila FIFO

Nesta subseção serão apresentados os resultados obtidos a partir da realização de experimentos com 20 simulações (replicações) do modelo implementado em 5.2.1, além dos resultados obtidos para análise de atraso.

5.3.1.1 Recursos Discretos

O algoritmo para análise de atraso apresentado no Apêndice A.1 foi aplicado a 4 dos 20 resultados obtidos e os resultados estão representados na Figura 37. A partir destes resultados apresentados na Tabela 15 é possível concluir que a procentagem média de atraso no processo é de aproximadamente $37,6\%$ e a porcentagem de atraso das atividades é de aproximadamente $60,94\%$, números muito próximos aos obtidos pela simulação sem as filas ($38,64\%$ e $62,04\%$ respectivamente). Esses resultados apontam que para determinados cenários, a implementação das filas com política FIFO pode não ser tão efetiva.

Tabela 15 – Resultados das Simulações com Fila FIFO e Recursos Discretos.

	Conclusão em atraso	**Atividades em atraso**
FIFO	37,6%	60,94%
1 R2	38,64%	62,04%

5.3.1.2 Recursos Discretos e Contínuos

O resultado da análise de atraso feita com base em 4 dos 20 resultados de simulações para o modelo contínuio é apresentado na Figura 38. A partir desses resultados, apresentados na Tabela 16 é possível concluir que a média da porcentagem de atraso do processo é de aproximadamente $0,62\%$ e que a média da porcentagem de atraso das atividades é de aproximadamente $18,35\%$. Se comparados aos resultados obtidos pela simulação sem filas ($0,64\%$ e $30,58\%$ respectivamente), é possível notar uma queda significativa na porcentagem de atividades em atraso com a implementação das filas FIFO.

Tabela 16 – Resultados das Simulações com Fila FIFO e Recursos Discretos e Contínuos.

	Conclusão em atraso	**Atividades em atraso**
FIFO	0,62%%	18,35%
1 R2	0,64%	30,58%

5.3.2 Resultado da Simulação: Fila Utilizando Diferentes Políticas de Prioridade

Como na subseção anterior, foram realizadas 20 experimentos de simulação para cada regra de prioridade anteriormente definida. Dos 20, foram selecionados aleatoriamente

```
====ATRASO====
De um total de 476 casos:
36.13% dos casos terminam o processo com atraso (> 105)
No geral, 60.54% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 243 casos
A3: 147 casos
A4: 309 casos
A5: 476 casos
A6/A7: 382 casos
A7: 172 casos
```

(a)

```
====ATRASO====
De um total de 476 casos:
39.71% dos casos terminam o processo com atraso (> 105)
No geral, 61.66% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 235 casos
A3: 156 casos
A4: 306 casos
A5: 476 casos
A6/A7: 399 casos
A7: 189 casos
```

(b)

```
====ATRASO====
De um total de 476 casos:
38.24% dos casos terminam o processo com atraso (> 105)
No geral, 60.96% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 250 casos
A3: 152 casos
A4: 299 casos
A5: 476 casos
A6/A7: 382 casos
A7: 182 casos
```

(c)

```
====ATRASO====
De um total de 476 casos:
36.34% dos casos terminam o processo com atraso (> 105)
No geral, 60.61% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 247 casos
A3: 135 casos
A4: 305 casos
A5: 476 casos
A6/A7: 395 casos
A7: 173 casos
```

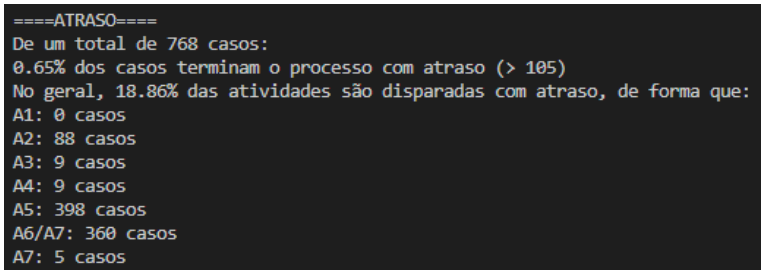
(d)

Figura 37 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e fila FIFO

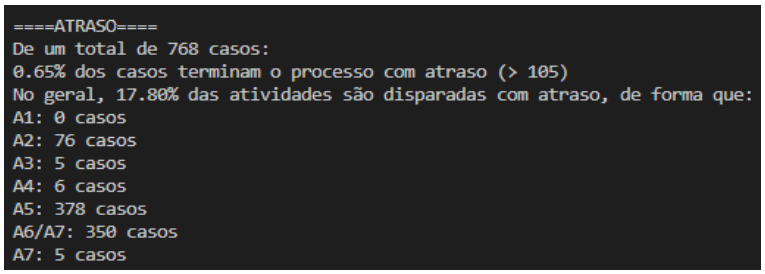
4 resultados para que seja aplicado o programa de análise de atrasos apresentado no Apêndice A.2

5.3.2.1 Recursos Discretos

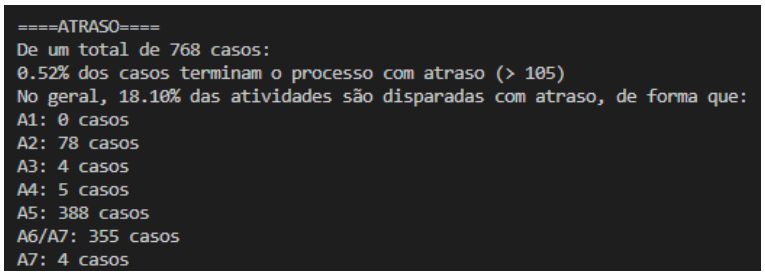
Os resultados da análise de atraso a partir das simulações para cada uma das três regras de prioridade foram apresentados nas Figuras 39, 40 e 41. É possível perceber uma semelhança na média dos atrasos tanto das atividades como do processo como um todo, que para os três processo está entre 38% e 40% para o processo como um todo e cerca de 70% para as atividades.



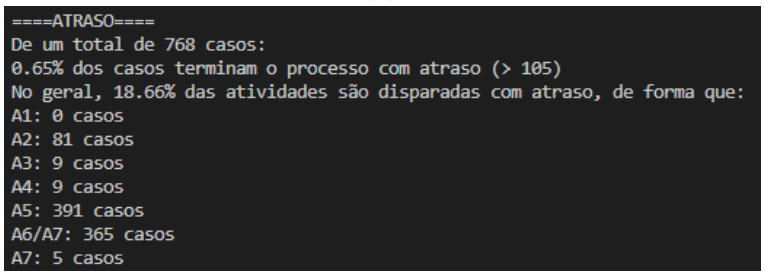
(a)



(b)



(c)



(d)

Figura 38 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e contínuos e fila FIFO

Ao compararmos as porcentagens de atraso obtidas com a implementação das filas e sem elas (Tabela 17), é possível perceber que a porcentagem de atraso nas ativiades aumentou com as filas e o atraso do processo como um todo se manteve. Como trabalho futuro pode-se investigar sobre as causas da fila não ter resolvido o problema dos atrasos e se existe alguma outra abordagem mais efetiva do que as filas para solucionar o problema de escalonamento para este caso.

O exemplo apresentado foi ilustrativo e poderá se tornar uma ferramenta de prototipação para testar *Workflow nets* com políticas de escalonamento distintase verificar qual abordagem é mais adequada para cada cenários. As políticas de escalonamento escolhidas

```
====ATRASO====
De um total de 476 casos:
35.71% dos casos terminam o processo com atraso (> 105)
No geral, 71.43% das atividades são disparadas com atraso, de forma que:
A1: 1 casos
A2: 247 casos
A3: 217 casos
A4: 451 casos
A5: 476 casos
A6/A7: 475 casos
A7: 173 casos
```

(a)

```
====ATRASO====
De um total de 476 casos:
34.03% dos casos terminam o processo com atraso (> 105)
No geral, 70.66% das atividades são disparadas com atraso, de forma que:
A1: 4 casos
A2: 232 casos
A3: 228 casos
A4: 443 casos
A5: 475 casos
A6/A7: 473 casos
A7: 163 casos
```

(b)

```
====ATRASO====
De um total de 476 casos:
38.45% dos casos terminam o processo com atraso (> 105)
No geral, 71.50% das atividades são disparadas com atraso, de forma que:
A1: 3 casos
A2: 236 casos
A3: 222 casos
A4: 452 casos
A5: 474 casos
A6/A7: 470 casos
A7: 185 casos
```

(c)

```
====ATRASO====
De um total de 476 casos:
36.13% dos casos terminam o processo com atraso (> 105)
No geral, 71.53% das atividades são disparadas com atraso, de forma que:
A1: 1 casos
A2: 205 casos
A3: 260 casos
A4: 456 casos
A5: 475 casos
A6/A7: 473 casos
A7: 173 casos
```

(d)

Figura 39 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e fila com prioridade para os casos com tempo mais próximo do limite máximo do intervalo de visibilidade(RP1).

foram algumas mais básicas e o processo ainda precisa ser testado com abordagens mais elaboradas e complexas. O objetivo primeiro deste trabalho era de produzir um modelo e a técnica de análise dos resultados. Como objetivos futuros pretende-se melhorar as políticas de escalonamento a fim de minimizar os atrasos.

5.3.2.2 Recursos Discretos e Contínuos

Os resultados da análise de atraso a partir das simulações para cada uma das três regras de prioridade foram apresentados nas Figuras 42, 43 e 44. É possível perceber uma

```
====ATRASO====
De um total de 476 casos:
37.82% dos casos terminam o processo com atraso (> 105)
No geral, 71.57% das atividades são disparadas com atraso, de forma que:
A1: 4 casos
A2: 226 casos
A3: 237 casos
A4: 449 casos
A5: 475 casos
A6/A7: 473 casos
A7: 180 casos
```

(a)

```
====ATRASO====
De um total de 476 casos:
37.39% dos casos terminam o processo com atraso (> 105)
No geral, 71.99% das atividades são disparadas com atraso, de forma que:
A1: 5 casos
A2: 243 casos
A3: 226 casos
A4: 456 casos
A5: 475 casos
A6/A7: 471 casos
A7: 180 casos
```

(b)

```
====ATRASO====
De um total de 476 casos:
41.39% dos casos terminam o processo com atraso (> 105)
No geral, 72.34% das atividades são disparadas com atraso, de forma que:
A1: 2 casos
A2: 234 casos
A3: 231 casos
A4: 454 casos
A5: 476 casos
A6/A7: 471 casos
A7: 198 casos
```

(c)

```
====ATRASO====
De um total de 476 casos:
39.50% dos casos terminam o processo com atraso (> 105)
No geral, 72.16% das atividades são disparadas com atraso, de forma que:
A1: 6 casos
A2: 222 casos
A3: 243 casos
A4: 454 casos
A5: 475 casos
A6/A7: 471 casos
A7: 190 casos
```

(d)

Figura 40 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e fila com prioridade para os casos com tempo mais distante do limite mínimo do intervalo de visibilidade (RP2).

semelhança na média dos atrasos das atividades e d do processo como um todo, que para os três processo é de cerca de $0,4\%$ para o processo como um todo, e cerca de 34% para as atividades.

Ao compararmos as porcentagens de atraso obtidas com a implementação das filas com prioridades e sem elas (Tabela 18), é possível perceber que houve uma queda na porcentagem de atraso na conclusão do processo como um todo com a aplicação das regras de prioridade $RP2$ e $RP3$, que tratam prioritariamente os casos mais proximos dos limites de conclusão tanto para a atividade quanto para o processo como um todo. Dessa forma, é possível concluir que as regras de prioridade relacionadas aos limites máximos

```
====ATRASO====
De um total de 476 casos:
37.61% dos casos terminam o processo com atraso (> 105)
No geral, 71.71% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 212 casos
A3: 251 casos
A4: 454 casos
A5: 476 casos
A6/A7: 476 casos
A7: 179 casos
```

(a)

```
====ATRASO====
De um total de 476 casos:
38.24% dos casos terminam o processo com atraso (> 105)
No geral, 71.50% das atividades são disparadas com atraso, de forma que:
A1: 0 casos
A2: 245 casos
A3: 219 casos
A4: 452 casos
A5: 476 casos
A6/A7: 468 casos
A7: 182 casos
```

(b)

```
====ATRASO====
De um total de 475 casos:
42.32% dos casos terminam o processo com atraso (> 105)
No geral, 72.04% das atividades são disparadas com atraso, de forma que:
A1: 1 casos
A2: 240 casos
A3: 217 casos
A4: 447 casos
A5: 474 casos
A6/A7: 473 casos
A7: 201 casos
```

(c)

```
====ATRASO====
De um total de 476 casos:
42.86% dos casos terminam o processo com atraso (> 105)
No geral, 71.95% das atividades são disparadas com atraso, de forma que:
A1: 2 casos
A2: 229 casos
A3: 228 casos
A4: 446 casos
A5: 474 casos
A6/A7: 472 casos
A7: 204 casos
```

(d)

Figura 41 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e fila com prioridade para os casos com tempo mais próximo do prazo limite para a conclusão do processo (RP3).

de conclusão do processo são mais efetivas do que a relacionada ao limite mínimo.

Tabela 17 – Resultados das Simulações com Filas de Prioridade e Recursos Discretos.

	Conclusão em atraso	**Atividades em atraso**
FIFO	37,6%	60,94%
1 R2	38,64%	62,04%
RP1	36%	71%
RP2	39%	72%
RP3	40%	72%

```
====ATRASO====
De um total de 476 casos:
0.21% dos casos terminam o processo com atraso (> 105)
No geral, 34.21% das atividades são disparadas com atraso, de forma que:
A1: 54 casos
A2: 422 casos
A3: 0 casos
A4: 327 casos
A5: 173 casos
A6/A7: 1 casos
A7: 0 casos
```

(a)

```
====ATRASO====
De um total de 476 casos:
0.84% dos casos terminam o processo com atraso (> 105)
No geral, 33.89% das atividades são disparadas com atraso, de forma que:
A1: 54 casos
A2: 422 casos
A3: 0 casos
A4: 334 casos
A5: 154 casos
A6/A7: 4 casos
A7: 0 casos
```

(b)

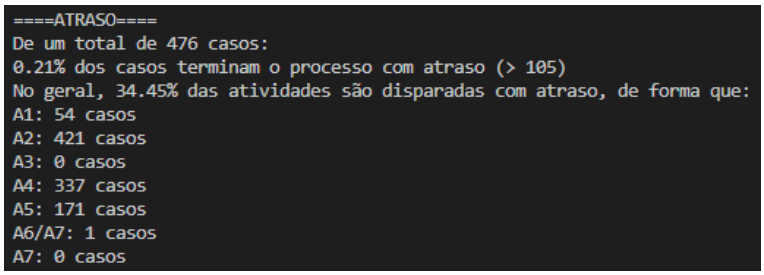
```
====ATRASO====
De um total de 476 casos:
0.63% dos casos terminam o processo com atraso (> 105)
No geral, 34.84% das atividades são disparadas com atraso, de forma que:
A1: 52 casos
A2: 424 casos
A3: 0 casos
A4: 338 casos
A5: 178 casos
A6/A7: 3 casos
A7: 0 casos
```

(c)

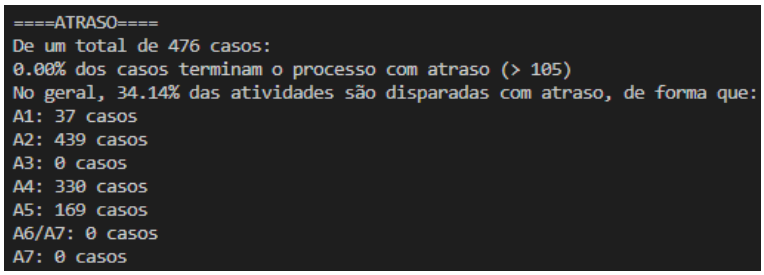
```
====ATRASO====
De um total de 476 casos:
0.63% dos casos terminam o processo com atraso (> 105)
No geral, 34.00% das atividades são disparadas com atraso, de forma que:
A1: 55 casos
A2: 421 casos
A3: 0 casos
A4: 329 casos
A5: 163 casos
A6/A7: 3 casos
A7: 0 casos
```

(d)

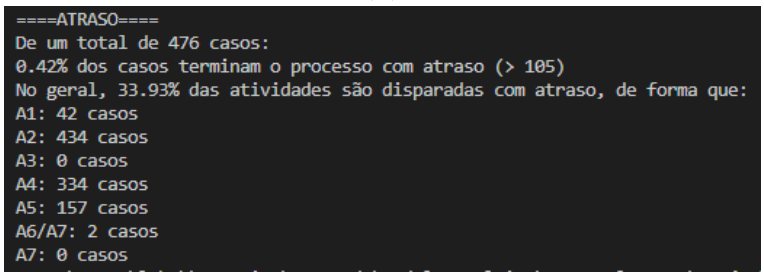
Figura 42 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e contínuos e fila com prioridade para os casos com tempo mais próximo do limite máximo do intervalo de visibilidade.



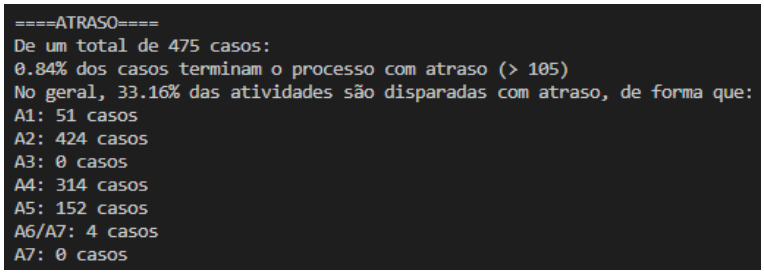
(a)



(b)



(c)

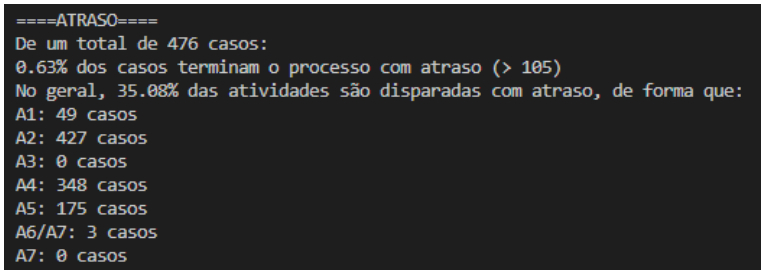


(d)

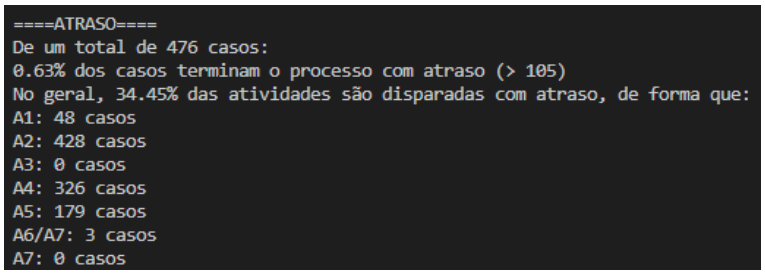
Figura 43 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e contínuos e fila com prioridade para os casos com tempo mais distante do limite mínimo do intervalo de visibilidade.

Tabela 18 – Resultados das Simulações com Filas de Prioridade e Recursos Discretos.

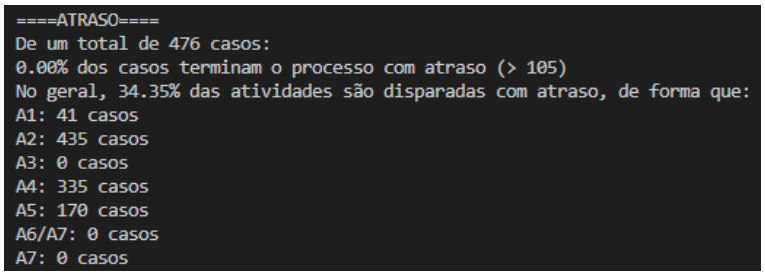
	Conclusão em atraso	**Atividades em atraso**
FIFO	0,62%%	18,35%
1 R2	0,64%	30,58%
RP1	0,57%	34%
RP2	0,36%	34%
RP3	0,31%	34%



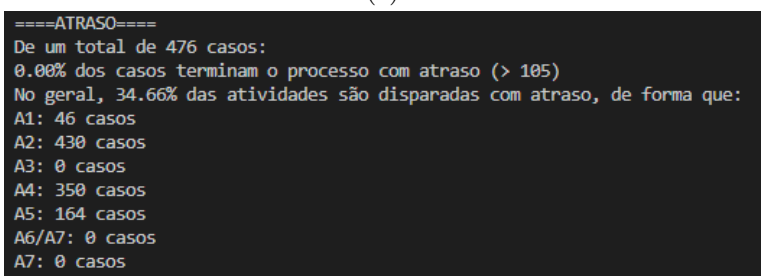
(a)



(b)



(c)



(d)

Figura 44 – Resultados relacionado ao atraso no processamento de atividades um uma *Workflow net* com recursos discretos e contínuos e fila com prioridade para os casos com tempo mais próximo do prazo limite para a conclusão do processo.

Capítulo 6

Conclusão

Este capítulo apresenta a conclusão desta pesquisa. A seção 6.1 apresenta suas principais constribuições. A seção 6.2 apresenta os trabalhos futuros que poderão ser desenvolvidos considerando os resultados obtidos nessa pesquisa. Finalmente, a seção 6.3 apresenta as contribuições em produção bibliográfica obtidas no contexto desta pesquisa.

6.1 Principais Contribuições

Este trabalho apresenta dois mecanismos de propagação com restrição de tempo, um para frente e outro para trás, a fim de calcular os intervalos de visibilidade simbólicos para cada atividade de um processo. Os intervalos são calculados com base nos cálculos de sequentes da Lógica Linear. Por serem intervalos de datas simbólicas, são genéricos e podem ser aplicados a todos os novos casos de um processo, sem a necessidade que sejam calculadas novas fórmulas.

O mecanismo de propagação para frente já havia sido apresentado em outros trabalhos, mas o mecanismo de propagação para trás com cálculo de datas simbólicas, não. O mecanismo de propagação para trás é apresentado neste trabalho a fim de calcular os limites máximos dos intervalos de visibilidade. O mecanismo de propagação para trás considera a rede invertida, ou seja, com todos os seus arcos invertidos, de forma que o lugar inicial da rede passa a ser o lugar final e vice versa. Além disso, utiliza os operadores $(min, -)$ no cálculo dos sequentes da Lógica Linear, diferente do mecanismo de propagação para frente, que utiliza os operadores $(max, +)$. Para cada caso são calculadas as datas numéricas em função da data de chegada do caso considerado e a partir delas é possível realizar uma análise dos prazos relacionados tanto aos processos como um todo quanto às atividades do processo.

Cálculos de sequentes da Lógica Linear aplicados aos roteiros das *Workflow nets* foram realizados para determinar os limites mínimos e máximos dos intervalos de visibilidade, que representam de forma simbólica as datas de início ao mais cedo e ao mais tarde das atividades a serem executadas e fornece informações essenciais para estabelecer heurísticas

de escalonamento que permitem o respeito dos prazos dos casos a serem tratados nos processos de negócios.

Um processo de tratamento de reclamações foi modelado na ferramenta CPN Tools utilizado redes de Petri Coloridas Hierárquicas. A partir do modelo do processo *Workflow net* foram definidos mecanismos de alocação de recursos, o que torna o modelo mais próximo à realidade de gestão de recursos dos sistemas de gerenciamento de processos de negócio e a realocação de recursos é um fator importante na melhoria do cumprimento dos prazos de conclusão estabelecidos para um determinado processo.

Além disso, o processo modelado também apresenta restrições de tempo determinadas pelos intervalos de visibilidade e possíveis políticas de escalonamento para a utilização dos recursos considerando os intervalos de visibilidade planejados de cada atividade. Para cada cenário distinto, foram realizadas diversas simulações na rede a fim de obter dados para a análise de atrasos. Os intervalos de visibilidade podem ser calculados dentro da própria ferramenta CPN Tools a cada introdução de novos casos a serem processados.

Programas implementados em linguagem Pyhton foram utilizados para análise dos dados das simulações; em particular para estimar a proporção de casos tratados sem e com atraso em relação ao plano inicial que depende dos intervalos de visibilidade permitidos para cada atividade do processo. Essas informações são importantes para que seja definido um possível redimensionamento (ou redistribuição) dos recursos envolvidos nas atividades do processo na tantativa de solucionar os problemas de atrasos.

Também foram implementadas diversas políticas de escalonamento dos recursos quando o dimensionamento dos recursos não pode mais ser redefinido. Foram considerados tanto recursos discretos (fichas) quanto recursos contínuos (porcentagem) para deixar o modelo o mais realista possível.

6.2 Trabalhos Futuros

Como trabalhos futuros pretende-se entender melhor o porquê de as políticas de escalonamento implementadas não terem resultados tão eficientes, e isso pode ser investigado por meio de monitores, que são mecanismos presentes no próprio CPN Tools para observar, inspecionar, controlar ou modificar uma simulação de uma rede de Petri.

Em particular, já que as políticas de escalonamento escolhidas foram baseadas em heurísticas bem tradicionais, pretende-se implementar políticas de escalonamento inteligentes aplicadas a casos mais específicos (casos com graus de prioridade diferentes, por exemplo).

Estudar outras formas de calcular as datas simbólicas dos limites mínimo e máximo dos intervalos de visibilidade e de analisar os resultados obtidos com as simulações envolvendo Inteligência Artificial.

Poderão finalmente ser criadas novas técnicas de análise de atrasos que poderão explorar melhor os resultados fornecidos pelas simulações, em particular para extrair dados necessários a elaboração de políticas de escalonamento inteligentes baseadas em aprendizage, por exemplo.

6.3 Contribuições em Produção Bibliográfica

O artigo (BRUNO; JULIA, 2022) foi publicado e apresentado na conferência *ICEIS (International Conference on Enterprise Information Systems)* no ano de 2022 e apresentou os mecanismos de propagação com restrição de tempo mostrado na capítulo 4 baseado na Lógica Linear para calcular fórmulas simbólicas para expressar os intervalos de visibilidade nos quais cada ficha (que representa um caso) deve iniciar e finalizar cada atividade.

Além disso, um outro artigo com o título "Resource Planning in Workflow Nets Based on a Symbolic Time Constraint Propagation Mechanism" foi submetido para ser publicado em *LNBIP (Lecture Notes in Business Information Processing) Series* publicado pela *Springer*. Este artigo considera os resultados obtidos em (BRUNO; JULIA, 2022) para modelar, simular e monitorar o processo de *workflow* utilizando rede de Petri Colorida Hierárquica na ferramenta CPN Tools.

Referências

AALST, W. M. P. **Timed coloured Petri nets and their application to logistics**. Tese (Doutorado), 1992.

AALST, W. M. Van der. The application of petri nets to workflow management. **Journal of circuits, systems, and computers**, World Scientific, v. 8, n. 01, p. 21–66, 1998. Disponível em: <https://doi.org/10.1142/s0218126698000043>.

AALST, W. M. Van der et al. Verification of workflow nets. In: **ICATPN**. [s.n.], 1997. v. 97, n. 708, p. 407–426. Disponível em: <https://doi.org/10.1007/3-540-63139-9_48>.

AALST, W. V. D.; HEE, K. M. V.; HEE, K. van. **Workflow management: models, methods, and systems**. [S.l.]: MIT press, 2004.

BERARD, B. et al. The expressive power of time petri nets. **Theoretical Computer Science**, Elsevier, v. 474, p. 1–20, 2013. Disponível em: <https://doi.org/10.1016/j.tcs.2012.12.005>.

BERTHOMIEU, B.; MENASCHE, M. An enumerative approach for analyzing time petri nets. In: PARIS. **IFIP congress**. [S.l.], 1983. v. 41, p. 46.

BOWDEN, F. D. A brief survey and synthesis of the roles of time in petri nets. **Mathematical and Computer Modelling**, Elsevier, v. 31, n. 10-12, p. 55–68, 2000. Disponível em: <https://doi.org/10.1016/S0895-7177(00)00072-8>.

BRUNO, L. R.; JULIA, S. A symbolic time constraint propagation mechanism proposal for workflow nets. In: **ICEIS (1)**. [S.l.: s.n.], 2022. p. 537–544.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Florianópolis: Editora da UFSC, 1997.

CELKO, J. Time token design methodology. **Software: Practice and Experience**, Wiley Online Library, v. 12, n. 10, p. 889–895, 1982. Disponível em: <https://doi.org/10.1002/spe.4380121003>.

CERONE, A.; MAGGIOLO-SCHETTINI, A. Time-based expressivity of time petri nets for system specification. **Theoretical Computer Science**, Elsevier, v. 216, n. 1-2, p. 1–53, 1999. Disponível em: <https://doi.org/10.1016/S0304-3975(98)00008-5>.

CHAMPAGNAT, R.; PRADIN-CHÉZALVIEL, B.; VALETTE, R. Petri nets and linear logic as an aid for scheduling batch processes. **ADPM 2000, Automation of mixed processes: Hybrid Dynamic Systems**, p. 107–112, 2000.

DAVID, R.; ALLA, H. **Discrete, continuous, and hybrid Petri nets**. Springer, 2010. v. 1. Disponível em: <https://doi.org/10.1016/S0304-3975(98)00008-5>.

ESQUIROL, P.; HUGUET, M.-J.; LOPEZ, P. Modelling and managing disjunctions in scheduling problems. **Journal of Intelligent Manufacturing**, Springer, v. 6, n. 2, p. 133–144, 1995. Disponível em: <https://doi.org/10.1007/BF00123685>.

FREITAS, J. C. J. de; JULIA, S. Fuzzy time constraint propagation mechanism for workflow nets. In: IEEE. **2015 12th International Conference on Information Technology-New Generations**. [S.l.], 2015. p. 367–372.

GIRARD, J.-Y. Linear logic. **Theoretical computer science**, Elsevier, v. 50, n. 1, p. 1–101, 1987. Disponível em: <https://doi.org/10.1017/CBO9780511629150.002>.

_____. Linear logic: its syntax and semantics. **London Mathematical Society Lecture Note Series**, Cambridge University Press, p. 1–42, 1995.

GIRAULT, F.; PRADIER-CHEZALVIEL, B.; VALETTE, R. A logic for petri nets. **Journal européen des systèmes automatisés**, v. 31, n. 3, p. 525–542, 1997.

GORGÔNIO, K. C. et al. Adaptação de modelos em redes de petri coloridas. Universidade Federal de Campina Grande, 2001.

HEE, K. van; SIDOROVA, N.; VOORHOEVE, M. Resource-constrained workflow nets. **Fundamenta Informaticae**, IOS Press, v. 71, n. 2-3, p. 243–257, 2006.

JENSEN, K. Coloured petri nets and the invariant-method. **Theoretical computer science**, Elsevier, v. 14, n. 3, p. 317–336, 1981. Disponível em: <https://doi.org/10.1016/0304-3975(81)90049-9>.

JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri nets: modelling and validation of concurrent systems**. Springer Science & Business Media, 2009. Disponível em: <https://doi.org/10.1007/b95112>.

JENSEN, K.; KRISTENSEN, L. M.; WELLS, L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. **International Journal on Software Tools for Technology Transfer**, Springer, v. 9, n. 3, p. 213–254, 2007. Disponível em: <https://doi.org/10.1007/s10009-007-0038-x>.

JESKE, J. C.; JULIA, S.; VALETTE, R. Fuzzy continuous resource allocation mechanisms in workflow management systems. In: IEEE. **2009 XXIII Brazilian Symposium on Software Engineering**. 2009. p. 236–251. Disponível em: <https://doi.org/10.1109/SBES.2009.17>.

JESKE, J. C. et al. Mecanismo de alocação de recurso fuzzy para sistemas de gerenciamento de workflow. Universidade Federal de Uberlândia, 2006.

JULIA, S.; OLIVEIRA, F. F. de; VALETTE, R. Real time scheduling of workflow management systems based on a p-time petri net model with hybrid resources. **Simulation Modelling Practice and Theory**, Elsevier, v. 16, n. 4, p. 462–482, 2008. Disponível em: <https://doi.org/10.1016/j.simpat.2008.01.006>.

JULIA, S.; SOARES, M. Verification of real time uml specifications through a specialized inference mechanism based on a token player algorithm and the sequent calculus of linear logic. In: **Proceedings of the 15th European Simulation Symposium and Exhibition (EMSS'03)**. [S.l.: s.n.], 2003. p. 65–70.

KHALFHOUI, S. et al. An algorithm for deriving critical scenarios in mechatronic systems. In: IEEE. **IEEE International Conference on Systems, Man and Cybernetics**. [S.l.], 2002. v. 3, p. 6–pp.

KOTB, Y. T.; BADREDDIN, E. Synchronization among activities in a workflow using extended workflow petri nets. In: IEEE. **Seventh IEEE International Conference on E-Commerce Technology (CEC'05)**. [S.l.], 2005. p. 548–551.

LIU, G. Complexity of the deadlock problem for petri nets modeling resource allocation systems. **Information Sciences**, Elsevier, v. 363, p. 190–197, 2016. Disponível em: <https://doi.org/10.1016/j.ins.2015.11.025>.

MARTOS-SALGADO, M.; ROSA-VELARDO, F. Dynamic soundness in resource-constrained workflow nets. In: SPRINGER. **Formal Techniques for Distributed Systems: Joint 13th IFIP WG 6.1 International Conference, FMOODS 2011, and 30th IFIP WG 6.1 International Conference, FORTE 2011, Reykjavik, Iceland, June 6-9, 2011. Proceedings**. [S.l.], 2011. p. 259–273.

MEDEIROS, F. F.; JULIA, S. Constraint analysis based on energetic reasoning applied to the problem of real time scheduling of workflow management systems. In: **ICEIS (3)**. [s.n.], 2017. p. 373–380. Disponível em: <https://doi.org/10.5220/0006275903730380>.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, IEEE, v. 77, n. 4, p. 541–580, 1989. Disponível em: <https://doi.org/10.1109/5.24143>.

OLIVEIRA, K. S.; JULIA, S. Detection and removal of negative requirements of deadlock-type in service-oriented architectures. In: **2020 International Conference on Computational Science and Computational Intelligence**. [S.l.: s.n.], 2020.

PANWALKAR, S. S.; ISKANDER, W. A survey of scheduling rules. **Operations research**, INFORMS, v. 25, n. 1, p. 45–61, 1977. Disponível em: <https://doi.org/10.1287/opre.25.1.45>.

PASSOS, L. M. S.; JULIA, S. Qualitative analysis of workflow nets using linear logic: Soundness verification. In: IEEE. **2009 IEEE International Conference on Systems, Man and Cybernetics**. 2009. p. 2843–2847. Disponível em: <https://doi.org/10.1109/ICSMC.2009.5346601>.

PASSOS, L. M. S.; JULIA, S. Linear logic as a tool for qualitative and quantitative analysis of workow processes. **International Journal on Artificial Intelligence Tools**, World Scientific, v. 25, n. 03, p. 1650008, 2016. Disponível em: <https://doi.org/10.1142/S0218213016500081>.

PETRI, C. **Kommunikation mit Automaten. Schriften des IIM Nr. 3, Institut fur Instrumentelle Mathematik, Bonn, West Germany. See, also, Communication with Automata (in English). Gri ss Air Force Base**. [S.l.], 1962.

PRADIN-CHÉZALVIEL, B. et al. Calculating duration of concurrent scenarios in time petri nets. **APII-Journal Européen des Systèmes Automatisés**, v. 33, n. 8-9, p. 943–958, 1999.

PRADIN-CHÉZALVIEL, B.; VALETTE, R.; KUNZLE, L. A. Scenario durations characterization of t-timed petri nets using linear logic. In: IEEE. **Proceedings 8th International Workshop on Petri Nets and Performance Models (Cat. No. PR00331)**. [S.l.], 1999. p. 208–217.

RATZER, A. V. et al. Cpn tools for editing, simulating, and analysing coloured petri nets. In: SPRINGER. **International Conference on Application and Theory of Petri Nets**. 2003. p. 450–462. Disponível em: <https://doi.org/10.1007/3-540-44919-1_28>.

REZENDE, L. P. d. et al. Workflow net possibilística para problemas de não conformidade em processos de negócios. Universidade Federal de Uberlândia, 2013.

RIVIERE, N.; PRADIN-CHEZALVIEL, B.; VALETTE, R. Reachability and temporal conflicts in t-time petri nets. In: IEEE. **Proceedings 9th International Workshop on Petri Nets and Performance Models**. [S.l.], 2001. p. 229–238.

SILVA, M.; VALETTE, R. Petri nets and flexible manufacturing. In: SPRINGER. **European Workshop on Applications and Theory in Petri Nets**. 1990. p. 374–417. Disponível em: <https://doi.org/10.1007/3-540-52494-0_38>.

SOARES, M. d. S. **Uma abordagem baseada num jogador de redes de Petri p-temporal e no cálculo de sequentes da Lógica Linear para a verificação de cenários de Sistemas Tempo Real especificados através de diagramas dinâmicos da UML**. Tese (Doutorado) — Master's thesis, Faculdade de Computação, Universidade Federal de Uberlândia, 2004.

SOARES, M. dos S.; JULIA, S.; VRANCKEN, J. Real-time scheduling of batch systems using petri nets and linear logic. **Journal of Systems and Software**, Elsevier, v. 81, n. 11, p. 1983–1996, 2008. Disponível em: <https://doi.org/10.1016/j.jss.2008.01.018>.

TSAI, J. J. P.; YANG, S. J.; CHANG, Y.-H. Timing constraint petri nets and their application to schedulability analysis of real-time system specifications. **IEEE transactions on Software Engineering**, IEEE, v. 21, n. 1, p. 32–49, 1995. Disponível em: <https://doi.org/10.1109/32.341845>.

VALETTE, R. Analysis of petri nets by stepwise refinements. **Journal of computer and system sciences**, Elsevier, v. 18, n. 1, p. 35–46, 1979. Disponível em: <https://doi.org/10.1016/0022-0000(79)90050-3>.

WANG, J.; LI, D. Resource oriented workflow nets and workflow resource requirement analysis. **International Journal of Software Engineering and Knowledge Engineering**, World Scientific, v. 23, n. 05, p. 677–693, 2013. Disponível em: <https://doi.org/10.1142/S0218194013400135>.

Apêndices

APÊNDICE **A**

Programas Para Análise dos Dados Gerados Pela Simulação no CPN Tools

Neste Apêndice, serão apresentados os programs criados na linguagem Python para a análise dos dados produzidos pelas simulações no CPN Tools.

A.1 Programa com cálculo de intervalos de visibilidade

```
#Tempo_processamento_casos.txt
process_time_file = open(r"C:\...\output\
    Tempo_processamento_casos.txt","r")

lines_list_process_time = process_time_file.readlines()
my_data_times = [[(val) for val in line.strip("][").split()] for
     line in lines_list_process_time[0:]]

#sera' considerado apenas o numero de linhas do arquivo "
    Tempo_processamento_casos.txt" pois e' o numero real de casos
     que atingiram o final do processamento
lines = len(lines_list_process_time)
columns_process_time = len(lines_list_process_time[0])


#calcula intervalo de visibilidade para a6 e a7
a1 = (5.0, 10.0)
a2 = (20.0, 30.0)
a3 = (25.0, 35.0)
transition = (0.0, 0.0)
a4 = (0.0, 0.0)
```

```
a5 = (15.0, 25.0)
a6 = (5.0, 15.0)
a7 = (20.0, 30.0)
a8 = (0.0, 0.0)

visibility_interval = []
visibility_interval_list = []

def calculate_visibility_intervals(ds,activity):
    dsmin = ds
    endTime = dsmin + 105
    dp1min = dsmin + a1[0]
    dp2min = dsmin + a1[0]
    dp3min = dp2min + a2[0]
    dp4min = dp2min + a3[0]
    dp5min = dp2min + max(a2[0], a3[0]) + transition[0]
    dp6min = dp5min + a4[0]
    dp7min = dp6min + a5[0]
    dp8min = dp7min + activity[0]
    dendmin = dp8min + a8[0]
    dendmax = endTime
    dp8max = dendmax - a8[1]
    dp7max = dp8max - activity[1]
    dp6max = dp7max - a5[1]
    dp5max = dp6max - a4[1]
    dp4max = dp5max - transition[1]
    dp3max = dp5max - transition[1]
    dp2max = dp3max - a3[1]
    dp1max = dp3max - a2[1]
    dsmax = dp3max - min(a2[1],a3[1]) - a1[1]
    visibility_interval.append((dsmin, dsmax))
    visibility_interval.append((dp1min, dp1max))
    visibility_interval.append((dp2min, dp2max))
    visibility_interval.append((dp3min, dp3max))
    visibility_interval.append((dp4min, dp4max))
    visibility_interval.append((dp5min, dp5max))
    visibility_interval.append((dp6min, dp6max))
    visibility_interval.append((dp7min, dp7max))
    visibility_interval.append((dp8min, dp8max))
```

```
        visibility_interval.append((dendmin, dendmax))

#GERAL
init_time = 0
count_a6 = 0
count_a7 = 0
count = 0
activities_count = 0
simulation_time_list = []
bottleneck = [0,0,0,0,0,0,0,0,0]
process_time_list = []
activity_chooses = []
aux = list(tuple())
aux_tuple = tuple()

#para cada caso, pega a informa  o de para qual caminho ele
    seguiu, calcula seus intervalos de visibilidade e analisa se
    houve atraso no processo como um todo
for data in my_data_times:
    simulation_time = int(data[0]) - int(data[1])
    init_time = int(data[1])
    activity_choose = data[2]
    if activity_choose == "pay":
        count_a6 += 1
        calculate_visibility_intervals(init_time,a6)
    elif activity_choose == "letter":
        count_a7 += 1
        calculate_visibility_intervals(init_time,a7)
    simulation_time_list.append(simulation_time)
    if simulation_time > 105:
        count+=1
    visibility_interval_list.append(visibility_interval)
    visibility_interval = []

#transforma o arquivo "Tempo_processamento_casos.txt" em um
    array de arrays com os tempos de processamento de cada caso
    em sua respectiva posi  o
for data in my_data_times:
    process_times = data[3].strip("[]()").split(",")
```

```
    it = iter(process_times)
    for item in it:
        y = list(aux_tuple)
        y.append(int(item.strip("()")))
        y.append(int(next(it).strip("()")))
        aux_tuple = tuple(y)
        aux.append(aux_tuple)
        aux_tuple = tuple()
    process_time_list.append(aux)
    aux = []

#para cada caso, a partir do tempo de processamento e do
   intervalo de visibilidade para cada atividade, analise se
   houve atraso
for line in range(lines):
    for item in range(len(process_time_list[line])):
        item_index = process_time_list[line][item][0]
        item_process_time = process_time_list[line][item][1]
        item_visibility_interval = visibility_interval_list[line
           ][item_index]
        activities_count += 1

        if item_process_time > item_visibility_interval[1]:
            bottleneck[item_index] += 1

bottleneck.pop(0)
bottleneck.pop(3)

#caminho escolhido por cada caso: A6 ou A7
activity_pay_percentage = (count_a6/lines)*100
activity_letter_percentage = (count_a7/lines)*100
print("====CAMINHO ESCOLHIDO====")
print(f'{activity_pay_percentage:,.2f}% dos casos foram pelo
   caminho PAY')
print(f'{activity_letter_percentage:,.2f}% dos casos foram pelo
   caminho SEND LETTER\n')

#porcentagem de casos em atraso
print("====ATRASO====")
```

```
print('De um total de ' + str(lines) + ' casos:')
process_delay_percentage = ((count/lines)*100)
print(f'{process_delay_percentage:,.2f}% dos casos terminam o
    processo com atraso (> 105)')

#para os casos de atraso, onde est o os atrasos
activity_delay_percentage = ((sum(bottleneck)/activities_count)
    *100)
print(f'No geral, {activity_delay_percentage:,.2f}% das
    atividades s o disparadas com atraso, de forma que:')
for i in range(len(bottleneck)):
    index_activity = str(i+1) if i != 5 else '6/A7'
    print('A' + index_activity + ': ' + str(bottleneck[i]) + '
        casos')
```

A.2 Programa com entrada do arquivo com os intervalos de visibilidade

```
# Tempo_final_casos
process_time_file = open(r"C:\...\output\
    Tempo_processamento_casos.txt","r")

lines_list_process_time = process_time_file.readlines()
my_data_times = [[(val) for val in line.strip("][").split()] for
    line in lines_list_process_time[0:]]

#ser  considerado apenas o n mero de linhas do arquivo "
    Tempo_final_casos.txt" pois    o n mero real de casos que
    atingiram o final do processamento
lines = len(lines_list_process_time)
columns_process_time = len(lines_list_process_time[0])

# Intervalos_visibilidade
visibility_intevals_file = open(r"C:\...\output\
    Intervalos_visibilidade.txt","r")

lines_list_visibility_intevals = visibility_intevals_file.
    readlines()
my_data_visibility_intervals = [[(val) for val in line.strip("][
```

```
    ").split()] for line in lines_list_visibility_intevals[0:]]

#GERAL
count = 0
count_a6 = 0
count_a7 = 0
activities_count= 0
simulation_time_list = []
bottleneck = [0,0,0,0,0,0,0,0,0]
process_time_list = []
visibility_inteval_list = []
aux = list(tuple())
aux_tuple = tuple()

#para cada caso, pega a informa  o de para qual caminho ele
    seguiu e analisa se houve atraso no processo como um todo
for data in my_data_times:
    simulation_time = int(data[0]) - int(data[1])
    init_time = int(data[1])
    activity_choose = data[2]
    if activity_choose == "pay":
        count_a6 += 1
    elif activity_choose == "letter":
        count_a7 += 1
    simulation_time_list.append(simulation_time)
    if simulation_time > 105:
        count+=1

#transforma o arquivo "Tempo_processamento_casos.txt" em um
    array de arrays com os tempos de processamento de cada caso
    em sua respectiva posi  o
for data in my_data_times:
    process_times = data[3].strip("[]()").split(",")
    it = iter(process_times)
    for item in it:
        y = list(aux_tuple)
        y.append(int(item.strip("()")))
        y.append(int(next(it).strip("()")))
        aux_tuple = tuple(y)
```

```
            aux.append(aux_tuple)
            aux_tuple = tuple()
        process_time_list.append(aux)
        aux = []

#transforma o arquivo "Intervalos_visibilidade.txt" em um array
    com os intervalos de visibilidade para cada atividade
#esses intervalos foram calculados por meio de uma m dia dos
    tempos de processamento de A6 e A7
for data in my_data_visibility_intervals:
    visibility_intervals = data[1].strip("[]()").split(",")
    it = iter(visibility_intervals)
    for item in it:
        y = list(aux_tuple)
        y.append(float(item.strip("()")))
        y.append(float(next(it).strip("()")))
        aux_tuple = tuple(y)
        aux.append(aux_tuple)
        aux_tuple = tuple()
    visibility_inteval_list.append(aux)
    aux = []

for line in range(lines):
    for item in range(len(process_time_list[line])):
        item_index = process_time_list[line][item][0]
        item_process_time = process_time_list[line][item][1]
        visibility_interval = visibility_inteval_list[line][
            item_index]
        activities_count += 1

        if item_process_time > visibility_interval[1]:
            bottleneck[item_index] += 1
            if item_index == 0:
                print(str(item_process_time) +  '>' + str(
                    visibility_interval[1]))

bottleneck.pop(0)
bottleneck.pop(3)
```

```
#caminho escolhido por cada caso: A6 ou A7
activity_pay_percentage = (count_a6/lines)*100
activity_letter_percentage = (count_a7/lines)*100
print("====CAMINHO ESCOLHIDO====")
print(f'{activity_pay_percentage:,.2f}% dos casos foram pelo
    caminho PAY')
print(f'{activity_letter_percentage:,.2f}% dos casos foram pelo
    caminho SEND LETTER\n')

#porcentagem de casos em atraso pelos intervalos de visibilidade
     do CPN Tools
print("====ATRASO====")
print('De um total de ' + str(lines) + ' casos:')
process_delay_percentage = ((count/lines)*100)
print(f'{process_delay_percentage:,.2f}% dos casos terminam o
    processo com atraso (> 105)')

#para os casos de atraso, onde est o os atrasos
activity_delay_percentage = ((sum(bottleneck)/activities_count)
    *100)
print(f'No geral, {activity_delay_percentage:,.2f}% das
    atividades s o disparadas com atraso, de forma que:')
for i in range(len(bottleneck)):
     index_activity = str(i+1) if i != 5 else '6/A7'
     print('A' + index_activity + ': ' + str(bottleneck[i]) + '
        casos')
```