WorkFlow net Possibilística aplicada aos Sistemas de Gerenciamento de Processos de Negócios Flexíveis

Leiliane Pereira de Rezende

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

Uberlândia
2017

Leiliane Pereira de Rezende

WorkFlow net Possibilística aplicada aos Sistemas de Gerenciamento de Processos de Negócios Flexíveis

Tese de doutorado 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 Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Stéphane Julia

Uberlândia
2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

R467w
2017

Rezende, Leiliane Pereira de, 1988-
WorkFlow net possibilística aplicada aos sistemas de gerenciamento de processos de negócios flexíveis / Leiliane Pereira de Rezende. - 2017.
313 f. : il.

Orientador: Stéphane Júlia.
Tese (doutorado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Negócios - Processamento de dados - Teses. 3. Fluxo de trabalho - Teses. 4. Gerenciamento de recursos de informação. - Teses. I. Júlia, Stéphane. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da tese intitulada "**WorkFlow net Possibilística aplicada aos Sistemas de Gerenciamento de Processos de Negócios Flexíveis**" por **Leiliane Pereira de Rezende** como parte dos requisitos exigidos para a obtenção do título de **Doutora em Ciência da Computação**.

Uberlândia, 23 de Junho de 2017

Orientador: _____

Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Carlos Roberto Lopes
Universidade Federal de Uberlândia

_____

Prof(a). Dr(a). Fernanda Francielle de Oliveira Malaquias
Universidade Federal de Uberlândia

_____

Prof. Dr. Paulo Eigi Miyagi
Universidade de São Paulo

_____

Prof. Dr. Ricardo Lüders
Universidade Tecnológica Federal do Paraná

_____

Prof(a). Dr(a). Janette Cardoso
Institut Supérieur de l'Aéronautique et de l'Espace

Dedico este trabalho à minha mãe Odelice e à minha irmã Luliane, que juntas me deram forças quando as minhas falharam e amparo quando os caminhos se tornaram difíceis de percorrer sozinha.

Agradecimentos

Gostaria de agradecer ...

Primeiramente a Deus, pela graça da vida, por renovar a cada momento as minhas forças e disposição, e pelo discernimento concedido ao longo dessa jornada.

À minha mãe, o alicerce da minha vida, por sempre me apoiar, pelas palavras de carinho, pelos sábios conselhos, pela amizade verdadeira e pelo amor incondicional a mim dedicado.

À minha irmã, pela presença constante, pelo carinho, paciência, confiança, amizade, cuidado, apoio, e principalmente pelo amor, em todos os momentos da minha vida.

Ao meu pai e ao meu avô, que no curto período de tempo a nós permitido, me ensinaram a ser forte e brigar pelo que desejo, me deram esperança de um mundo melhor e a consciência de auxiliar os menos favorecidos.

Às minhas tias, tios, primas, primos, amigos e cunhado pelos momentos compartilhados de pura diversão, amor e alegria, os quais, distantes ou não, eu sei que posso contar com eles.

À Simone e ao Gérard Julia pelo acolhimento durante meu ano na França, cuidando-me e preocupando-se comigo de forma tão carinhosa.

À professora Janette Cardoso por me proporcionar um ano de muitas experiencias, seja no campo profissional ou pessoal.

Em especial, ao professor Stéphane Julia pelo profissionalismo, por acreditar em mim, por me mostrar o caminho da ciência, mas acima de tudo pela preocupação, paciência, compreensão, confiança e orientação dado a mim ao longo desta jornada.

Por fim, a todos os professores que contribuíram pelo meu aprendizado, pela amizade, carinho e estímulo a mim dados ao longo de todo o trabalho. Ao secretário da FACOM, Erisvaldo, pelo profissionalismo e por sempre ter me ajudado quando precisei e à CAPES pelo apoio financeiro.

Aos que me ajudaram a ser quem sou, que depositam confiança em mim e para os quais sou uma esperança, resta-me afincadamente não vos desiludir.

A todos, o meu muito obrigada !

"Construí amigos, enfrentei derrotas, venci obstáculos, bati na porta da vida e disse-lhe:
Não tenho medo de vivê-la!"
(Augusto Cury - O Vendedor de Sonhos)

Resumo

Os Sistemas de Gerenciamento de *WorkFlow* (WfMS's) estão sendo cada vez mais usados pelas organizações para reduzir custos e melhorar o desempenho e a eficiência dos processos de negócios. Eles são usados para definir, registrar e coordenar a execução de processos de negócio estruturados por grupos de pessoas e geralmente compartilhados entre várias organizações. No campo da engenharia de sistemas, a atividade de modelagem desempenha um papel chave na compreensão e no controle do comportamento dos sistemas de gerenciamento de processos. Devido às mudanças no ambiente de trabalho, os modelos dos processos de negócios (usados nos WfMS's) devem ser continuamente adaptados, podendo torná-los extremamente complexos. Além disso, tais modelos podem ou não respeitar as boas propriedades que garantem a noção de correção do modelo do processo de negócio. Logo, a habilidade de se autoadaptarem à disponibilidade dos recursos, às falhas do sistema, às modificações do ambiente e às necessidades dos clientes é um fator chave para os WfMS's. Considerando isto, o termo flexibilidade, como sendo a habilidade de se ajustar tanto ao comportamento e às estruturas quanto à necessidade de assegurar a sobrevivência de um sistema, especialmente na presença de comportamentos incertos, tem sido um importante tópico de estudo nos processos de negócios. No entanto, nos modelos de processos de negócios, geralmente existe pouca ou nenhuma flexibilidade proporcionada pelos sistemas para acomodar a evolução natural dos mesmos ou os objetivos organizacionais. Considerando que a falta de flexibilidade nos WfMS's dificulta o suporte às situações inesperadas ocasionadas nas práticas de trabalho, o modelo de *WorkFlow net* possibilística foi estendido à teoria das *WorkFlow nets* interorganizacionais e aplicado aos problemas de modelagem tais como desvios, falhas de comunicação, *deadlock* e cancelamento de processos a fim de tratar a flexibilização dos processos de *workflow* em tempo de execução. Além disto, um mecanismo de inferência especializado do tipo *token player*, baseado no novo modelo apresentado, é definido, implementado, simulado e validado na ferramenta CPN Tools, cujo papel principal é mostrar a execução do modelo proposto aplicado a cinco modelos de processos de negócio diferentes. Com o uso da ferramenta CPN Tools, foi possível avaliar a resolução de situações inesperadas

em tempo de execução pelo *token player* implementado considerando possíveis situações de desvios de comportamento nos processos de negócios.

Palavras-chave: Sistema de Gerenciamento de *WorkFlow*, processo de negócios, *WorkFlow net*, *WorkFlow net* Interorganizacional, rede de Petri Possibilística, flexibilidade de processo, tolerância a falha, *deadlock*, cancelamento de processo, CPN Tools.

Abstract

Organizations are increasingly using Workflow Management Systems (WfMS) to reduce costs and improve the performance and efficiency of important business processes. WfMS are used for the modeling, analysis and coordination of structured business processes by groups of people or among multiple organizations. In the field of systems engineering, modeling plays a key role for understanding and controlling the behavior of the systems. Due to changes in the workplace, the business process models (used in WfMS's) should be continuously adapted which would make them extremely complex. Furthermore, such models may or may not respect the good properties which guarantee the correctness notion of the business process model. Then, the ability to dynamically self-adapt to resource variability, system faults, changing environments and user needs is a key factor for WfMS. Considering this, the term flexibility, as the ability to adjust both behaviors and structures as the necessity to ensure survival, especially in the presence of uncertain behavior, especially in the presence of uncertain behaviors, has been an important topic of study in business processes. Nevertheless, in the business process model, there exists generally little or no flexibility provided by systems to accommodate the natural evolution of the work process or organizational goals. Considering that the lack of flexibility in WfMS makes difficult the support unexpected situations occurring in the work practices, the possibilistic WorkFlow net model was extended to interorganizational WorkFlow nets theory and applied to modeling problems such as deviation, communication failures, deadlock and cancellation in order to treat the flexibility in workflow processes during the time execution. In addition, a specialized inference mechanism, called token player, based on the new model presented is defined, implemented, simulated and validated in the CPN Tools, whose main role is to show the execution of the model proposed aplied to five different business process models. Using the CPN Tools, it was possible to evaluate the resolution of unexpected situations in time execution by token player implemented considering possible situations of behavioral deviations in business processes.

Keywords: Workflow Management System, business processes, WorkFlow net, interorganizational WorkFlow net, possibilistic Petri nets, process flexibility, fault tolerance, deadlock, process cancellation, CPN Tools.

Lista de ilustrações

Lista de tabelas

Lista de siglas

AHP Analytical Hierarchy Process

AOP Aspect-Oriented Programming

BPM Business Process Management

BPMN Business Process Model and Notation

CPN Colored Petri Net

FMS's Flexible Manufacturing Systems

IA Inteligência Artificial

ICP Input Communication Place

IOWF-net WorkFlow net Interorganizacional

OCP Output Communication Place

PIPE Plataform Independent Petri net Editor

RPN Rede de Petri Nebulosa

U(IOWF-net) Unfolded Interorganizational WorkFlow net

UML Unified Modeling Language

WfMS's WorkFlow Management Systems

WF-net WorkFlow net

YAWL Yet Another WorkFlow Language

Lista de símbolos

$A(N, M)$	conjunto das marcações acessíveis
A_V	vetor de variáveis formais
A_{tc}	associação de uma condição de disparo a cada transição
A_{ta}	associação de uma ação de disparo a cada transição
AC	relação de comunicação assíncrona
b	objeto pertencente à uma rede de Petri a objeto
B	conjunto de objetos
C_e	conjunto de quatro elementos $\{+, -, 0, \pm\}$
C_d	o conjunto de sequências de disparo que conduzem o estado do processo a um estado de *deadlock*
C_s	conjunto de sequências de disparo
C_{aso}	classe de objeto "$Caso$"
C_{or}	conjunto finito de cores de uma rede de Petri Colorida
C_{sc}	função *sub-conjunto de cores* de uma rede de Petri Colorida
D	matriz de incidência de sinais
E	função de expressões de arco de uma rede de Petri Colorida
E_d	conjunto de estados de *deadlock*
F	valor falso do tipo de dado primitivo Booleano
$F^{\blacktriangledown}$	relação entre os elementos dos modelos em WF-net que compõem uma IOWF-net
$F^{\blacktriangle}$	relação entre os elementos dos modelos em WF-net possibilística que compõem uma IOWF-net possibilística

G	função de guarda de uma rede de Petri Colorida
G_A	grafo das marcações acessíveis
H_i	Sifão i qualquer
i	lugar de início de uma *WorkFlow net*
I	matriz de incidência
l_i	uma linha i qualquer de uma matriz
LL	Lista de lugares
$LEAF$	elemento de $PLANT$ candidato a um lugar-sifão mínimo
m	quantidade de lugares
M	marcação
M'	uma marcação qualquer
M_i	marcação i qualquer
M_0	marcação inicial
M_f	marcação final
n	quantidade de transições
$\mathbb{N}$	conjunto dos números naturais
N	rede de Petri marcada
N_{CPN}	rede de Petri colorida marcada
N_{PT}	rede Predicado-Transição marcada
N_{RPO}	rede de Petri a Objeto marcada
o	lugar de término de uma *WorkFlow net*
p_i	lugar i qualquer
P	conjunto finito de lugares
$P^{\blacktriangledown}$	união dos conjuntos finitos de lugares pertencentes aos modelos em WF-net que compõem uma IOWF-net
$P^{\blacktriangle}$	união dos conjuntos finitos de lugares pertencentes aos modelos em WF-net possibilística que compõem uma IOWF-net possibilística
P'	conjunto de lugares não vazio denominado como sifão
P''	conjunto de lugares não vazio denominado como trap
P_{AC}	conjunto de elementos de comunicação assíncrona (lugares de comunicação)

PN_k	um modelo de processo em WF-net possibilística com lugar de início i_k e lugar de término o_k com $k \in \{1, \ldots, w\}$
$Post$	matriz de pós-condição
Pre	matriz de pré-condição
$PLANT_j$	o conjunto de sifões obtido pelo $SPROUT$ a partir de $SEED_j$
q	quantidade de processos modelados em WF-net
R	rede de Petri sem marcação
R_r	rede *Reset*
s	sequência de disparo
S_n	substituição das variáveis associadas aos arcos de entrada de uma transição, com $n \in \mathbb{N}$
SB	conjunto de sifões básicos
SM	conjunto de sifões mínimos
SR	uma transição com arcos de entrada saindo dos lugares o de cada modelo e com arcos de saída entrando nos lugares i de cada modelo
$SEED_j$	lugar p_j correspondente à coluna D_j escolhida no passo 1 do algoritmo de detecção de sifão
$SPROUT$	processo recursivo descrito na 1ª ETAPA do algoritmo
t_i	transição i qualquer
T	conjunto finito de transições
T^*	subconjunto finito de transições
$T^{\blacktriangledown}$	união dos conjuntos finitos de transições pertencentes aos modelos em WF-net que compõem uma IOWF-net
$T^{\blacktriangle}$	união dos conjuntos finitos de transições pertencentes aos modelos em WF-net possibilística que compõem uma IOWF-net possibilística
T^D	conjunto das transições responsáveis diretamente e indiretamente pelo estado de *deadlock*
T^E	conjunto de transições de entrada especificado pela relação AC
T^S	conjunto de transições de saída especificado pela relação AC
U	valor incerto
w	quantidade de processos modelados em WF-net possibilística
W_i	lugar de espera i
WF	WorkFlow net

WP	WorkFlow net possibilística
V	valor verdadeiro do tipo de dado primitivo Booleano
V_s	vetor característico da sequência de disparo S
V_f	conjunto de variáveis formais
X	somatório de vetores colunas correspondentes aos lugares pertencentes à CP

Sumário

Capítulo 1

Introdução

Um processo de negócio pode ser definido como um conjunto de atividades inter-relacionadas que podem, ou não, ser executadas simultaneamente, com alguma especificação de controle e fluxo de dados entre as atividades com o objetivo de obter resultados com valor para um cliente, sendo este interno ou externo à organização (SHARP; MCDERMOTT, 2001) (PADUA et al., 2004). Devido à complexidade e à variedade dos mesmos, as organizações contemporâneas usam a tecnologia da informação para automatizá-los ou apoiar as suas atividades.

A área de Gerenciamento de Processos de Negócio, do inglês *Business Process Management (BPM)*, tem recebido uma considerável atenção nos últimos anos, de acordo com Aalst et al. (2010), devido ao seu potencial em aumentar significativamente a produtividade das organizações e em economizar gastos. Além disso, os autores destacam que o conceito de processo é fundamental e serve como um ponto de partida para o entendimento de como o negócio opera e quais oportunidades existem para coordenar suas atividades constituintes.

A automação de um processo de negócio, no todo ou em parte, durante o qual documentos, informações ou tarefas são passadas de um participante para outro para a realização de ações, de acordo com um conjunto de regras procedurais, é dada através de um *workflow* (MEMBERS, 1994) (LOMAZOVA, 2010). Tipicamente, os *workflows* são usados para descrever a interação entre empregados, recursos materiais e computacionais.

Os sistemas de Gerenciamento de *WorkFlow*, do inglês *WorkFlow Management Systems (WfMS's)*, são pacotes de software genéricos para gerenciamento de processos de negócio (ALONSO et al., 1997) (ESHUIS, 2002) (AALST; HEE, 2004). Eles têm como principal objetivo suportar a definição, a execução, o registro e o controle de processos de negócios (AALST, 1998a). Nos WfMS's, tais processos são modelados em *workflows* permitindo a definição da sequência das atividades humanas e computacionais, incluindo interação com ferramentas e aplicações auxiliares bem como os responsáveis e recursos de cada uma (HOLLINGSWORTH, 1994). Em particular, os WfMS's permitem o gerenciamento das atividades fazendo com que a informação certa chegue até a pessoa certa no

momento certo Aalst (1998a).

As três funções principais que caracterizam os WfMS's são: construção, controle da execução e interação (MEMBERS, 1994). Para a função de construção existem ferramentas para definição de *workflows* e para modelagem dos processos de negócios. A função de controle de execução é desempenhada pelo motor do sistema. Tal motor instancia os modelos dos processos e com base nestes modelos sequencia as ações a serem realizadas. Esta sequenciação é chamada de execução do modelo. Por fim, a função de interação consiste em gerenciar a interação de ações com usuários e outras aplicações para realização das atividades descritas.

Cada uma das três funções anteriormente citadas pode ser desempenhada por uma ou mais ferramentas. Dessa forma, os WfMS's têm funcionalidades específicas para: modelagem dos processos, execução dos modelos, interação com usuário, interação com aplicações auxiliares, monitoração da execução dos modelos dos processos e interação com outros WfMS's. A tarefa de gerenciamento nos WfMS's é feita a partir de modelos do processo ou da definição do *workflow*. Uma vez definido, o motor do sistema realiza a instanciação e a execução do modelo do processo. Durante a execução, o motor interpreta as ações definidas e provê os recursos necessários para a sua realização, sejam aplicações externas ou a realização de tarefas por empregados da empresa. Os WfMS's monitoram assim a execução dos modelos dos processos em tempo real (MEMBERS, 1994).

Quando os *workflows* envolvem apenas a ordenação rígida das atividades dos processos de negócios, a coordenação das atividades é relativamente simples. Entretanto, tendo em vista a dificuldade de modificar os modelos dos processos de negócios (KAPURUGE; HAN; COLMAN, 2010), os mesmos são frequentemente sujeitos à intervenções manuais na medida que os funcionários humanos tentam manipulá-los de forma a adaptarem-os às mudanças das práticas de trabalho. Como consequência, a produtividade é prejudicada e o tempo de processamento pode aumentar. Além disto, tal intervenção acarreta outra penalidade: as ações corretivas realizadas de forma *ad-hoc*[1] não costumam ser adicionadas à memória organizacional; logo as mesmas não são incorporadas nas futuras iterações dos modelos dos processos (ADAMS, 2010).

Considerando ainda que as organizações modernas lidam com processos administrativos relativamente complexos, envolvendo inúmeros usuários, assim como diversas instâncias em execução simultânea e distribuídas em uma mesma rede (ALONSO et al., 2000), os WfMS's devem suportar *workflows* compartilhados entre várias organizações. Cada parceiro de negócio tem então seu *workflow* privado conectado aos *workflows* de outros parceiros (SILVA et al., 2013), produzindo, assim, um *workflow* interorganizacional composto por um conjunto finito de *workflows* fracamente acoplados essencialmente por meio de mecanismos de comunicação assíncrona (AALST, 1998b). O funcionamento de muitas organizações depende criticamente de modelos que descrevam precisamente a dinâmica

[1] ad-hoc é uma expressão em latim que significa "para isto", "para um fim específico".

desses acoplamentos. Além disso, a maioria das aplicações devem aceitar situações, tais como falhas de comunicação, falta de resposta por parte dos usuários, prazos não cumpridos, bloqueio total (*deadlock*) e comportamentos inesperados das aplicações (ALONSO et al., 2000).

Um fator chave para avaliar os WfMS's é a capacidade de se adaptarem dinamicamente aos novos ambientes, à disponibilidade dos recursos, às mudanças nas necessidades dos usuários e às possíveis falhas do sistema (HSIEH; LIN, 2013). Para tanto, o termo flexibilidade é usado para designar o grau ao qual um sistema é capaz de lidar ou suportar os desvios esperados ou não durante a execução das instâncias do modelo do processo, tanto dentro do contexto da mesma ou a partir de ambientes externos, sem impacto negativo sobre o processo ou acerca de sua realização(ADAMS, 2010).

Flexibilidade, tanto no nível de processo quanto no de produto, tem sido um tema importante nos sistemas de manufatura desde os anos oitenta (TOLIO, 2009). Do ponto de vista de modelo de um processo, ela é definida como a capacidade de ajustar tanto o comportamento quanto a estrutura do modelo de forma a assegurar a continuidade da realização do processo especialmente diante de comportamentos incertos (GÜNSEL et al., 2012). As pesquisas em torno deste tema permitiram aos sistemas de manufatura responderem às demandas dos clientes mesmo quando situações inesperadas e vagas ocorrem, além de casos onde informações aleatórias e aproximadas sobre a forma de realização dos processos são consideradas (TOLIO, 2009) (ASATO et al., 2011) (KHALID; YUSOF; SABUDIN, 2012) (SILVA et al., 2014). No entanto, no setor administrativo, a flexibilidade proporcionada pelos WfMS's para acomodar a evolução dos modelos dos processos ou das metas organizacionais a serem atingidas é insignificante ou nem existe (BHATT, 2015).

A falta de flexibilidade nos processos de negócios ocorre principalmente por causa da dificuldade em prever todas as alternativas de execução possíveis ao especificar um *workflow* (HEINL et al., 1999). Além disso, a necessidade de adaptar continuamente os *workflows* de acordo com a evolução das práticas e necessidades de negócio torna-os complexos e extensos, despendendo muito de esforço humano para o entendimento e manutenção dos mesmos, uma vez que a configuração original pode se tornar difícil de ser reconhecida após sofrer numerosas redefinições (BUIJS et al., 2012).

Assim, a presente pesquisa visa prover um mecanismo responsável por monitorar em tempo real processos de negócios flexíveis. Para tanto, uma nova técnica de modelagem juntamente com um novo motor do sistema são definidos com o propósito de suportar possíveis falhas de comunicação, inconsistências/desvios/exceções e decisões humanas não completamente previstas. Além disto, possíveis problemas estruturais, os quais, em alguns casos, não permitem a evolução correta do processo (situações de *deadlock* por exemplo) são contornados.

1.1 Motivação

Para Lomazova (2010), a modelagem dos processos de negócios é a primeira e mais importante fase no ciclo de vida dos WfMS's. Várias linguagens de modelagem podem ser utilizadas no contexto da modelagem de processos de negócio como, por exemplo, *Business Process Model and Notation (BPMN)* (OMG, 2011a), *Yet Another WorkFlow Language (YAWL)* (AALST et al., 2010) (AALST; HOFSTEDE, 2005), rede de Petri (MURATA, 1989) e, até mesmo, diagramas de atividade da *Unified Modeling Language (UML)* (OMG, 2011b).

As linguagens de modelagem podem ser classificadas como formal, informal ou semi formal. Entende-se por linguagem formal aquela que possui um conjunto de regras de formação bem-definidas e um conjunto de regras de derivação que permitem desconsiderar o significado do que está descrito no sistema e trabalhar apenas com a sua forma, com o intuito de verificar novos significados ou propriedades (LOH; CASTILHO, 1991). Uma linguagem informal é toda aquela que não é formal e uma linguagem semi formal faz uso dos dois tipos de linguagens, isto é, a especificação é feita considerando tanto a linguagem formal quanto a informal (LOH; CASTILHO, 1991). É importante frisar que ambiguidades e confusões não podem ser prevenidas em um modelo de processos de negócio descrito por uma linguagem informal ou semi formal. Para resolver este problema, é necessário um modelo de processos de negócio com uma semântica formal (AALST; HEE, 2004).

Modelos com uma semântica formal não são usuais para profissionais da área de negócio. No entanto, uma rede de Petri pode auxiliar na solução desse problema, pois possuí representação gráfica, é de fácil aprendizado, funciona como linguagem de comunicação entre especialistas de diversas áreas e permite a descrição dos aspectos estáticos e dinâmicos do sistema a ser representado (MURATA, 1989). Além disto, permite a utilização de importantes métodos de análise (CARDOSO; VALETTE, 1997).

Uma extensão da rede de Petri para modelagem de processos de negócios foi definida por Aalst (1998a) como WorkFlow net (WF-net). Vários trabalhos passaram a considerar este modelo formal e específico como referência na área de modelagem e análise, usando-o em suas abordagens como pode ser verificado, por exemplo, em (AALST, 1998a) (AALST; HEE, 2004) (KOTB; BADREDDIN, 2005) (JESKE; JULIA; VALETTE, 2009) (AALST et al., 2011) (PLA et al., 2012) (HSIEH; LIN, 2014) (GHARIB; GIORGINI, 2015).

Geralmente, um modelo de processo permite melhorar a comunicação entre as pessoas e apoia a automação (CUGOLA et al., 1995). Por outro lado, os processos de negócio ao serem automatizados, independentemente da linguagem de modelagem utilizada, tendem a estabelecer a forma como as empresas organizam o trabalho, forçando-as a ajustá-lo ao sistema de automação. Em outras palavras, uma empresa que utiliza tais sistemas normalmente não é capaz de implementar seus processos comerciais de outras formas mesmo que mais apropriada à companhia. Com isso, os processos de negócios são modelados de forma a se "encaixarem no sistema", podendo causar diversos problemas (PESIC, 2008).

Um dos principais é a incompatibilidade entre a forma preferida de trabalhar e a forma como o sistema opera. Este problema pode forçar as empresas a "executarem" seus processos de negócios inadequadamente. Outro problema a destacar é a possibilidade da criação de duas realidades paralelas: a forma como é executado "fora do sistema" e a forma como é registrado no sistema da empresa. Estes tipos de problemas podem inibir o uso de técnicas como os WfMS's nas companhias e, além disso, nem sempre é possível definir um modelo do processo que "atenda" a todas as necessidades de uma organização ao longo do tempo.

Heinl et al. (1999), a fim de reportar os problemas decorrentes do uso dos WfMS's, realizaram um estudo de caso numa grande empresa, que utiliza WfMS's para apoiar mais de 400 processos. Deste estudo quatro problemas foram encontrados: (1) a quase impossibilidade de identificar antecipadamente todas as etapas do processo de negócio; (2) mesmo se uma etapa é identificada, não é óbvio como incluí-la no modelo do processo; (3) nem sempre é possível prever a ordem dos passos previamente definidos; (4) mapear os processos de negócios para os modelos de processo é propenso a erros. Lembrando ainda que organizações de grande porte abrangem uma quantidade significativa de pessoas responsáveis por realizarem um grande número de processos e que cada uma delas se comportam de uma maneira diferente em diferentes situações, os modelos dos processos estão cada vez mais extensos e complexos requerendo um controle efetivo para evitar erros (MOHAMMED; REDOUANE; BERNARD, 2007). Entretanto, ao mesmo tempo, precisam de mais flexibilidade para lidar com as situações imprevistas, normalmente ocasionadas pelas discrepâncias entre as atividades do mundo real e as representações formais das mesmas.

A importância da flexibilidade nos WfMS's tem sido reconhecida por muitos pesquisadores desde meados dos anos noventa (PESIC, 2008) (ADAMS, 2010). As principais razões para a ascensão deste conceito são a globalização dos mercados e a ênfase na qualidade e na personalização. Como resultado, a flexibilidade, como a habilidade de se adaptar a novos ou a diferentes requisitos ou ainda à modificação dos mesmos, emergiu como uma vantagem competitiva e um requisito necessário em muitas atividades organizacionais (GÜNSEL et al., 2012)

Como discutido por muitos pesquisadores, para se manterem no ambiente dinâmico do mercado global, as organizações devem ser ágeis e adaptáveis e, devido a isto, as mesmas estão ávidas para tirar vantagem de processos de negócios flexíveis (GOLDEN; POWELL, 2000) (CHANOPAS; KRAIRIT; KHANG, 2006) (QI; LUO, 2007). Considerando isto, diversas tentativas para classificar o termo flexibilidade têm sido propostas na literatura (CARLSEN; KROGSTIE; LINDLAND, 1997) (HEINL et al., 1999) (AALST; JABLONSKI, 2000) (SOFFER, 2005) (KUMAR; NARASIPURAM, 2006) (SNOWDON et al., 2007) (SCHONENBERG et al., 2008b) (SCHONENBERG et al., 2008a).

A primeira taxonomia abrangente referente às características concretas que aprimoram

a flexibilidade nos WfMS's foi proposta em 1999 por Heinl et al. (1999). Mais tarde, Schonenberg et al. (2008b) revisaram a questão da flexibilidade e ajustaram a taxonomia original proposta por Heinl et al. (1999) aos desenvolvimentos das tecnologias relacionadas aos *workflows* (SNOWDON et al., 2007) (SCHONENBERG et al., 2008a). Quatro tipos principais de flexibilidade foram propostos por eles:

1. *flexibilidade de design*[2] que corresponde à capacidade de incorporar caminhos de execução alternativos no modelo do processo em tempo de *design*.

   Os padrões de *design* definidos por Aalst et al. (2003) para os *workflows* podem ser vistos como uma classificação deste tipo de flexibilidade para as linguagens imperativas[3]. Porém, descrever todos os possíveis caminhos de execução em um modelo de processo em tempo de *design* pode ser indesejável do ponto de vista da complexidade do modelo ou impossível devido ao número desconhecido ou ilimitado dos possíveis caminhos de execução (SCHONENBERG et al., 2008b);

2. *flexibilidade por desvio*[4] que corresponde à capacidade de uma instância se desviar, em tempo de execução, do caminho prescrito pelo modelo do processo original, sem alterá-lo fisicamente.

   O conceito de desvio é particularmente adequado para a especificação dos modelos de processos, os quais se destinam a guiar as possíveis sequências de execução em vez de restringir as opções que estão disponíveis. Nas linguagens imperativas, isto pode ser obtido através da aplicação de novas regras; no entanto, isso pode resultar em muitos esforços para implementar várias mudanças repetidamente. Já, nas abordagens declarativas[5], desvios ocorrem basicamente através da violação de restrições opcionais (SCHONENBERG et al., 2008a);

3. *flexibilidade por subespecificação*[6] que corresponde à capacidade de executar um modelo de processo com uma especificação incompleta, isto é, um modelo com informações insuficientes para permitir que o mesmo seja concluído.

   Neste tipo de flexibilidade, as partes não especificadas do modelo são finalizadas em tempo de execução através de um link entre (1) a parte que falta e alguma funcionalidade pré-especificada ou (2) através da modelagem de uma nova funcionalidade (ROSA et al., 2013). Em (1), um fragmento do modelo do processo é selecionado a

[2] do inglês "Flexibility by design"

[3] Nas linguagens imperativas o modelo é visto como um conjunto de rotinas e sub rotinas e enfatiza-se as mudanças de estado do programa (JONES; L.; WADLER, 1993)

[4] do inglês "Flexibility by deviation"

[5] Nas linguagens declarativas, o foco é descrever o que o modelo deve fazer e não em como seus procedimentos devem funcionar, ou seja, as linguagens declarativas enfatizam a descrição declarativa de uma tarefa em vez de sua decomposição passo a passo como em uma definição algorítmica do fluxo de execução de uma máquina (HARTEL et al., 2001).

[6] do inglês "flexibility by underspecification"

partir de um conjunto preexistente de fragmentos totalmente predefinidos. Observe-se que a seleção é limitada e não é permitido modelar um novo fragmento durante a execução do sistema. Em (2), um novo fragmento do modelo do processo é modelado a partir do zero ou composto por fragmentos de outros modelos de processos existentes. A construção deste novo fragmento pode ser limitado por um conjunto de restrições (SMANCHAT; LING; INDRAWAN, 2008);

4. *flexibilidade por mudança*[7] que corresponde à capacidade de modificar o modelo do processo em tempo de execução de forma que uma ou todas as instâncias em execução sejam migradas para o novo modelo.

   A complexidade relacionada a este tipo de flexibilidade se deve ao fato de que para cada instância em execução a ser migrada para o novo modelo, é necessário encontrar um estado que corresponda ao seu atual no novo modelo do processo, o que nem é sempre possível. Além disso, como cada instância em execução contém uma história, a qual determina o seu estado atual, vários tipos de problemas podem ocorrer quando uma mudança ocorre como, por exemplo, falta ou perda de dados, situações de *deadlocks*, retrabalhos e outros conforme indicado em (ELLIS; KEDDARA; ROZENBERG, 1995) e (RINDERLE; REICHERT; DADAM, 2004). Weber, Reichert e Rinderle-Ma (2008) descrevem um conjunto de padrões de mudança indicando em cada padrão os problemas enfrentados quando se desejam inseri-los em um modelo de processo.

As abordagens já propostas na literatura suportam no máximo três tipos das flexibilidades acima descritas. A abordagem que mais contempla estes tipos de flexibilidades é a linguagem de modelagem de processos YAWL proposta por Adams (2010) e inspirada na WF-net. Entretanto, esta abordagem não garante, por exemplo, a propriedade de correção *Soundness*[8] após qualquer alteração no modelo (AALST; HEE, 2004). Tal propriedade garante que, uma vez iniciada a execução de um modelo de processo de negócio, o mesmo será totalmente concluído sem restar qualquer tarefa pendente.

Outras abordagens, tais como *Declare* (PESIC; AALST, 2006), ADEPT (DADAM; REICHERT, 2009) e FLOWer (AALST; WESKE; GRÜNBAUER, 2005), incorporam outros tipos de flexibilidades nos seus modelos. Entretanto, *Declare* e ADEPT não têm um fluxo de controle pré-definido a seguir durante a execução, mas um conjunto de restrições a respeitar. No ADEPT, ao realizar as trocas no modelo do processo em tempo de execução, a propriedade de correção *Soundness* pode não ser mais garantida. Para garantí-la, faz-se necessário uma reavaliação do modelo. Outras propostas, como por exemplo, (THOMPSON; TORABI, 2009), (ADAMS, 2010), (ROZINAT; AALST, 2008), (ADRIANSYAH; DONGEN; AALST, 2010), (SILVA et al., 2010), (AALST; ADRIANSYAH; DONGEN,

[7] do inglês "flexibility by change"

[8] A propriedade de correção *Soundness* é definida na subseção 2.2.1

2012) e (BAEK; HAN; CHUNG, 2013), apenas detectam os desvios e/ou as inconsistências a partir da comparação do modelo de referência com uma instância do mesmo sendo executada ou tendem a perder algumas propriedades do modelo.

Uma alternativa para lidar com a flexibilidade nos modelos dos processos de negócio pode estar nas abordagens baseadas no conhecimento incerto como a apresentada em (CÎMPAN; OQUENDO, 2000). O modelo do processo é definido com variáveis *fuzzy* e distribuições de possibilidade permitindo representar a informação imprecisa que existe quando as atividades são executadas por humanos. Esta alternativa já tem sido utilizada nos sistemas de manufatura flexíveis, do inglês *Flexible Manufacturing Systems (FMS's)* (VALETTE; COURVOISIER; MAYEUX, 1982) (MURATA; SUZUKI; SHATZ, 1999) (ASATO et al., 2012). Entende-se por certo/incerto a propriedade resultante da falta de informação para decidir se a sentença é verdadeira ou falsa e, por preciso/impreciso a propriedade relacionada com o conteúdo de uma sentença (DUBOIS; PRADE, 1985).

Um dos primeiros estudos que combina a representação possibilística da informação com a estrutura precisa de uma rede de Petri quando se considera sistemas de eventos discretos é descrito em (CARDOSO, 1999) e (CARDOSO; VALETTE; DUBOIS, 1999). A principal característica da rede de Petri possibilística/*Fuzzy* é permitir modelar aspectos de incertezas e mudanças na dinâmica dos sistemas de eventos discretos. Dessa forma, possíveis desvios que acontecem durante a execução do modelo do processo são descritos evitando que a ocorrência deles, em relação a um modelo de referência, se transforme numa inconsistência. Entretanto, a maioria das abordagens que englobam o uso da rede de Petri *Fuzzy* foi aplicada em processos industriais dos FMS's.

As abordagens propostas em (REZENDE; JULIA; CARDOSO, 2012) e (REZENDE; JULIA; CARDOSO, 2012) são baseadas em marcações imprecisas e disparos incertos derivados da rede de Petri possibilística para permitirem a ocorrência de certos desvios sem que inconsistências sejam geradas durante a execução do modelo do processo. Uma das limitações desta abordagem é que foram considerados apenas modelos de processos modelados por uma WF-net que respeita a propriedade *Soundness* (AALST; HEE, 2004). Além disto, na abordagem proposta, somente quando o disparo é certo é que uma ação pode ser executada. Desta forma, possíveis tratamentos automáticos quando o disparo é incerto, por exemplo, são impossibilitados ou devem ser realizados de forma manual.

Um modelo de um processo de negócio que atende à propriedade *Soundness* é ideal, porém, sabe-se que, na prática, nem sempre tal propriedade é satisfeita nos modelos dos processos. Considerando a importância do critério de correção *Soundness*, Fahland et al. (2011) verificaram 735 processos de negócios de vários domínios, como telecomunicações e serviços financeiros. Os autores mostram que apenas 46% dos modelos dos processos verificados são, de fato, *sound*. Ou seja, a maioria deles não satisfizeram o critério de correção *Soundness* e mesmo assim se encontram em execução. O trabalho desenvolvido por Fahland et al. (2011) permite concluir que, apesar da importância do critério de

correção *Soundness*, na prática, em quase metade dos casos, os modelos dos processos não satisfazem tal critério, o que pode permitir, por exemplo, a ocorrência de situações de *deadlock*.

Considerando a possibilidade de existência de *deadlocks* durante à execução dos modelos dos processos, muitos pesquisadores, tais como (EZPELETA; COLOM; MARTÍNEZ, 1995), (GANG; MING, 2004), (KOHLER; SCHAAD, 2008), (LI et al., 2012) e (MOHANTY; KUMARA, 2013), têm proposto métodos de análise ou de detecção, durante a execução dos mesmos, com o objetivo de encontrar as possíveis ocorrências de *deadlocks* relacionados à alocação de recursos a fim de evitá-las durante a execução. Outras propostas, como por exemplo, (BARKAOUI; ABDALLAH, 1995), (CHU; XIE, 1997), (IORDACHE; MOODY; ANTSAKLIS, 2002), (AWAD; PUHLMANN, 2008) e (SILVA et al., 2013), modificam o modelo do processo inserindo uma estrutura de controle responsável por remover qualquer situação de *deadlock*.

Levando em conta a crescente cooperação interorganizacional entre empresas distribuídas ao redor do mundo e os mercados tão suscetíveis às mudanças (BALESTRIN; VERSCHOORE; JUNIOR, 2010), os processos de negócio com seus diversos parceiros, cada um com seu modelo do processo local privado, comunicando entre si, estão sujeitos a problemas de sincronização e, consequentemente, situações de *deadlock* de modo que em um contexto interorganizacional garantir *Soundness* é ainda mais difícil. Isso acontece por vários motivos. Um motivo é o fato de que tais processos não são definidos em conjunto, ou seja, cada organização define seu modelo do processo individualmente e, depois de prontos, estes são colocados para trabalhar em conjunto. Além disso, as organizações nem sempre estão dispostas a modificar seus modelos.

Uma propriedade alternativa a *Soundness* é a *Relaxed Soundness* proposta por Dehnert e Rittgen (2001). A ideia desta propriedade é garantir que cada atividade associada ao *workflow* pertença a pelo menos uma sequência de execução que seja capaz de alcançar o estado final do mesmo. Desta forma, mesmo que existam situações de *deadlocks*, tal critério garante a correta conclusão para algumas de suas sequências de execução.

Observa-se que a necessidade de flexibilizar os processos de negócio se tornou um diferencial entre as organizações (AFFLERBACH et al., 2014). Cada vez mais os modelos dos processos precisam ser capazes de se adaptarem a certas mudanças de ambiente de forma a se adequarem às reais necessidades em tempo de execução. Entretanto, nota-se que em muitos casos, ao se alterar os modelos dos processos, algumas de suas propriedades e/ou características podem ser perdidas além de possibilitar um crescimento na complexidade do mesmo.

1.2 Formulação do Problema

Dada a motivação do trabalho, as seguintes questões de pesquisa foram levantadas:

- Como a combinação da estrutura rigorosa da WF-net definida por Aalst e Hee (2004) e da rede de Petri possibilística definida por Cardoso (1999) possibilita, tanto no caso mono-processo quanto no caso interorganizacional, a representação de comportamentos flexíveis nos WfMS's?
- Como manter as "boas propriedades" do modelo do processo ao permitir comportamentos flexíveis durante a execução do modelo do processo?
- Como problemas estruturais no modelo do processo podem ser evitados/contornados durante a execução?

1.3 Hipóteses

A fim de responder as questões de pesquisa seis hipóteses são assumidas:

- o padrão de mudança "*Parallelize Process Fragments*" proposto por Weber, Reichert e Rinderle-Ma (2008) para paralelizar fragmentos do modelo do processo que, originalmente, eram puramente sequencial, pode ser incorporado nos modelos dos processos a partir do uso da noção de disparo incerto;
- os padrões de projeto "Cancel activity" e "Cancel case" propostos por Aalst et al. (2003), responsáveis por capturar a interferência de uma atividade na execução das outras atividades em certas circunstâncias, podem ser incorporados nos modelos dos processos a partir do uso da noção de disparo incerto sem perder a decidibilidade da propriedade *Soundness*;
- as falhas decorrentes da comunicação entre os processos parceiros podem ser contornadas através da noção de disparo incerto nos modelos de processos interorganizacionais;
- os estados de *deadlock* presentes nos modelos dos processos interorganizacionais devido à sincronização entre os processos paralelos podem ser evitados em tempo de execução através da noção de disparo incerto;
- a ferramenta de modelagem *CPN Tools* permite, através do uso da linguagem de programação funcional *Standard ML*, implementar, simular e validar o uso da noção de disparo incerto a fim de avaliar a robustez das técnicas propostas.

1.4 Objetivos da Pesquisa

Em (REZENDE; JULIA; CARDOSO, 2012) e (REZENDE; JULIA; CARDOSO, 2012), os conceitos de marcação imprecisa e disparo incerto obtidos a partir da lógica possibilística foram aplicados à WF-net com o intuito de permitir uma maior flexibilização nos

processos de negócios modelados de forma procedimental - isto é, a ordem das atividades é explicitamente especificada - sem qualquer modificação na estrutura do modelo. Na abordagem proposta, através da noção de marcação imprecisa, os disparos incertos são explorados de forma a permitir a recuperação de possíveis eventos "perdidos" devido ao não respeito de certas restrições de tempo associadas com as atividade do modelo do processo. Na dissertação de mestrado de Rezende (2013), a WF-net possibilística juntamente com as regras de disparo foram definidas. Entretanto, o "jogador" (*token player*) aplicado aos problemas de desvios foi apresentado informalmente e não foi implementado e validado em nenhuma ferramenta de modelagem e simulação de rede de Petri. Além disto, todas as considerações foram realizadas apenas para os modelos de processos de negócio que respeitam a propriedade *Soundness* e nenhum processo de negócio compartilhado entre organizações foi considerado. Por fim, as ações definidas nos modelos dos processos modelados por uma WF-net possibilística são executadas somente se um disparo certo é efetuado; em qualquer outro tipo de disparo, nenhuma ação pode ser realizada dificultando, assim, a realização de um tratamento adequado quando situações imprevistas são ocasionadas. A partir dos resultados alcançados nessa pesquisa, pode-se delinear novos objetivos.

1.4.1 Objetivo Geral

O objetivo deste trabalho é propor um nova técnica de modelagem, nomeada *WorkFlow net* interorganizacional possibilística, que seja capaz de suportar possíveis falhas de comunicação e problemas estruturais ocasionadas entre processos parceiros. Além disto, a definição da técnica de modelagem, nomeada *WorkFlow net* possibilística definida em (REZENDE, 2013) é alterada com o intuito de contemplar uma ação para cada possível mudança de comportamento ocasionada durante a execução do modelo do processo de acordo com a qualidade das informações recebidas durante a sua execução. Por fim, um mecanismo de inferência especializado do tipo *token player* responsável por gerir, em tempo real, a execução do modelo do processo é definido.

1.4.2 Objetivos Específicos

- ❑ definir um procedimento baseado na WF-net possibilística capaz, em um modelo de processo estritamente sequencial, de iniciar a execução de novas atividades antes da conclusão de suas anteriores sem que as propriedades inerentes do modelo sejam prejudicadas;

- ❑ definir um procedimento baseado na WF-net possibilística capaz de cancelar a execução da instância do modelo do processo ou das atividades escalonadas ou em execução. Tal procedimento preservará a decidibilidade das boas propriedades do modelo;

- definir um procedimento baseado nas estruturas de tipo sifão da teoria da rede de Petri e na noção de disparo incerto capaz de contornar, em tempo de execução, situações de *deadlock* por motivos estruturais presentes em um modelo em WF-net que respeita a propriedade *Relaxed Soundness* (não *sound*). Para isto, este procedimento determina as possíveis alternativas de execução que desviem das situações de *deadlock* presentes no modelo.

- construir, simular e validar o mecanismo de inferência especializado do tipo *token player* definido neste trabalho. Para isto, a ferramenta de simulação CPN Tools foi escolhida por apresentar um alto grau de flexibilidade na elaboração de modelos de simulação de forma a permitir a configuração de disparos incertos na presença de informações imprecisas.

1.5 Contribuições

A realização desta pesquisa contribui com a área de processos de negócios no que diz respeito ao conceito de flexibilidade nos modelos de processos que atendam completamente ao critério de correção *Soundness* ou parcialmente (*Relaxed Soundness*). A principal contribuição que se espera deste trabalho é a de propor duas novas técnicas de modelagem, nomeadas *WorkFlow net* possibilística e *WorkFlow net* interorganizacional possibilística (IOWF-net possibilística), juntamente com um novo mecanismo de inferência especializado, do tipo *token player*, de forma a suportar possíveis falhas de comunicação, inconsistências/desvios/exceções e decisões humanas não completamente previstas e permitir contornar possíveis problemas estruturais, os quais, em alguns casos, não possibilitam finalizar corretamente a execução do modelo do processo (situações de *deadlock* por exemplo).

Levando em consideração os problemas decorrentes do uso dos WfMS's reportado no trabalho realizado por Heinl et al. (1999) e a taxonomia referente às características concretas que aprimoram a flexibilidade nos WfMS's proposta por (SCHONENBERG et al., 2008b), neste trabalho, durante a execução de um modelo de processo modelado por uma WF-net possibilística ou por uma IOWF-net possibilística, a ordem dos passos previamente definidos podem ser alteradas, a existência de alguns desvios em relação às etapas modeladas são permitidas e possíveis erros são contornados. Assim sendo, as 4 propostas de contribuição descritas a seguir são obtidas:

1. implementar uma solução para a paralelização parcial de um modelo de processo estritamente sequencial de forma que algumas atividades se iniciem antes da conclusão de suas anteriores sem que os roteiros do modelo sejam redefinidos ou que situações de inconsistências de estados aconteçam durante a execução do mesmo;

2. implementar uma solução para o cancelamento da execução de processos sem perder a decidibilidade das propriedades do modelo e com a finalização correta das atividades, ou seja, uma vez alcançado o estado final nenhuma atividade remanescente se encontrará em execução;

3. implementar uma solução para recuperar o estado do modelo do processo após situações de desvios, em particular, falhas de comunicação entre processos interorganizacionais;

4. implementar uma solução para detectar e evitar os estados de *deadlock* ocasionados, por motivos estruturais, durante a execução de um modelo de processo modelado por uma WF-net *Relaxed Soundness*;

1.6 Organização do Texto

O presente texto encontra-se estruturado em 7 capítulos, organizados da forma como se segue.

No Capítulo 2, a fundamentação teórica necessária para a compreensão desta pesquisa é apresentada. Na Seção 2.1 faz-se uma introdução sobre as definições básicas da rede de Petri bem como seus elementos, propriedades e extensões. Nas Seções 2.2 e 2.3 a *WorkFlow net* e a *WorkFlow net* interorganizacional são definidas juntamente com a propriedade *Soundness* e *Relaxed Soundness* aplicadas a cada uma delas. Por fim, a Seção 2.4 apresenta o conceito de sifão para a definição e detecção de situações de *deadlock* em um modelo em rede de Petri.

No Capítulo 3, os trabalhos relacionados com o objetivo da tese são apresentados. Nele, as principais contribuições existentes na literatura abordando o tema de "flexibilidade nos processos de negócios" são expostas. Além destas, também são expostas as principais abordagens propostas para contornar/evitar situações de *deadlock* e agregar a noção de cancelamento ao processo de negócio. A cada abordagem citada, as limitações existentes em cada uma são apresentadas.

No Capítulo 4, a WF-net possibilística proposta inicialmente no contexto do curso de Mestrado em Ciência da Computação pela aluna responsável pela presente tese é redefinida e aplicada em problemas específicos relacionados aos processos de negócio. Na Seção 4.1, a definição da WF-net possibilística é alterada com o objetivo de considerar uma ação para cada possível mudança no modelo do processo de acordo com a qualidade das informações recebidas. Na Seção 4.2, o modelo é aplicado aos processos de negócio estritamente sequenciais com o objetivo de realizar uma paralelização parcial na ordenação das atividades. Na Seção 4.3, a noção de cancelamento é introduzido no modelo de forma a considerar a possibilidade de cancelamento de atividades sem que a viabilidade de verificação das propriedades do modelo seja perdida.

No Capítulo 5, a *WorkFlow net* interorganizacional possibilística é proposta a fim de contornar possíveis falhas de comunicação e situações de *deadlock* presentes nos processos de negócio interorganizacionais. A definição da *WorkFlow net* interorganizacional possibilística é dada na Seção 5.1. Os problemas de comunicação tais como mensagens perdidas, atrasadas ou inesperadas entre os processos parceiros são considerandos na Seção 5.1. As possíveis situações de *deadlock* causadas, em sua grande maioria, por razões estruturais dos processos são consideradas na Seção 5.3.

No Capítulo 6, o mecanismo de inferência especializado do tipo *token player* definido no Capítulo 4 é configurado na ferramenta CPN Tools a fim de permitir a modelagem, isto é, a edição e execução dos modelos de processos usados para ilustrar as propostas descritas nos Capítulos 4 e 5. Os resultados obtidos a partir da execução dos modelos são comparados com os resultados obtidos quando a noção de disparo incertos é desconsiderada.

Finalmente, no Capítulo 7, são apresentadas as considerações finais, bem como as principais contribuições (em particular as produções bibliográficas que foram geradas e apresentadas durante o trabalho de tese) e sugestões de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo, a fundamentação teórica necessária para o entendimento do trabalho é apresentada. O capítulo é dividido em quatro seções. A Seção 2.1 aborda os principais conceitos de uma rede de Petri, assim como seus elementos, propriedades e extensões. As Seções 2.2 e 2.3 apresentam respectivamente a *WorkFlow net* e a *WorkFlow net* interorganizacional, as quais modelam processos de negócio. Por fim, a Seção 2.4 apresenta a ocorrência e a detecção de situações de *deadlock* estrutural em uma rede de Petri.

2.1 Rede de Petri

A teoria da rede de Petri originou-se na tese de Carl Adam Petri, apresentada em 1962 à Universidade de Darmstadt em (PETRI, 1962). Uma rede de Petri é uma ferramenta gráfica e matemática que se adapta a inúmeras aplicações em que as noções de eventos e evoluções simultâneas são importantes (CARDOSO; VALETTE, 1997). Ela é um tipo de grafo, bipartido e orientado, o qual permite a modelagem, análise, simulação e controle de Sistemas a Eventos Discretos (MURATA, 1989). Uma rede de Petri representa, portanto, processos em sistemas dinâmicos em que há relação de concorrência, paralelismo e sincronização de informações.

O formalismo matemático é um dos pontos mais importantes de uma rede de Petri. Por ser formal, é possível realizar uma análise precisa dos modelos para verificar propriedades estruturais e comportamentais. Além disso, permite a visualização gráfica dos processos e uma comunicação relativamente fácil entre as partes interessadas, sejam elas envolvidas ou não nas técnicas de programação. De acordo com Cardoso e Valette (1997) e Rillo (1988), as inúmeras vantagens de uma rede de Petri podem ser resumidas pelas seguintes considerações:

- o formalismo matemático inerente de uma rede de Petri permite uma melhor compreensão sobre o comportamento do processo;

- engloba especificação, modelagem, análise, avaliação de desempenho e de implementação em uma única ferramenta;
- pode-se descrever uma ordem parcial entre vários eventos;
- os estados, bem como os eventos, são representados explicitamente;
- a representação gráfica pode ser utilizada como documentação e os modelos podem ser simulados, de modo que não só a estrutura do processo, mas também o seu comportamento dinâmico podem ser observados;
- possui capacidade para modelagem hierárquica, com fundamentação matemática bem desenvolvida, que pode ser utilizada para análise;
- representa o processo em diferentes níveis de abstração e os processos individuais e inter-relacionados em sistemas distribuídos;
- descreve de maneira precisa e formal as sincronizações, possibilitando a segurança de funcionamento;
- diferentes propriedades de sistemas concorrentes, tais como conflito, concorrência, *deadlock*, sincronismo e exclusão mútua podem ser verificadas formalmente.

Segundo Murata (1989) e David e Alla (2010), os elementos básicos que permitem a definição da estrutura de um modelo em rede de Petri são:

- **Lugar**: representado por um círculo, pode ser interpretado como uma condição, um estado parcial ou de espera, algum procedimento, um conjunto de recursos, um estoque, etc. É considerado o elemento passivo do modelo;
- **Transição**: representada por barra ou retângulo e é associada a um evento que ocorre. É o elemento ativo do modelo;
- **Ficha**: representada por um ponto num lugar, é um indicador de que a condição associada ao lugar é verdadeira, como, por exemplo, um recurso disponível em um certo processo;
- **Arco**: representado por setas que ligam as transições aos lugares e os lugares às transições.

A Figura 1 ilustra um modelo de processo representado por uma rede de Petri composta pelos elementos básicos descritos acima.

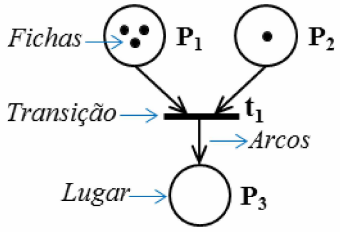


Figura 1 – Elementos básicos de um modelo de processo representado por uma rede de Petri.

Formalmente, uma rede de Petri é definida da seguinte forma (MURATA, 1989) (DIAZ, 2009) (DAVID; ALLA, 2010):

Definição 1 ***(Rede de Petri)*** *Uma rede de Petri é uma quádrupla*

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

onde:

- P *é um conjunto finito de lugares de dimensão* n*;*
- T *é um conjunto finito de transições de dimensão* m*;*
- $Pre : P \times T \to \mathbb{N}$ *define os arcos de entrada das transições;*
- $Post : P \times T \to \mathbb{N}$ *define os arcos de saída das transições.*

Um lugar p é chamado de lugar de entrada de uma transição t se, e somente se, existe um arco orientado de p para t. Um lugar p é chamado de lugar de saída de uma transição t se, e somente se, existe um arco orientado de t para p. Tais conceitos são ilustrados através da Figura 1. Nela, os lugares P_1 e P_2 são lugares de entrada da transição t_1 e o lugar P_3 é um lugar de saída da transição t_1.

De um modo formal, Murata (1989) utiliza as notações $\bullet p$ e $p\bullet$ para denotar, respectivamente, o conjunto de transições que compartilham p como lugar de saída ou como lugar de entrada e $\bullet t$ e $t\bullet$ para denotar, respectivamente, o conjunto de lugares de entrada ou saída de uma transição t. Considerando a Figura 1, tem-se que: $\bullet P_1 = \emptyset$, $\bullet P_2 = \emptyset$, $\bullet P_3 = \{t_1\}$, $P_1\bullet = \{t_1\}$, $P_2\bullet = \{t_1\}$, $P_3\bullet = \emptyset$, $\bullet t_1 = \{P_1, P_2\}$ e $t_1\bullet = \{P_3\}$.

Um estado de uma rede de Petri, denominado marcação, é uma distribuição de fichas sobre os lugares. A definição de uma rede de Petri marcada é dada abaixo (MURATA, 1989) (DAVID; ALLA, 2010):

Definição 2 ***(Rede de Petri Marcada)*** *Uma rede de Petri marcada é uma dupla*

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

onde:

- ❏ *R é uma rede de Petri;*
- ❏ *M é a marcação dada pela aplicação* $M : P \rightarrow \mathbb{N}$.

$m(p)$ é o número de fichas (*tokens* em inglês) 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 ($M^T = [m(p_1), m(p_2), \ldots, m(p_n)]$ onde T é o transposto do vetor). Para o exemplo da Figura 1, a marcação inicial M_0 é dada por $M_0^T = [\ 3 \quad 1 \quad 0\ ]$. Observa-se que um lugar p pode conter, em um dado momento, zero ou mais fichas, as quais são representadas graficamente por pontos pretos (•).

Em uma rede de Petri, a ocorrência de um evento no processo é representada pelo disparo da transição ao qual o evento está associado. Uma condição necessária para o disparo de uma transição é que ela esteja habilitada, isto é, cada lugar de entrada da transição deve possuir ao menos uma ficha. Por exemplo, a transição t_1 da rede de Petri da Figura 2a não está habilitada. Em contrapartida, a transição t_1 da rede de Petri da Figura 2b está habilitada, pois há uma ficha em cada lugar de entrada desta transição.

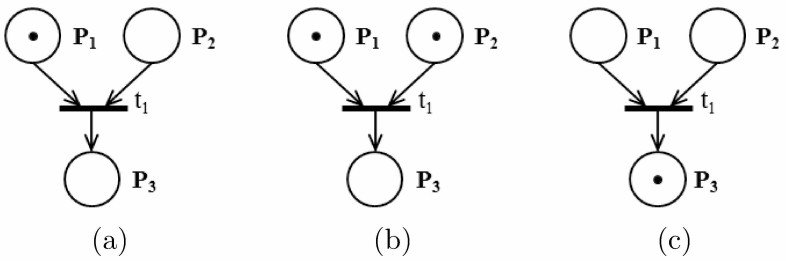


Figura 2 – Exemplos de sensibilização e disparo de transição em uma rede de Petri.

O disparo de uma transição, segundo (DAVID; ALLA, 2010), consiste de: (1) retirar as fichas dos lugares de entrada e (2) depositar fichas em cada lugar de saída. Retirar as fichas de um lugar indica que esta condição não é mais verdadeira após a ocorrência do evento. Por outro lado, ao depositar uma ficha em um lugar indica que esta atividade está sendo executada após a ocorrência do evento. Deste modo, o disparo de uma transição modifica a marcação e representa, no modelo, a mudança de estado ocorrida no processo devido à ocorrência do evento associado a ela.

As Figuras 2b e 2c mostram um exemplo de disparo de transição. A marcação inicial é representada pela Figura 2b e a marcação final, obtida após o disparo da transição t_1, é representada pela Figura 2c. Neste caso, t_1 remove uma ficha de cada lugar de entrada (P_1 e P_2) e deposita uma ficha em cada lugar de saída (P_3).

Formalmente, a evolução dinâmica de uma rede de Petri, isto é, a execução da rede, é dada pelas seguintes definições (MURATA, 1989) (DAVID; ALLA, 2010):

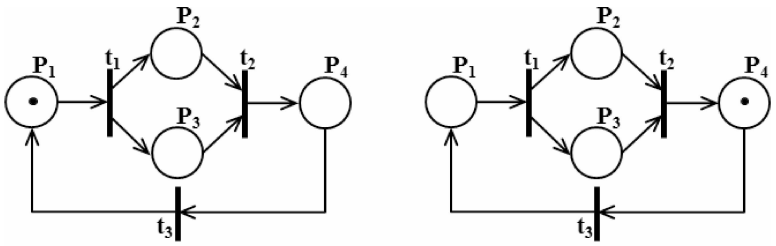
Definição 3 ***(Transição habilitada)*** *Uma transição t está habilitada (ou sensibilizada) se, e somente se:*

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

Definição 4 ***(Disparo de uma transição)*** *Se t está habilitada por uma marcação M, uma nova marcação M' é obtida através do disparo de t de maneira que:*

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

Uma sequência de disparo s pode ser representada por um vetor chamado *vetor característico* da sequência s. Sua dimensão é igual ao número de transições da rede de Petri e cada componente representa o número de ocorrências da transição t na sequência de disparo s. Considerando a marcação inicial $M_0^T = [\ 1 \quad 0 \quad 0 \quad 0\ ]$ na rede da Figura 3a, o disparo da sequência s = t_1t_2 leva à marcação $M_1^T = [\ 0 \quad 0 \quad 0 \quad 1\ ]$ representada na Figura 3b com o *vetor característico* $V_s^T = [\ 1 \quad 1 \quad 0\ ]$. Diferentes sequências de disparo podem ter um único vetor característico, pois este não considera a ordem de disparo das transições.



(a) (b)

Figura 3 – Sequência de disparo de transições.

Considerando as definições de marcação e sequência de disparo, Murata (1989) apresenta as seguintes notações:

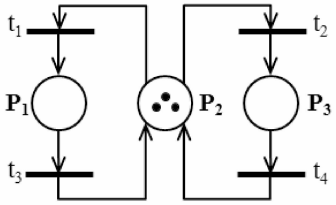
1. $M_1 \xrightarrow{t} M_2$: a transição t está habilitada na marcação M_1 e o seu disparo resulta na marcação M_2;

2. $M_1 \rightarrow M_2$: há alguma transição t tal que $M_1 \xrightarrow{t} M_2$;

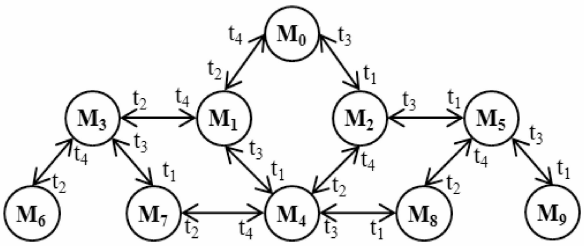
3. $M_1 \xrightarrow{s} M_n$: a sequência de disparo $s = t_1t_2 \ldots t_{n-1}$ leva da marcação M_1 para a marcação M_n, isto é, $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \cdots \xrightarrow{t_{n-1}} M_n$.

O conjunto das marcações acessíveis $A(N, M_0)$ de uma rede de Petri marcada N é o conjunto das marcações M que podem ser atingidas a partir da marcação inicial M_0 através de uma sequência de disparo s ($A(N, M_0) = \{M_0, \exists s \,|\, M \xrightarrow{s} M_0\}$). Se este conjunto é finito, o mesmo pode ser representado sob a forma de um grafo $G_A = (N, M_0)$ orientado e rotulado. Os vértices representam as marcações alcançáveis do conjunto $A(N, M_0)$ e são rotulados com a indicação da marcação correspondente. Uma aresta (seta) representa o disparo de uma transição, ou seja, dois vértices M' e M'' são ligados se existe uma transição t habilitada que permite passar de uma marcação M' a uma outra M'' ($M' \xrightarrow{t} M''$). Cada aresta é rotulada com o nome da transição t correspondente.

A Figura 4b representa o grafo das marcações acessíveis para a rede de Petri da Figura 4a considerando a marcação inicial $M_0^T = [\ 0 \quad 3 \quad 0\ ]$. Tal grafo pode ser usado para verificar uma variedade de propriedades de uma rede de Petri como, por exemplo, se uma rede é livre de *deadlock* (MURATA, 1989), sendo fundamental para o estudo das propriedades da dinâmica dos processos.



(a) Modelo de um processo representado por uma rede de Petri



M_0^T = [0 3 0] M_5^T = [2 1 0]
M_1^T = [0 2 1] M_6^T = [0 0 3]
M_2^T = [1 2 0] M_7^T = [1 0 2]
M_3^T = [0 1 2] M_8^T = [2 0 1]
M_4^T = [1 1 1] M_9^T = [3 0 0]

(b) Grafo de Marcações Acessíveis

Figura 4 – Marcações Acessíveis.

Uma rede de Petri é representada algebricamente através da matriz de incidência I por meio da relação entre os lugares e as transições. Levando-se em consideração que os elementos Pre e $Post$ de uma de rede de Petri podem ser representados na forma de matrizes, a matriz de incidência I, com dimensão $n \times m$ (n é a quantidade de lugares e m a de transições), é formada pela subtração de $Post$ com Pre ($I = Post - Pre$). Um elemento igual a *zero* indica que não existe arco entre os correspondentes *lugar* e *transição*. A notação matricial da rede de Petri da Figura 3a é dada por:

$$Pre = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix} \qquad Post = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad I = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

Cada coluna pertencente à matriz de incidência I corresponde à modificação da marcação através do disparo da transição associada. Por exemplo, a primeira coluna da matriz de incidência I mostrada anteriormente indica que o disparo da transição t_1 consiste da remoção de uma ficha no lugar P_1 e da adição de uma ficha em cada um dos lugares P_2 e P_3. A partir disto, conclui-se que a matriz de incidência fornece o balanço das fichas na rede quando ocorre o disparo das transições.

A evolução da marcação de uma rede de Petri é dada pela seguinte equação:

$$M_f = M_0 + I \cdot V_s \tag{5}$$

onde M_f é uma marcação final[1] obtida através da adição de uma marcação inicial[2] M_0 ao produto da matriz de incidência I pelo *vetor característico* V_s, o qual representa uma sequência de disparo s.

A equação (5), chamada *Equação Fundamental* da rede de Petri, descreve a evolução dos estados de uma rede de Petri bem como possibilita a análise de suas propriedades comportamentais e estruturais. Para exemplificar o funcionamento de tal equação, a rede de Petri da Figura 3a e a sequência de disparo $s = t_1t_2$ serão consideradas. A marcação final M_f obtida é apresentada abaixo (a qual pode ser observada na Figura 3b):

$$M_f = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Por meio de operações matemáticas, diferentes vetores V_s podem ser derivados, mas nem sempre estes correspondem a uma sequência de disparo válida.

2.1.1 Propriedades de uma Rede de Petri

Uma rede de Petri não se resume apenas a uma ferramenta que permite a modelagem de problemas que tenham atividades concorrentes; ela é utilizada também para descrever e analisar um processo (DAVID; ALLA, 2010). Em (MURATA, 1989), as propriedades de alcançabilidade, limitabilidade, vivacidade e reiniciabilidade relativas a uma rede de Petri marcada são definidas e reagrupadas sob o nome genérico de "*boas propriedades*".

A definição das boas propriedades de uma rede de Petri é dada a seguir (MURATA, 1989):

Definição 5 ***(Alcançabilidade)*** *Uma marcação M' é dita alcançável pela marcação M_0, se existe uma sequência finita de disparo de transições ($s = t_1t_2t_3 \ldots t_{n-1}$) que, aplicada à rede a partir de M_0, possibilita alcançar a marcação M' ($M_0 \xrightarrow{s} M'$).*

A propriedade de alcançabilidade[3] é a base fundamental para o estudo das propriedades da dinâmica de qualquer processo. Essa propriedade garante que certos estados ou operações sempre serão alcançados ou realizados. A verificação da alcançabilidade para um modelo, sob o ponto de vista de um processo de negócio, é um meio de verificar a possibilidade de execução de um caso (cenário).

[1] A marcação final representa o estado final do processo correspondente
[2] A marcação inicial representa o estado inicial do processo correspondente
[3] tradução para *reachability*

Definição 6 ***(Limitabilidade)*** *Uma rede de Petri marcada $N =< R, M >$ é limitada ou k-limitada se, e somente se, todos os seus lugares p são k-limitados ($\forall\, m' \in A(R, M_0)$, $m'(p) \leq k$).*

A propriedade de limitabilidade[4] garante que o número de fichas em cada lugar não excede um número finito k para qualquer marcação alcançável a partir da marcação inicial. Em especial, uma rede marcada N é chamada de binária se, e somente se, a rede for 1-limitada, ou seja, k é igual à 1.

Definição 7 ***(Vivacidade)*** *Uma rede de Petri marcada é viva se, e somente se, toda transição t pode ser habilitada a partir de qualquer marcação M' do grafo das marcações acessíveis ($\forall t \in T$, $\exists m_1$, $m_2 \in A(R, M) \,|\, m_1 \xrightarrow{t} m_2$).*

A propriedade de vivacidade[5] garante que o processo seja livre de bloqueios (*deadlock free*).

Definição 8 ***(Reiniciabilidade)*** *Uma rede de Petri é reiniciável se, a partir de qualquer marcação acessível m' de $G_A(R, M_0)$ é possível encontrar uma sequência de disparo s que leve a rede de Petri de volta à marcação M_0 ($\forall m' \in A(R, M_0)$, $\exists s \,|\, m' \xrightarrow{s} M_0$).*

A propriedade de reiniciabilidade[6] é de vital importância quando um processo modelado tem seu funcionamento repetitivo. Uma rede de Petri é repetitiva se existe uma sequência de disparos s válida onde $s(t) \neq 0$ para todas as transições $t \in T$ indicando que todas as ações são executadas um número de vezes infinito. Se a sequência existir, mas apenas algumas ações forem disparadas ilimitadamente, a rede é dita parcialmente repetitiva.

2.1.2 Propriedades Estruturais

Nesta subseção são consideradas as propriedades derivadas diretamente da estrutura de uma rede de Petri. Tais propriedades são definidas por meio dos componentes conservativos de lugar e componentes repetitivos estacionários. A partir destes elementos estruturais, pode-se utilizar também a informação sobre a marcação, definindo-se, assim, os invariantes lineares de lugar e de transição, que permitem informações adicionais sobre o comportamento da dinâmica de uma rede de Petri.

---

[4] tradução para *boundedness*
[5] tradução para *liveness*
[6] tradução para *reversibility*

2.1.2.1 Componentes Conservativos

Os componentes conservativos de uma rede de Petri são representados por seus invariantes de lugar, ou seja, são conjuntos de lugares da rede nos quais a soma das fichas é constante durante toda a evolução das marcações. A equação que permite determinar a evolução das marcações é a equação fundamental (Equação 5). Para se obter uma soma ponderada de marcações, multiplica-se inicialmente a equação fundamental por um vetor f^T:

$$f^T \cdot M_f = f^T \cdot M_0 + f^T \cdot I \cdot V_s \tag{6}$$

Para que a equação fundamental da rede de Petri se torne independente das sequências de disparo s, faz-se necessário tornar o produto $f^T \cdot I$ nulo. Se f é a solução do produto, ou seja:

$$f^T \cdot I = 0 \text{ com } f > 0 \tag{7}$$

as seguintes definições podem ser obtidas (PETERSON, 1983) (DIAZ, 2009):

Definição 9 ***(Componente Conservativo)*** *Um componente conservativo é um conjunto de lugares $p_i \in P$ correspondentes aos elementos não nulos f_i do vetor coluna f.*

Definição 10 ***(Rede de Petri Conservativa)*** *Uma rede de Petri é dita conservativa se todos os lugares da rede pertencem a um componente conservativo.*

Definição 11 ***(Invariante de Lugar)*** *Um invariante de lugar é dado pela função linear $f^T \cdot M_f = f^T \cdot M_0 \ \forall M_f \in A(R, M_0)$ cujo valor é uma constante que depende apenas da marcação inicial da rede. Ele corresponde a uma restrição sobre os estados e as atividades do processo que será sempre verificada, quaisquer que sejam suas evoluções.*

2.1.2.2 Componentes Repetitivos

Os componentes repetitivos são representados pelos seus invariantes de transição, isto é, são conjuntos de transições da rede que, ao serem disparados em determinada sequência, retornam à marcação inicial. O componente repetitivo não depende da marcação inicial, ao contrário do invariante de transição correspondente de quem sua existência depende.

Para encontrar o conjunto de transições que, uma vez disparadas a partir de uma marcação da rede, faz retornar a esta mesma marcação, utiliza-se novamente a equação fundamental (Equação 5). E fácil verificar que, para se obter $M_f = M_0$, a sequência s deve ser tal que o vetor V_s verifique:

$$I \cdot V_s = 0 \tag{8}$$

A partir da equação acima, pode-se definir que (PETERSON, 1983) (DIAZ, 2009):

Definição 12 ***(Componente Repetitivo)*** *Toda solução V_s da Equação (8) é chamada componente repetitivo estacionário da rede de Petri.*

Definição 13 ***(Rede de Petri Repetitiva)*** *Uma rede de Petri é dita repetitiva se todas as transições $t \in T$ pertencem a um componente repetitivo estacionário.*

Definição 14 ***(Invariante de Transições)*** *Se V_s é o vetor característico de uma sequência s efetivamente disparada a partir de uma marcação acessível, então esta sequência s, a qual pode ser repetida indefinidamente, é um invariante de transições.*

2.1.3 "Jogador" de Rede de Petri - *Token Player*

A implementação de uma rede de Petri em uma grande variedade de domínios se dá através do "Jogador" de rede de Petri, também conhecido como *Token Player* (VALETTE; ATABAKHCHE, 1987). Tal "jogador" é um motor de inferência especializado responsável por executar a rede de Petri, isto é, deslocar as fichas de modo a respeitar as regras de disparo das transições (CARDOSO; VALETTE, 1997).

Esta implementação consiste em "simular" ou "imitar" o comportamento de um processo representado por uma rede de Petri interpretada. A Figura 5 descreve a estrutura de um "jogador" para rede de Petri interpretada[7] definida por Valette e Atabakhche (1987). Este "jogador" deve ser capaz de sincronizar o disparo das transições com a ocorrência de um evento externo associado a elas (CARDOSO; VALETTE, 1997). Isto corresponde à recepção de mensagens de atualização do conhecimento ou, então, a requisições provenientes do processo. Além disto, a partir de uma marcação dada, todas as transições internas habilitadas devem ser disparadas antes de considerar o evento externo seguinte.

O "jogador" descrito na Figura 5 permite especificar dois possíveis comportamentos, os quais são:

1. o "jogador" verifica se existe alguma transição habilitada. Caso exista, confere se a interpretação associada a esta transição é válida; se sim, realiza o disparo. Enquanto existe uma transição habilitada que possa ser disparada, o "jogador" realizará o respectivo disparo, entretanto, caso não haja mais nenhuma, o "jogador" entrará num estado no qual nenhuma transição poderá ser disparada enquanto um evento externo não for percebido. Esse comportamento é representado pela parte esquerda da Figura 5;

2. o "jogador" é despertado por um evento externo (recepção de uma mensagem, por exemplo). O "jogador" procura a transição correspondente ao evento externo, dispara-a e, em seguida, executa o procedimento anterior (primeiro comportamento). Caso nenhuma transição relacionada ao evento externo possa ser disparada, entende-se que há algum problema na interação entre o mecanismo modelado e o seu ambiente, então um alarme é ativado. Este comportamento corresponde à parte da direita da Figura 5.

[7] Uma rede interpretada é uma rede não autônoma que introduz o tempo, a interação com o ambiente ou dados contidos na ficha (VALETTE; ATABAKHCHE, 1987)

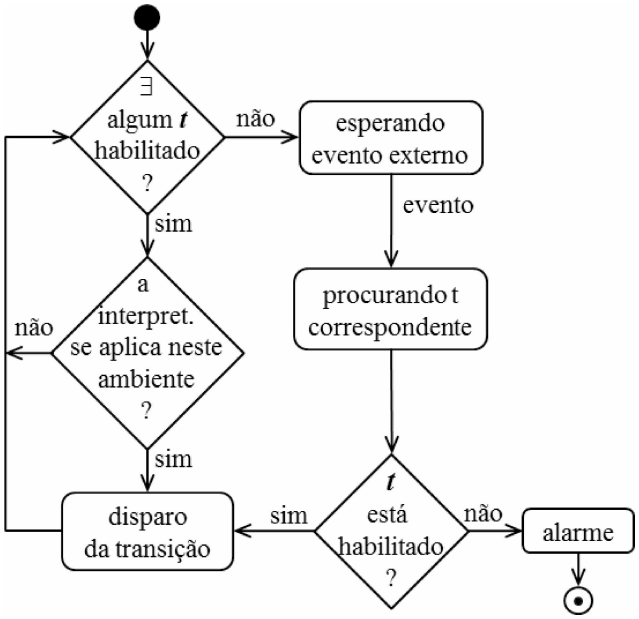


Figura 5 – Estrutura de um "Jogador" de rede de Petri (VALETTE; ATABAKHCHE, 1987).

2.1.4 Tipos de Rede de Petri

Buscando tornar o poder de modelagem mais abrangente em relação às características dos processos modelados, extensões da rede de Petri têm sido propostas a partir do conceito fundamental descrito anteriormente. As extensões necessárias para o entendimento deste projeto de pesquisa serão apresentadas a seguir. São elas: rede de Petri com informações temporais, rede Predicado-Transição, rede de Petri colorida, rede de Petri a objetos e rede de Petri possibilística.

2.1.4.1 Rede de Petri com Informações Temporais

Uma rede de Petri clássica não permite a modelagem do tempo, tais como as datas de ocorrência de início e fim das operações, de forma explícita. Entretanto, elas podem ser estendidas para considerar o tempo, dado que informações temporais são úteis para sequenciar eventos, comparar suas durações bem como determinar o intervalo existente entre elas. Desta forma, os casos onde a avaliação do tempo é um critério de fundamental importância podem ser representados e analisados. Vários trabalhos foram realizados no sentido de utilizar explicitamente o tempo como parâmetro contínuo e quantificável em uma rede de Petri.

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

- *tempo de sensibilização*: determina o tempo que deve transcorrer desde o momento da habilitação de uma transição até o momento do seu disparo. O disparo ocorre de forma atômica, ou seja, disparo instantâneo, indivisível;

- *tempo de disparo*: o disparo de uma transição ocorre em três fases: tempo que transcorre para a retirada da ficha, disparo atômico e tempo que transcorre para a colocação da ficha.

Outro aspecto importante é quanto a efetivação do disparo de uma transição habilitada em relação ao tempo. Para alguns processos, 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 ser em uma determinada data também.

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 propriedades da restrição (CERONE; MAGGIOLO-SCHETTINI, 1999). Independente desses aspectos, podem-se agrupar as extensões temporais da rede de Petri em quatro grandes categorias:

- Rede de Petri Temporal (*Time Petri Net*): a restrição temporal é um intervalo de tempo associado a cada transição. Inicialmente usada na descrição de protocolos de comunicação, seu campo de aplicação tem se ampliado para áreas tais como: manufatura, processos em tempo real, validação e verificação de processos(BERTHOMIEU; MENASCHE, 1983) (CERONE; MAGGIOLO-SCHETTINI, 1999) (BOWDEN, 2000) (BÉRARD et al., 2013).

- Rede de Petri Temporizada (*Timed Petri Net*): permite que transições ou lugares retenham marcas durante um intervalo de tempo. Comumente usada na análise de processos de manufatura (LAHAYE; KOMENDA; BOIMOND, 2014) (DOTOLI et al., 2014) (ZUBEREK, 1991).

- Fichas Temporizadas: a restrição temporal é associada à ficha ao se propagar na rede. São largamente utilizadas em problemas de produção (CELKO, 1982) (TüYSüZ; KAHRAMAN, 2010).

- Modelos Probabilísticos: são as chamadas redes de Petri Estocástica usadas para estimar o desempenho de processos de grande porte. Probabilidades são então associadas com uma das opções: transições, lugares ou fichas (MURATA, 1989).

2.1.4.2 Rede Predicado-Transição

A rede Predicado-Transição, proposta por (GENRICH, 1987), é fundamentada no formalismo matemático da lógica de predicados (CANTÚ, 1990). Nela, variáveis são associadas aos arcos da rede de modo que as transições descrevem operações da lógica de predicados sobre essas variáveis incorporando, assim, o conceito de individualidade na marcação de uma rede de Petri (CARDOSO; VALETTE, 1997). Os lugares em uma rede Predicado-Transição são chamados de predicados e as fichas que neles se encontram representam as condições válidas do predicado. A definição formal dada por Genrich (1987) é descrita a seguir:

Definição 15 ***(Rede Predicado-Transição marcada)*** *Uma rede Predicado-Transição marcada é uma sêxtupla*

$$N_{PT} =< R, V_f, A_V, A_{tc}, A_{ta}, M_0 > \tag{9}$$

onde:

- R *é uma rede de Petri;*
- V_f *é um conjunto de variáveis formais;*
- A_V *associa a cada arco um vetor de variáveis formais de* V_f*;*
- $A_{tc} : T \rightarrow L_c(V_f)$ *é uma aplicação que associa, a cada transição, uma condição de disparo sob a forma de um predicado utilizando as variáveis formais pertencentes aos vetores* A_V *associadas aos arcos de entrada da transição;*
- $A_{ta} : T \rightarrow L_a(V_f)$ *é uma aplicação que associa, a cada transição, uma ação de disparo sob a forma de uma sequência de afetações de valores às variáveis formais pertencentes aos vetores* A_V *associados aos arcos de saída da transição em função dos valores das variáveis pertencentes aos vetores* A_V *associados aos arcos de entrada da transição;*
- M_0 *é a marcação inicial, associando a cada lugar* $p \in P$ *um vetor de variáveis cujo valor deve ser definido para o instante inicial.*

L_c e L_a são linguagens da lógica de predicados mais a classe de expressões algébricas simples para denotar combinações lineares. Estas linguagens expressam, respectivamente, as condições e ações associadas às transições.

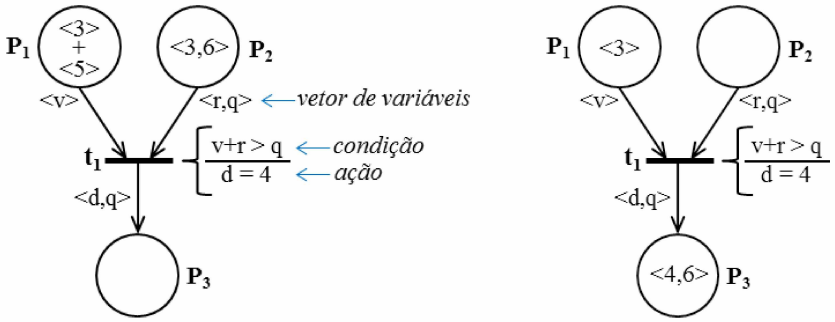
As variáveis associadas às fichas podem permitir ou não o disparo de uma transição (CARDOSO; VALETTE, 1997). Para isto, condições suplementares de disparo, as quais são escritas como fórmulas lógicas utilizando variáveis, são atribuídas a cada transição (VILLANI, 2004). Assim, uma transição t_i só pode ser disparada quando a condição

associada a ela é satisfeita. De modo similar, a ação associada à transição t_i define os valores das variáveis que serão associadas aos vetores dos arcos de saída. Se nenhuma ação é definida, os valores das variáveis não são alterados (TOMIYAMA, 2007).

Um exemplo de rede Predicado-Transição é representado na Figura 6a. Para esta rede são definidos:

- o conjunto de variáveis formais $V_f = \{r, q, v, d\}$;
- os vetores de variáveis associados aos arcos: $A_V(p_1, t_1) =< v >$, $A_V(p_2, t_1) =< r, q >$, $A_V(t_1, p_3) =< d, q >$;
- a condição $A_{tc}(t_1) = L_c(v, r, q) = (v + r) > q$;
- a ação $A_{ta}(t_1) = L_a(d) = 4$
- a marcação inicial M_0:

$$M_0 = \begin{bmatrix} < 3 > + < 5 > \\ < 3, 6 > \\ 0 \end{bmatrix}$$



(a) Antes do disparo (b) Após o disparo

Figura 6 – Exemplo de disparo em uma rede Predicado-Transição.

Na Figura 6a, pode se observar que a transição t_1 está habilitada para o disparo apenas com a marcação $< 5 >$ para p_1 e $< 3, 6 >$ para p_2. A outra possibilidade seria com a marcação $< 3 >$ para p_1 e $< 3, 6 >$ para p_2; entretanto a condição $A_{tc}(t_1)$ não é válida para tal marcação. Desta forma, após o disparo de t_1, utilizando a marcação $< 5 >$ para p_1 e $< 3, 6 >$ para p_2; a ação $A_{ta}(t_1)$ é executada obtendo, assim, a marcação $< 4, 6 >$ para p_3, como mostrado na Figura 6b.

2.1.4.3 Rede de Petri Colorida

Para modelar processos constituídos de muitos componentes com funcionalidades similares que interagem entre si, um modelo em rede de Petri pode exibir grande número de sub-redes idênticas. Isso se deve ao fato de que para diferenciar dois componentes

similares mas distintos, é preciso especificar uma sub-rede para cada caso, com estruturas idênticas (SILVA, 2014). Além disto, as fichas em uma rede de Petri geralmente representam objetos ou recursos em um processo modelado, os quais podem ter atributos distintos que não podem ser representados por uma simples ficha (AALST, 1992). A fim de resolver este problema, Jensen (1981) definiu o que hoje se conhece por rede de Petri Colorida, do inglês *Colored Petri Net (CPN).*

Uma CPN combina a teoria da rede de Petri com a linguagem de programação funcional *Standard ML* (MILNER; TOFTE; MACQUEEN, 1997) para obter uma linguagem de modelagem gráfica para processos concorrentes. A base formal para modelar concorrência e sincronização é fornecida pela rede de Petri e as primitivas para modelar a manipulação de dados e criação de modelos compactos e parametrizáveis é fornecida pela linguagem de programação funcional.

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

Definição 16 ***(Rede de Petri Colorida)*** *Uma rede de Petri Colorida associada a uma marcação inicial é uma sétupla*

$$N_{CPN} =< R, N, C_{or}, C_{sc}, G, E, M_0 > \tag{10}$$

onde:

- R *é uma rede de Petri;*
- C_{or} *é um conjunto de tipos não-vazios chamados cores;*
- $C_{sc} : P \rightarrow C_{or}$ *é uma função de cor que associa a cada lugar um sub-conjunto de* C_{or} *(as cores possíveis para este lugar);*
- $G : T \rightarrow exp$ *é uma função de guarda que associa a cada transição uma expressão de tipo booleano tal que* $\forall t \in T | Type(G(t)) = Boolean \wedge Type(Var(G(t))) \subseteq C_{or}$*;*
- $E : (Pre \cup Post) \rightarrow \{expressões\}$ *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:* $\forall a \in (Pre \cup Post) | Type(E(a)) = C_{or} \wedge Type(Var(E(a))) \subseteq C_{or}$*;*
- 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.*

Numa CPN, um lugar só pode comportar fichas cujos valores respeitem a cor (domínio) associada ao lugar. Tais fichas são representadas 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 e servem para manipular a(s) informação(ões) contida(s) nas fichas.

As transições determinam a dinâmica da CPN e podem apresentar "expressões de guarda" associadas a elas. Estas, por sua vez, são expressões booleanas que indicam os tipos de fichas que possibilitam ativar uma transição, restringindo assim o disparo das transições.

Um exemplo de uma CPN é representado na Figura 7. Neste exemplo, existem 4 lugares representados por P_1, P_2, P_3 e P_4 que podem conter cores do tipo *int* (números inteiros). Nestes lugares, apenas o lugar P_1 possui fichas associadas. Nele 5 fichas, cada uma com um valor inteiro diferente, foram definidas ($1'2$, por exemplo, representa uma ficha de valor 2). Cada uma das transições apresenta uma função de guarda, ou seja, a transição t_1 seleciona apenas valores pares e a transição t_2 apenas valores ímpares. Além disto, os cinco arcos presentes no modelo contêm expressões sobre eles. Nos dois arcos que saem de P_1 e no arco que incide em P_4, a expressão associada a cada um impõe que apenas valores inteiros podem percorrer esse arco (considere-se que o tipo da variável x foi definido como *int*). Os outros dois arcos apresentam declarações com códigos de estrutura de controle em linguagem ML. O arco que incide em P_2 permite passar por ele apenas valores que não são múltiplos de 3 (condição $x\,mod\,3 <> 0$). De maneira similar, o arco que incide em P_3 permite percorrer por ele somente valores que são múltiplos de 3 (condição $x\,mod\,3 = 0$).

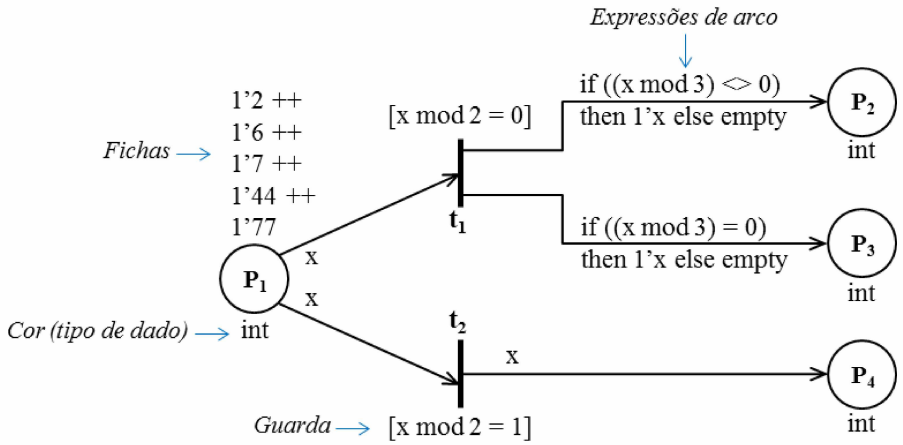


Figura 7 – Modelo de um processo representado por uma rede de Petri Colorida.

Em uma CPN as funções de guarda e inscrições em arcos estão encarregados de impor condições para que uma ficha possa deslocar-se ou não de um lugar para outro na rede. Para compreender a regra de disparo em uma CPN, é essencial, primeiramente,

compreender os conceitos de variável de transição, ligação[8] e elemento de ligação. As variáveis de transição são as expressões associadas aos arcos que incidem nas transições. Por exemplo, na Figura 7, a transição t_1 tem x como sua variável de transição. Uma ligação é a associação das variáveis de transição de um dado arco a um valor possível de cores. Por exemplo, para a mesma variável de transição x, são possíveis as ligações $l_1 =< x = 2 >$, $l_2 =< x = 6 >$, $l_3 =< x = 7 >$, $l_4 =< x = 44 >$ e $l_5 =< x = 77 >$. Um elemento de ligação é um par (t, b) onde t é uma transição e b é uma ligação para t. Como ilustração, considerando os exemplos anteriores, temos $el_1 = (t_1, l_1)$, $el_2 = (t_1, l_2)$, $el_3 = (t_1, l_3)$, $el_4 = (t_1, l_4)$ e $el_5 = (t_1, l_5)$ como elementos de ligação.

Uma transição em uma CPN é dita estar habilitada se possui em seus lugares um número de fichas suficiente para satisfazer as inscrições dos arcos (existir um elemento de ligação) e se sua guarda for satisfeita. Com isso, são retiradas fichas dos lugares de entrada e são adicionadas fichas aos lugares de saída de acordo com as expressões (avaliação das inscrições) dos arcos que ligam uma transição aos seus lugares de entrada e de saída. Considerando isto, a transição t_1, por exemplo, é habilitada apenas pelos elementos de ligação l_1, l_2 e l_4, dado que os outros não satisfazem a expressão de guarda $x\,mod\,2 = 0$. A marcação final obtida a partir da execução do modelo apresentado na Figura 7 é mostrada na Figura 8

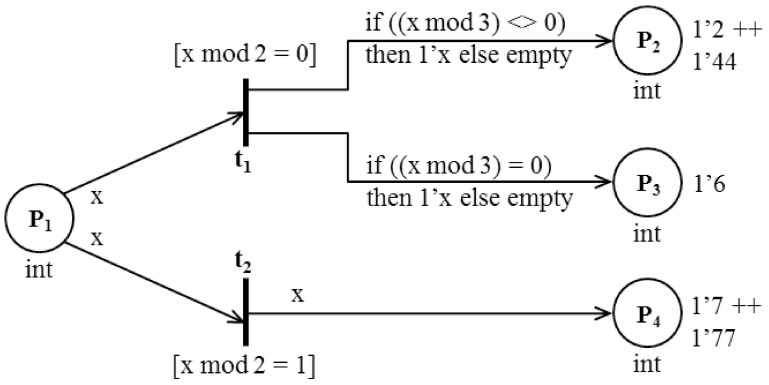


Figura 8 – Marcação obtida a partir do disparo de t_1 e t_2 considerando a marcação apresentada na Figura 7.

A aplicação em modelagem, simulação e análise de um processo descrito em CPN depende fortemente da existência de ferramentas computacionais que ofereçam suporte à criação e manipulação de tais modelos. E, por isso a ferramenta CPN Tools foi desenvolvida para edição, simulação e para análise de um modelo em CPN (JENSEN; KRISTENSEN, 2009). Essa ferramenta permite que os usuarios trabalhem diretamente com a representação gráfica da rede.

[8] Tradução para *binding*

2.1.4.3.1 Ferramenta CPN Tools

CPN Tools (JENSEN; KRISTENSEN; WELLS, 2007) é uma ferramenta para edição, simulação e análise cronometrada e não cronometrada de modelos em CPN. A ferramenta resulta do projeto de pesquisa CPN 2000, da Universidade de Aarhus em Dinamarca, patrocinado pelo Danish National Centre for IT Research, Universidade George Mason, Hewlett-Packard, Nokia e Microsoft. O projeto foi coordenado pela Universidadede Aarhus de 2000 até 2010, mas, a partir de 2011, passou a ser responsabilidade da Universidade de Tecnologia de Eindhoven, na Holanda.

Ao usar a ferramenta CPN Tools é possível, por meio da análise do espaço de estados (em inglês, *state space*), verificar propriedades do modelo e, através da simulação, investigar o comportamento e analisar o desempenho do processo modelado (JENSEN; KRISTENSEN; WELLS, 2007). Além disto, ela fornece condições para modelar processos distribuídos, considerando os vários tipos de processos envolvidos.

Os modelos de processos distribuídos em CPN podem ser estruturados no CPN Tools em módulos. Baseado em um mecanismo de estruturação hierárquica, a ideia básica é combinar modelos de partes dos processos em CPN de forma a permitir a construção de um modelo global (JENSEN; KRISTENSEN, 2009). Uma estruturação bem definida, a qual permite a análise formal do modelo.

Um dos elementos para a estruturação de um modelo em CPN Hierárquica é o lugar de fusão (em inglês, *fusion place*) (JENSEN; KRISTENSEN, 2009). O objetivo destes lugares é permitir a especificação de um conjunto de lugares como sendo idênticos, ou seja, todos estes lugares representam um único lugar, embora possam estar desenhados como lugares individuais. Isto significa que, quando uma ficha é adicionada/removida de um destes lugares, é como se esta ficha idêntica está sendo adicionada/removida de todos os outros lugares. Na ferramenta CPN Tools, ao definir um lugar de fusão, uma etiqueta é utilizada para definir o nome do conjunto de fusão ao qual ele pertence. A Figura 9 mostra um lugar de fusão com a etiqueta $Fusion1$ adicionada a ele.

Figura 9 – Exemplo de um lugar de fusão com etiqueta.

Quando todos os lugares de um mesmo conjunto de fusão pertencem à mesma parte ou página (no contexto de CPN Tools) da CPN, e têm apenas uma instância, os lugares de fusão são nada mais que um mecanismo para evitar cruzamentos de arcos (JESKE; JULIA; VALETTE, 2009). Dessa forma, é possível simplificar a representação gráfica da rede sem mudar o seu significado.

Em CPN Tools, cada elemento da CPN possui atributos descritos na linguagem CPN ML. Os atributos dos respectivos elementos de um modelo serão apresentados a seguir.

Inscrições nos Lugares

Existem três tipos de inscrições, duas opcionais e somente uma obrigatória, que podem ser associadas a um lugar como mostra a Figura 10:

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

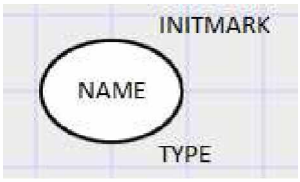


Figura 10 – Atributos de um lugar de uma CPN.

A inscrição da marcação de um lugar tem a seguinte sintaxe:

$$< quantidade\,de\,fichas >' < valor\,da\,ficha >$$

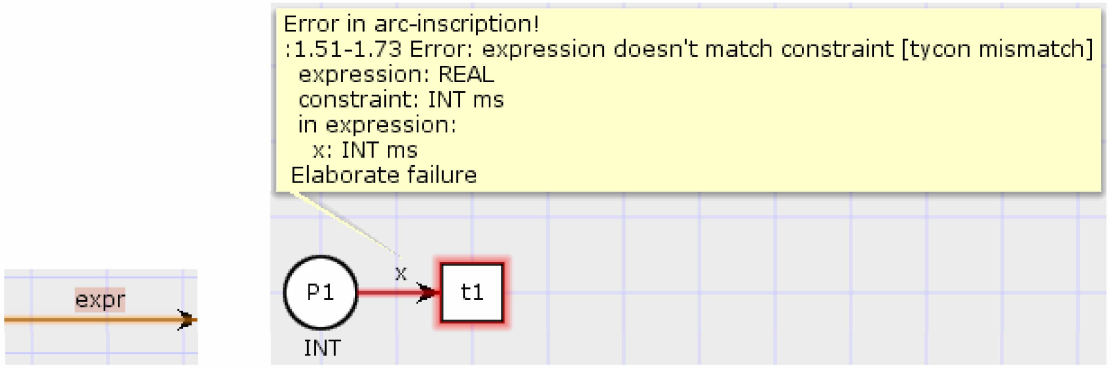
Quando se tem mais de uma ficha com valores distintos, o operador $++$ é responsável por concatená-las. Isto pode ser observado na Figura 12 através, por exemplo, do lugar P_1. Tal lugar possui, como marcação inicial, 6 fichas do tipo $STRING$ (sequência de caracteres) sendo uma ficha com o valor "X" e cinco fichas com o valor "Z".

Uma ficha pode ser temporizada. Para isto um retardo de tempo, indicado pela expressão $@+ < TEMPO >$, é associado ao valor da ficha. Considerando a Figura 13, tanto o lugar $P1$ quanto o lugar $P2$ possuem fichas temporizadas. Por exemplo, o lugar $P1$ contém uma ficha com o valor "A" disponível para ser utilizada, de fato, apenas no tempo corrente igual à 80.

Inscrições nos Arcos

Os arcos têm apenas uma única inscrição - a inscrição de arco. Antes de inserí-la no arco, o texto padrão é *expr* como indicado na Figura 11a. Tal inscrição de arco é uma expressão CPN ML. Esta expressão tem que coincidir com o conjunto de cores associado ao lugar ligado ao arco, caso contrário, o CPN Tools emite uma mensagem de erro durante a verificação de sintaxe do arco, como pode ser visto na Figura 11b.

Existe uma diferença básica entre as inscrições dos arcos de entrada e as dos arcos de saída de uma transição. A expressão do arco de entrada de uma transição é constituída



(a) Texto padrão

(b) Mensagem de erro

Figura 11 – Expressão de Arco.

pela escolha da(s) ficha(s) do(s) lugar(es) de entrada da transição. Em contrapartida, a expressão do arco de saída de uma transição é um construtor para a criação de novas fichas. Este construtor, frequentemente, usa a(s) variável(is) da(s) inscrição(ões) do(s) arco(s) de entrada da transição. Na Figura 12, o construtor declarado no arco de saída da transição $t1$ utiliza as variáveis x e y, obtidas por intermédio das inscrições dos arcos de entrada, para produzir uma nova ficha em $P3$ através da operação de concatenação $x \wedge y$.

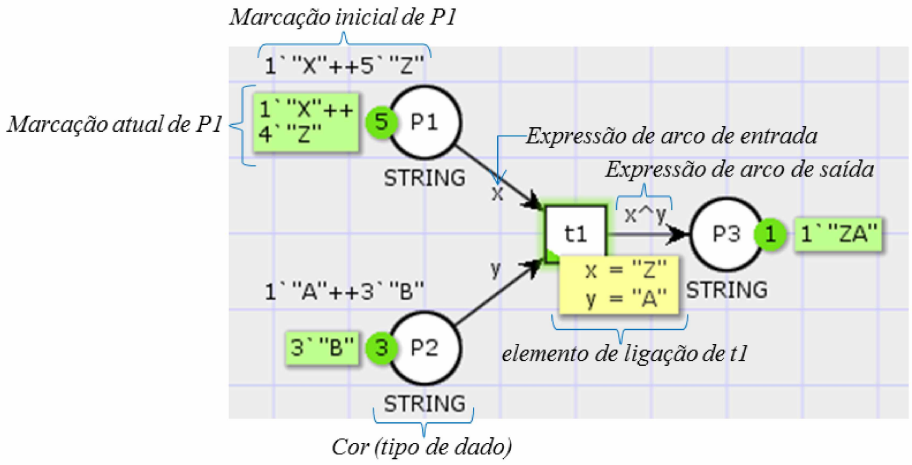


Figura 12 – Expressões nos Arcos de Entrada e de Saída de uma Transição.

As expressões dos arcos de saída das transições podem ser temporizadas. O retardo de tempo, indicado pela expressão @+ $< TEMPO >$ associada ao arco de saída, garante que a nova ficha produzida seja, de fato, disponibilizada nos lugares de saída apenas quando o tempo corrente alcance o tempo associado à ficha. Na Figura 13, a inscrição do arco $(t1, P1)$ contém um retardo de tempo no valor de 20 unidades. Assim, a cada 20 unidades de tempo, duas fichas são produzidas: uma ficha, cópia da ficha que se encontra em $P1$, é adicionada ao lugar $P2$ e outra ficha, de mesmo valor (A), é colocada em $P1$ com seu tempo de retardo acrescentado em 20 unidades de tempo.

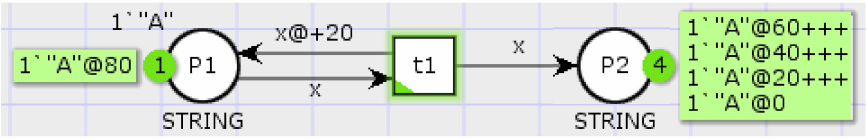


Figura 13 – Expressão temporizada em um arco de saída de uma transição.

Inscrições nas Transições

São cinco as inscrições, todas opcionais, que podem ser associadas às transições, como mostra a Figura 14:

- ❏ NAME - nome;
- ❏ P_NORMAL - prioridade de disparo;
- ❏ [] - guarda;
- ❏ @+ - retardo de tempo;
- ❏ $input(); output(); action();$ - segmento de código.

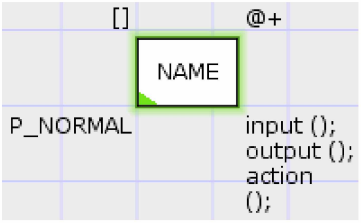


Figura 14 – Inscrições que podem ser associadas a uma transição.

Além do nome, cada transição pode ter também 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. A prioridade padrão associada a todas as transições em um modelo implementado em CPN Tools é a P_NORMAL. Se porventura deseja-se modificar a prioridade de disparo de uma transição, deve-se alterar para P_HIGH ou P_LOW ou definir novas prioridades através da definição de novas constantes inteiras.

Quando a prioridade de uma transição t é P_HIGH, significa que t será a primeira a ser disparada dentre todas as transições habilitadas disponíveis. No extremo oposto se encontra a prioridade P_LOW, ou seja, quando a prioridade de t é P_LOW o seu disparo é a último a ser realizado dentre todas as transições habilitadas.

A inscrição referente ao retardo de tempo tem que ser uma expressão com resultado positivo. Além disto, o retardo de tempo é sempre relativo ao tempo atual, isto é, se, por exemplo, o tempo atual é 20 e, o retardo de tempo associado à transição é $@ + 15$, então, o tempo associado às fichas colocadas nos lugares de saída da transição, quando do seu disparo, será igual $20 + 15$.

Quando uma transição não possui um tempo associado, o retardo de tempo relacionado ao disparo da mesma é zero. E, diferentemente do retardo de tempo associado a um dado arco, onde o retardo é associado somente às fichas relativas à este arco, quando associado a uma transição, todas as fichas produzidas nos lugares de saída desta transição terão o retardo de tempo associado a elas.

A inscrição de guarda é uma expressão booleana da linguagem CPN ML avaliada como verdadeira ou falsa. Ela pode ser composta por zero, uma ou várias expressões booleanas $[b-expr1, b-expr2, ..., b-exprn]$. Quando nenhuma expressão de guarda é associada a uma dada transição, o valor da inscrição é sempre verdadeiro. Em contrapartida, se uma expressão de guarda é associada a uma dada transição, o disparo da mesma só poderá ocorrer se a expressão for avaliada como verdadeira.

Se uma ação, expressa por um segmento de código, for definida para uma determinada transição, quando a mesma é disparada, esta ação é então executada. 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:

- padrão de entrada (*Input pattern*) – opcional;
- padrão de saída (*Output pattern*) – opcional;
- código (*Code action*) – obrigatório.

Um *Input pattern* é um conjunto de variáveis CPN precedidas pela palavra reservada *input*. Tais variáveis, usadas no *Code action*, não podem ter seus valores modificados. Quando o *Input pattern* não é declarado, nenhuma variável CPN pode ser usada no *Code action*.

Um *Output pattern* é um conjunto de variáveis CPN precedidas pela palavra reservada *output*. O valor destas variáveis é modificado de acordo com o resultado da execução do *Code action*. Se o *Output pattern* não é declarado, então, nenhuma variável CPN pode ser calculada.

Por fim, um *Code action* é uma expressão ML precedida pela palavra reservada *action* e executado como uma declaração local em um ambiente contendo variáveis CPN especificadas no *Input pattern*. Nesta expressão, nenhuma declaração de conjunto de cores, variáveis CPN ou de referência pode ser realizada. No entanto, ela pode utilizar constantes pré-declaradas ou declaradas pelo usuário, operações e funções. Além disto, novas funções e constantes podem ser definidas localmente por meio da cláusula *let-in-end*.

Na Figura 15, um retardo de tempo e um segmento de código são associados à transição t_1 com o objetivo de gerar novas fichas com certo retardo (de tempo) nos lugares $P1$ e $P2$ sendo que estas fichas têm valores diferentes entre si. Além destas inscrições, uma inscrição de guarda é associada à transição t_2 para permitir o disparo da mesma somente

quando o segundo elemento da ficha for maior ou igual a 6. Ao observar o estado do modelo na Figura 15, pode-se notar que a transição t_2 está habilitada por três fichas, entretanto, devido à presença da guarda, o disparo não é efetivado dado que todas as três fichas contém o segundo elemento menor que 6.

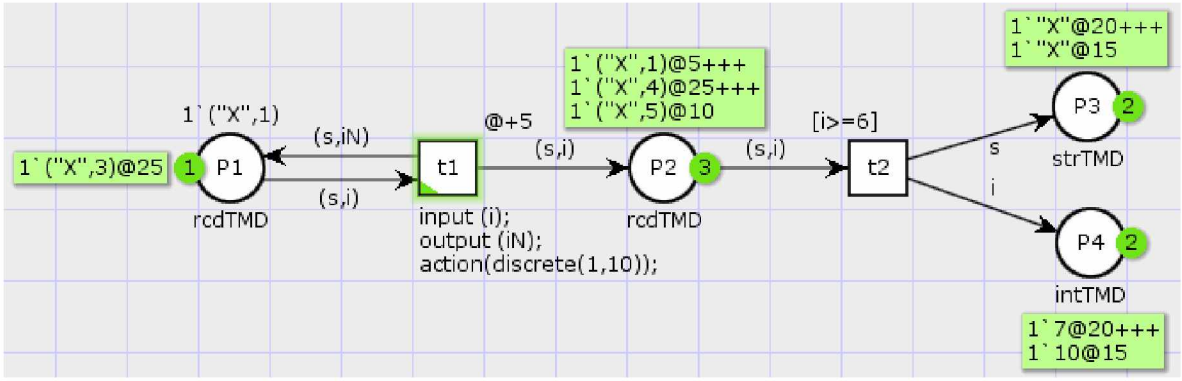


Figura 15 – Inscrição de guarda, retardo de tempo e segmento de código associadas a uma transição.

2.1.4.4 Rede de Petri a objetos

Antes de definir uma rede de Petri a objetos, é importante considerar a noção de objeto. O conceito de objeto, cada vez mais utilizado em Engenharia de *Software*, consiste em estruturar um programa computacional (aplicativo ou software) em torno das entidades envolvidas, encapsulando, ao mesmo tempo, estruturas de dados, sob a forma de uma lista de atributos e métodos de transformação destes dados. Esta abordagem opõe-se ao procedimento funcional segundo o qual se definem de maneira separada as estruturas de dados e as funções que o processo deve executar (CARDOSO; VALETTE, 1997).

Uma *classe de objetos* é definida por um conjunto de *atributos* (também chamados *propriedades*) e um conjunto de *operações* ou *métodos* que permitem manipular os valores dos atributos (BEZERRA, 2007). Pode-se definir uma classe a partir de outra por *herança*. A nova classe (chamada subclasse) *herda* as definições de atributos da classe anteriormente definida. Estas definições podem ser completadas por atributos e operações específicas.

As classes são apenas definições. Um objeto particular é uma *instância de classe*. Pode-se atribuir valores aos atributos destes objetos e executar suas operações. O encapsulamento confere aos objetos uma certa autonomia. Um objeto *nasce* (instanciação) e *desaparece* (destruição) dinamicamente durante a execução do programa, mas tais eventos devem ser raros em relação ao número de eventos correspondendo à execução de operações sobre os atributos.

A rede de Petri a objetos definidas por Sibertin-Blanc (2001) baseia-se na integração da rede Predicado/Transição e do conceito de um processo com descrição orientado a objetos. Neste tipo de rede, as fichas são consideradas como n-uplas de instâncias de classes de objetos e transportam estruturas de dados definidas como conjuntos de atributos de classes

específicas. Nas transições, por sua vez, são associadas operações, as quais correspondem às pré-condições (filtros) que operam sobre os atributos dos objetos situados nos lugares de entrada, e ações que modificam estes valores. Uma operação associada a uma transição t só poderá ser executada por um objeto se este estiver localizado num lugar de entrada de t (CARDOSO; VALETTE, 1997).

A definição formal da rede de Petri a objetos é dada a seguir (CARDOSO; VALETTE, 1997) (SIBERTIN-BLANC, 2001):

Definição 17 ***(Rede de Petri a objetos)*** *Uma rede de Petri a objetos marcada pode ser definida pela sêxtupla*

$$N_{RPO} =< R, C_{lass}, V_f, A_{tc}, A_{ta}, M_0 > \tag{11}$$

onde:

- R *é uma rede de Petri;*
- C_{lass} *representa um conjunto finito de classes de objetos; para cada classe, um conjunto de atributos é definido e organizado em uma hierarquia;*
- V_f *é um conjunto de variáveis formais, cujos tipos são dados por* C_{lass}*;*
- A_{tc} *é uma aplicação que, a cada transição, associa uma condição que envolve o conjunto de atributos e métodos dos objetos por meio das variáveis formais* V_f *associados aos arcos de entrada;*
- A_{ta} *é uma aplicação que, a cada transição, associa uma ação que envolve os atributos ou métodos das variáveis formais associados aos arcos de saída, permitindo modificar seus atributos específicos por meio da invocação de seus métodos;*
- M_0 *é a marcação inicial que associa, a cada lugar, uma soma dos objetos (n-uplas de instâncias de classes que pertencem a* C_{lass}*).*

Um exemplo de rede de Petri a objetos é representado na Figura 16. Para esta rede são definidos:

- o conjunto de classes $C_{lass} = \{Produto, Pedido\}$ com:

$$\text{Produto} = \begin{cases} nome : identificador; \\ codigo : integer; \\ preco : float; \end{cases} \qquad \text{Pedido} = \begin{cases} codigo : integer; \\ custo : float; \\ tipo : identificador; \end{cases}$$

- os tipos das variáveis:
  - $tipo(pr) = Produto$;

- $tipo(pd) = Pedido$.

❏ os tipos dos lugares:

- $tipo(Estoque\ de\ Produtos) = Produto$;
- $tipo(Buffer\ de\ Pedidos) = Pedido$;
- $tipo(Pedidos\ Processados) = Pedido$.

❏ a marcação inicial M_0 dada pelos objetos que se encontram nos lugares:

$$M_0 = \begin{bmatrix} < pr1 > + < pr2 > + < pr3 > \\ < pd1 > + < pd2 > \\ 0 \end{bmatrix}$$

em que os objetos $< pr1 >$, $< pr2 >$, $< pr3 >$ são da classe *Produto*, e os objetos $< pd1 >$ e $< pd2 >$ são da classe *Pedido*.

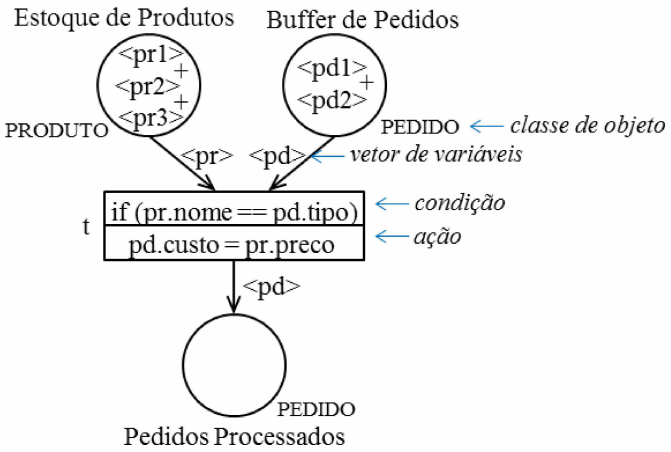


Figura 16 – Modelo de um processo representado por uma rede de Petri a objeto.

Nos lugares *Estoque de Produto* e *Buffer de Pedidos*, os objetos (fichas) possuem valores de atributos. Por exemplo, para o objeto $< pr1 >$, os valores de atributos podem ser dados por:

$$\text{Produto } pr1 = \begin{cases} nome : home\ theater; \\ codigo : 123440; \\ preco : 278,50; \end{cases}$$

E para o objeto $< pd2 >$, valores dos atributos podem ser dados por:

$$\text{Pedido } pd2 = \begin{cases} codigo : 123440; \\ custo : 00,00; \\ tipo : home\ theater; \end{cases}$$

A definição detalhada que fixa as regras de sensibilização e disparo das transições da rede de Petri a objetos encontra-se em (SIBERTIN-BLANC, 2001).

As variáveis dos arcos de entrada das transições podem ser substituídas por variáveis (objetos) de mesmo tipo que se encontram nos lugares de entrada das transições. Entretanto, é necessário que seja satisfeita a pré-condição de cada transição. Se esta pré-condição é atendida, o disparo da transição remove os objetos dos lugares de entrada da transição, altera os valores dos atributos de acordo com a ação existente na transição e, em seguida, produz um novo objeto com valores atualizados nos lugares de saídas correspondentes.

Pode-se notar que, na Figura 16, a transição t pode ser habilitada pela marcação inicial. Os valores dos atributos da variável *pr* associada ao arco que liga o lugar *Estoque de Produtos* à transição t podem ser substituídos pelos valores dos atributos de um dos objetos que se encontram em *Estoque de Produtos*. Da mesma forma, os valores dos atributos da variável *pd* associada ao arco que liga o lugar *Buffer de Pedidos* à transição t podem ser substituídos pelos valores dos atributos de um dos objetos que se encontram em *Buffer de Pedidos*.

Inicialmente, a pré-condição associada à transição t deve ser satisfeita para que a transição t seja sensibilizada (habilitada). A pré-condição $if(pr.nome == pd.tipo)$ verifica se os atributos *nome*, da variável *pr*, e *tipo*, da variável *pd*, possuem os mesmos valores. Se a condição é satisfeita, a transição t está habilitada e pode ser disparada. A ação associada à transição t é então executada, registrando o valor do atributo *preco*, da variável *pr*, no atributo *custo*, da variável *pd*. Após o disparo da transição t, um novo objeto, com o valor do atributo *custo* modificado, é produzido no lugar *Pedidos Processados*.

Na rede de Petri a objetos ilustrada pela Figura 16, como o par de objetos $(pr1, pd2)$ verifica a condição associada à transição t, a transição t pode ser disparada, produzindo o objeto *pd*2 no lugar *Pedidos Processados*. A Figura 17 ilustra a rede de Petri a objetos com a nova marcação. Para os demais objetos da rede, os disparos podem ser realizados da mesma forma.

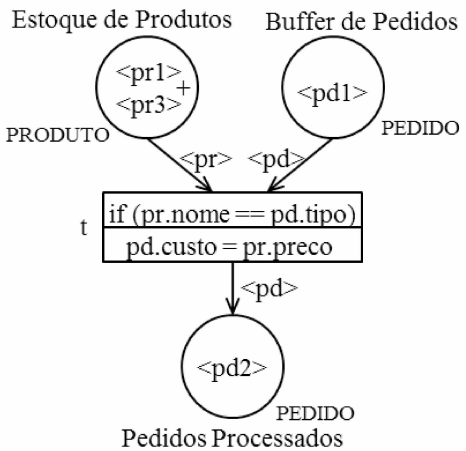


Figura 17 – Disparo da transição t_1 do modelo da Figura 16.

2.1.4.5 Rede de Petri Possibilística

Uma rede de Petri possibilística combina a teoria da possibilidade (DUBOIS; PRADE, 2009) com a rede de Petri a objeto (SIBERTIN-BLANC, 2001) e tem como objetivo levar em conta as informações incompletas sobre um processo. Neste modelo, um lugar, uma transição e uma sequência de disparos denotam, respectivamente, um possível estado parcial, uma possível troca de estado e um possível comportamento (CARDOSO, 1999). A principal vantagem desta rede é a capacidade de atualizar o estado do processo com informações incompletas sem originar estados inconsistentes. Uma rede de Petri possibilística é definida formalmente a partir de uma rede de Petri a objeto.

Por ela ser derivada da rede de Petri a objeto, as fichas numa rede de Petri possibilística são instâncias de classes ou subclasses de objetos e seus atributos podem estar envolvidos nos predicados associados com as transições atuando como condições extras no disparo das mesmas. Ela também pode ser modificada pela execução de uma ação quando a transição correspondente é disparada.

A fim de representar a incerteza, uma rede de Petri possibilística associa uma distribuição de possibilidade $\pi_b(p)$ à localização do objeto b (p corresponde à um lugar da rede). A partir disso, as seguintes declarações podem ser enunciadas:

- $\pi_b(p) = 1$ representa o fato de que o lugar p é uma possível localização do objeto b;
- $\pi_b(p) = 0$ expressa a certeza de que o objeto p não está presente no lugar b.

Considerando as declarações acima, dizer que é "completamente possível" (possibilidade igual a 1) que o objeto b esteja no lugar p não o proíbe de estar em outro lugar p', ou seja, não inibe que a localização do objeto b seja imprecisa. Em compensação, dizer que é "completamente certo" que o objeto b esteja no lugar p o impede de se localizar em qualquer outro lugar.

A marcação numa rede de Petri possibilística corresponde ao conhecimento acerca do estado do modelo do processo em um determinado instante do tempo. Formalmente, esta marcação é um mapeamento:

$$M : B \times P \rightarrow \{0, 1\} \tag{12}$$

onde B é um conjunto de objetos e P um conjunto de lugares. Se $M(b, p) = 1$, existe uma possibilidade do objeto b estar no lugar p. Caso contrário ($M(b, p) = 0$), não existe a possibilidade do objeto b estar no lugar p. A partir disto, a marcação M permite representar (CARDOSO, 1999):

- *uma marcação precisa*: $M(b, p_i) = 1$ e $\forall\ p_j \neq p_i$, $M(b, p_j) = 0$ representa o comportamento normal do processo, isto é, cada objeto está localizado em apenas um lugar (estado bem conhecido);

- *uma marcação imprecisa*: $M(b, p_i) = 1$ e $\exists\ p_j \neq p_i |\ M(b, p_j) = 1$ representa algumas deduções realizadas durante a execução do modelo do processo, ou seja, a localização do objeto tem uma distribuição de possibilidade sobre um conjunto de lugares. Nesta marcação não se pode afirmar que um objeto está num dado lugar, mas apenas que está em um lugar dentre de um conjunto de lugares.

A marcação evolui quando uma nova informação permite afirmar que o evento correspondente a uma transição ocorreu ou que é possível ter ocorrido. Assim sendo, a sua evolução depende da certeza (ou incerteza) da ocorrência do evento esperado. Para tanto, o disparo de uma transição é decomposta em dois passos:

- *início do disparo*: os objetos são colocados nos lugares de saída de t, mas não são removidos de seus lugares de entrada;

- *término do disparo*: pode ser o cancelamento do disparo (os objetos são removidos dos lugares de saída de t), ou a finalização do disparo (os objetos são removidos dos lugares de entrada de t).

Considerando a decomposição acima apresentada, dois tipos de disparo são definidos para uma rede de Petri possibilística:

- *disparo certo*: corresponde ao disparo de uma transição como é usualmente entendido na rede de Petri clássica, ou seja, a alteração do estado do processo modelado é instantânea. Neste tipo de disparo, a marcação antes do disparo, isto é, a localização de todos os objetos envolvidos, deve ser precisa, a ação associada à transição é executada e a nova marcação resultante do disparo da transição também é precisa;

- *disparo incerto* (ou pseudo-disparo): não corresponde a uma evolução normal do processo. Considera-se apenas o início do disparo, uma vez que não existe informação alguma que permita ter certeza sobre a ocorrência ou não do evento associado com a transição. Neste tipo de disparo a imprecisão sobre a marcação, isto é, a localização dos objetos, aumenta até que a ocorrência de um evento permita deduzir a correta localização dos mesmos, possibilitando assim retornar ao conhecimento certo do processo. Durante este período de tempo, duas marcações podem ser consideradas: a marcação M existente antes do disparo da transição e a marcação M' ($M \xrightarrow{t} M'$) alcançada após o disparo incerto da transição t correspondente ao evento.

Após o disparo incerto, a distribuição de possibilidade de cada objeto presente nos lugares de entrada e de saída da transição é igual à 1 (CARDOSO; VALETTE; DUBOIS, 1991). Este disparo permite evitar contradições no caso de um comportamento anormal do processo, pois o conhecimento impreciso leva em conta um conjunto maior de possíveis eventos.

Uma transição pode ser disparada de forma incerta se e, somente se, a distribuição de possibilidade dos objetos pertencentes aos lugares de saída da transição a ser pseudo-disparada for igual a 0. Caso essa restrição seja violada, diversos objetos podem ser inseridos nos lugares de saída criando incontáveis sequências de disparos, inviabilizando assim a restauração do processo quando uma informação precisa for recebida (CARDOSO; VALETTE; DUBOIS, 1991).

A interpretação de uma rede de Petri possibilística é definida anexando a cada transição uma função de autorização $\eta_{x_1,...,x_n}$, a qual representa uma condição extra:

$$\eta_{x_1,...,x_n} : T \rightarrow \{falsa, incerta, verdadeira\} \tag{13}$$

onde $x_1, ..., x_n$ são as variáveis associadas aos arcos de entrada da transição $t \in T$. Por meio desta interpretação é possível diferenciar as situações para as quais uma transição está habilitada por um disparo certo ou por um incerto.

Considerando uma transição t qualquer e uma possível substituição $b_1, ..., b_n$ para as possíveis variáveis $x_1, ..., x_n$ associadas aos arcos de entrada desta transição, diversas situações podem ser consideradas:

- t não está habilitada pela marcação mas sua interpretação é verdadeira; essa situação é proibida e um alarme é acionado;
- t está habilitada por uma marcação precisa e uma interpretação verdadeira; então um disparo certo pode ocorrer;
- t está habilitada por uma marcação precisa e uma interpretação incerta; então a transição t é pseudo-disparada e a incerteza aumenta;
- t está habilitada por uma marcação imprecisa e uma interpretação incerta; então t é pseudo-disparada;
- t está habilitada por uma marcação imprecisa e uma interpretação verdadeira: o algoritmo de defuzzificação, detalhado na subseção 2.1.4.5.1, é acionado e uma nova distribuição de possibilidade dos objetos envolvidos na marcação imprecisa é calculada a fim de voltar a uma marcação precisa.

As situações anteriormente descritas são ilustradas no fluxograma do "jogador" de rede de Petri possibilística mostrado na figura 18.

Para exemplificar o funcionamento de uma rede de Petri possibilística, a rede representada na Figura 19 é considerada. Para esta rede são definidos:

- o conjunto de classes $C_{lass} = \{Classe_1, Classe_2, (Classe_1, Classe_2)\}$ com:

$$Classe_1 = \left\{ \; data : date \; \right\}$$

Nenhum atributo ou função é definido para a classe $Classe_2$, pois, para o entendimento do exemplo, não é necessário tal detalhamento;

- as variáveis x e y associadas aos arcos de entrada da transição t pertencem aos tipos:
  - $\text{tipo}(x) = Classe_1$;
  - $\text{tipo}(y) = Classe_2$.
- os lugares pertencem aos tipos:
  - $\text{tipo}(P_1) = Classe_1$;
  - $\text{tipo}(P_2) = Classe_2$;
  - $\text{tipo}(P_3) = (Classe_1, Classe_2)$.
- a marcação inicial M_0 dada pelos objetos que se encontram nos lugares:

$$M_0 = \begin{bmatrix} < b_1 > + < b_2 > \\ < b_3 > \\ 0 \end{bmatrix}$$

em que os objetos $< b_1 >$ e $< b_2 >$ são da classe $Classe_1$, e o objeto $< b_3 >$ é da classe $Classe_2$;

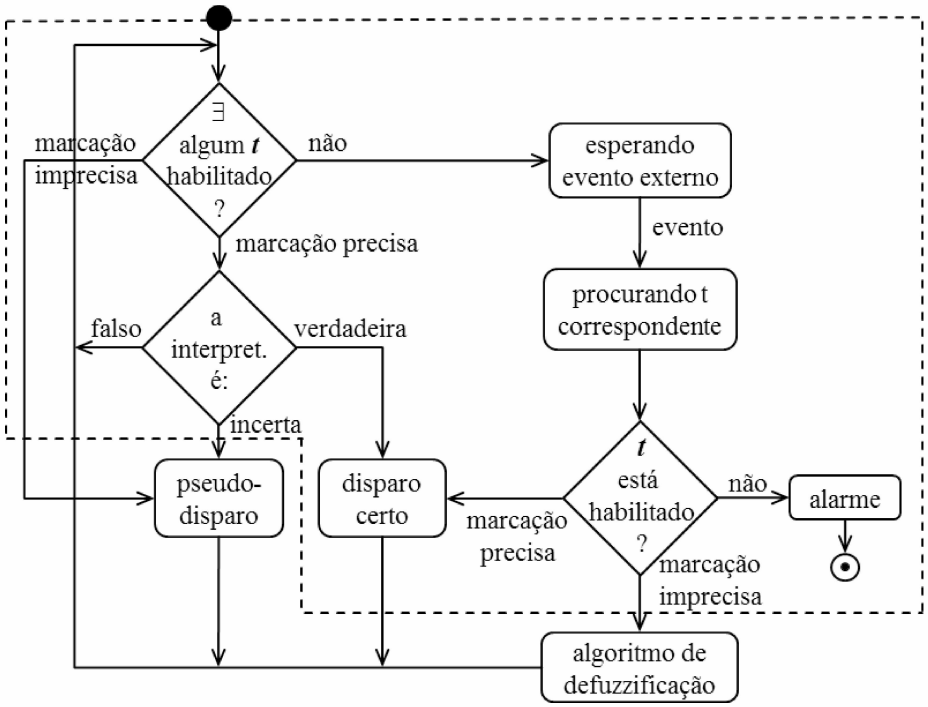


Figura 18 – "Jogador" de uma rede de Petri Possibilística (CARDOSO, 1999).

- a interpretação dada pela função de autorização:

$$\eta_{x,y} = \forall_y \begin{cases} incerto & se \quad (\tau < x.date) \wedge (sinal(x)) \\ verdadeiro & se \quad (\tau \geq x.date) \wedge (sinal(x)) \\ falso & caso \quad contrário \end{cases} \tag{14}$$

onde $sinal(x)$ é verdadeiro quando, considerando um ambiente externo, o sensor associado à transição está ativo e τ é o tempo corrente.

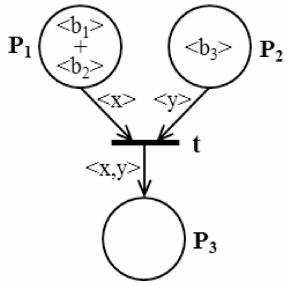


Figura 19 – Exemplo de um modelo de processo representado por uma rede de Petri Possibilística (CARDOSO, 1999).

A função de autorização impõe à transição t restrições para o disparo da mesma (CARDOSO, 1999). O disparo certo ocorre quando a mensagem ($sinal(x)$), sinalizando que o objeto em questão foi envolvido num evento associado à transição t, é recebida após o tempo determinado pelo atributo *data* associado ao objeto. Já o disparo incerto ocorre se a mensagem é recebida antes do tempo determinado, representando um comportamento possível mas não normal. Este disparo permite deduzir que a mensagem é errônea ou que o estado do processo atual não condiz com a representação do mesmo no modelo (a rede de Petri marcada).

A Figura 19 ilustra o comportamento de um processo na rede de Petri possibilística considerando uma certa marcação inicial com duas possíveis substituições para as variáveis associadas aos arcos de entrada da transição t ($S_1 = (b_1, b_3)$ e $S_2 = (b_2, b_3)$). Assumindo-se que $b_1.date = 20$, $b_2.date = 40$, $sinal(b_2) = verdade$ no tempo $\tau_1 = 25$ e $sinal(b_1) = verdade$ no tempo $\tau_2 = 35$. A evolução da marcação considerando os valores definidos anteriormente é descrita abaixo:

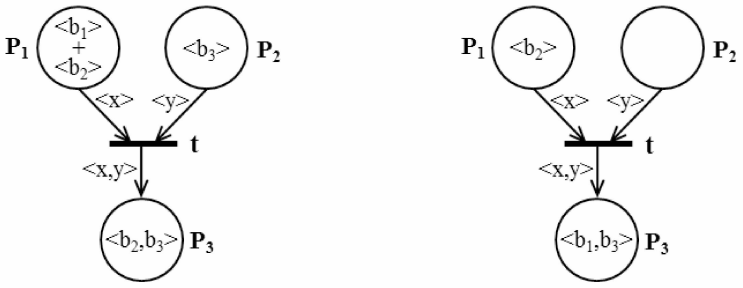
- no tempo $\tau = 25$, $sinal(b_2) = verdade$ é recebida mas não corresponde ao comportamento normal dado que $b_2.date > 25$ ($\eta_{b_2 b_3}(t) = incerto$); logo t é pseudo-disparada considerando a substituição S_2 (Figura 20a);

- após o tempo $\tau = 25$ a marcação é imprecisa e cobre duas alternativas:

  1. o evento correspondente ao disparo de t para a tupla $< b_2, b_3 >$ realmente ocorreu, ou seja, a informação dada por $sinal(b_2)$ foi correta; ou

2. o evento correspondente ao disparo de t para a tupla $< b_2, b_3 >$ não ocorreu, isto é, essa transição ainda pode ser disparada, com $< b_1, b_3 >$ ou com $< b_2, b_3 >$.

❑ no tempo $\tau = 35$, o recebimento de $sinal(b_1) = verdade$ corresponde ao comportamento normal dado que $b_1.date \leq 35$ ($\eta_{b_1 b_3} = verdadeira$). Como t está habilitada por uma marcação imprecisa e a interpretação associada a ela é verdadeira, o algoritmo de defuzzificação, detalhado na subseção 2.1.4.5.1, é chamado para retornar a uma marcação precisa. A aplicação do algoritmo cancela o disparo incerto da transição t para $< b_2, b_3 >$. Considerando agora que a marcação é precisa, tem-se o disparo certo de t utilizando a substituição S_1 (Figura 20b).



(a) Após o disparo incerto (b) Após o disparo certo

Figura 20 – Evolução da rede de Petri possibilística da Figura 19 (CARDOSO, 1999).

A figura 21 descreve a distribuição das possibilidades das instâncias b_1, b_2 e b_3 em função do tempo (as linhas mais espessas representam uma possibilidade igual a 1 e as linhas mais finas uma possibilidade igual a 0). Observando a distribuição de possibilidades, pode-se notar que durante o período de tempo $[25, 35]$ o estado do processo é incerto, ou seja, as instâncias b_2 e b_3 se encontram em dois lugares durante este tempo (b_2 em P_1

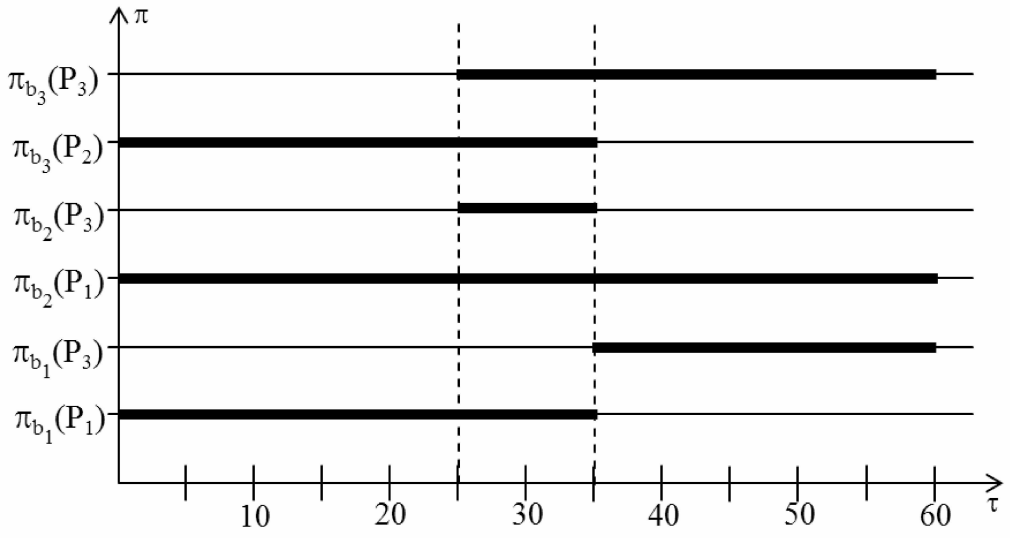


Figura 21 – Distribuição das possibilidades das localizações de b_1, b_2 e b_3 (CARDOSO, 1999).

e P_3 e b_3 em P_2 e P_3). Isto sucede devido à ocorrência do disparo incerto da transição t no tempo $\tau = 25$. Além disto, pode-se notar também que após o tempo $\tau = 35$ o estado do processo é certo, isto é, as instâncias b_1, b_2 e b_3 se encontram em um único lugar (P_3, P_1 e P_3 respectivamente). Tal estado é obtido a partir da aplicação do algoritmo de defuzzificação e do disparo certo da transição t no tempo $\tau = 35$.

2.1.4.5.1 Algoritmo de Defuzzificação

O disparo incerto de uma transição t, como mencionado anteriormente, pode ser considerado apenas como o início de um disparo certo. De fato, as fichas são adicionadas nos lugares de saída, mas não são removidas dos seus lugares de entrada. Isso ocorre pela falta de certeza que a transição t foi ou não realmente disparada.

Para restabelecer o estado certo do processo, faz se necessário a existência de um procedimento que realize uma nova computação das distribuições de possibilidade, levando em conta o recebimento de uma nova informação, ou seja, a função de autorização relacionada a uma transição t qualquer se tornou verdadeira ($\eta(t) = verdadeira$). Entretanto, tal procedimento recupera apenas o estado certo dos objetos envolvidos neste evento, ou seja, caso existam outros disparos incertos que não se relacionem diretamente ou indiretamente a este evento, eles não são recuperados, continuando assim numa situação de incerteza do conhecimento.

Ao realizar o novo cálculo das distribuições de possibilidade, deve-se decidir, para cada transição t que foi pseudo-disparada, se as mesmas foram ou não efetivamente disparadas. No primeiro, caso considera-se que o disparo foi finalizado (t realmente foi disparada) e, no segundo, que o disparo foi cancelado (t não foi disparada). Em outras palavras:

- disparo incerto finalizado: a transição foi disparada, pois o evento correspondente realmente ocorreu. As distribuições de possibilidade dos objetos envolvidos no disparo são redefinidas para 0 (zero) nos lugares de entrada da transição;

- disparo incerto cancelado: a transição não foi disparada pois o evento correspondente não ocorreu. As distribuições de possibilidade dos objetos envolvidos no disparo são redefinidas para 0 (zero) nos lugares de saída da transição.

O algoritmo de defuzzificação apresentado em (CARDOSO; VALETTE; DUBOIS, 1991) é descrito no Algoritmo 1 (considere LL uma lista de lugares). Para o seu correto funcionamento, nenhum disparo incerto pode ter sido realizado caso a distribuição de possibilidade dos objetos pertencentes aos lugares de saída da transição em questão era igual à 1. Se tal restrição for respeitada, este algoritmo sempre finaliza corretamente, pois a sequência de disparo é finita.

Para ilustrar o funcionamento do algoritmo, a rede de Petri possibilística apresentada na Figura 22a é utilizada. Neste exemplo, as transições t_1, t_2, t_3 e t_4 foram pseudo-

Algoritmo 1 Defuzzificação

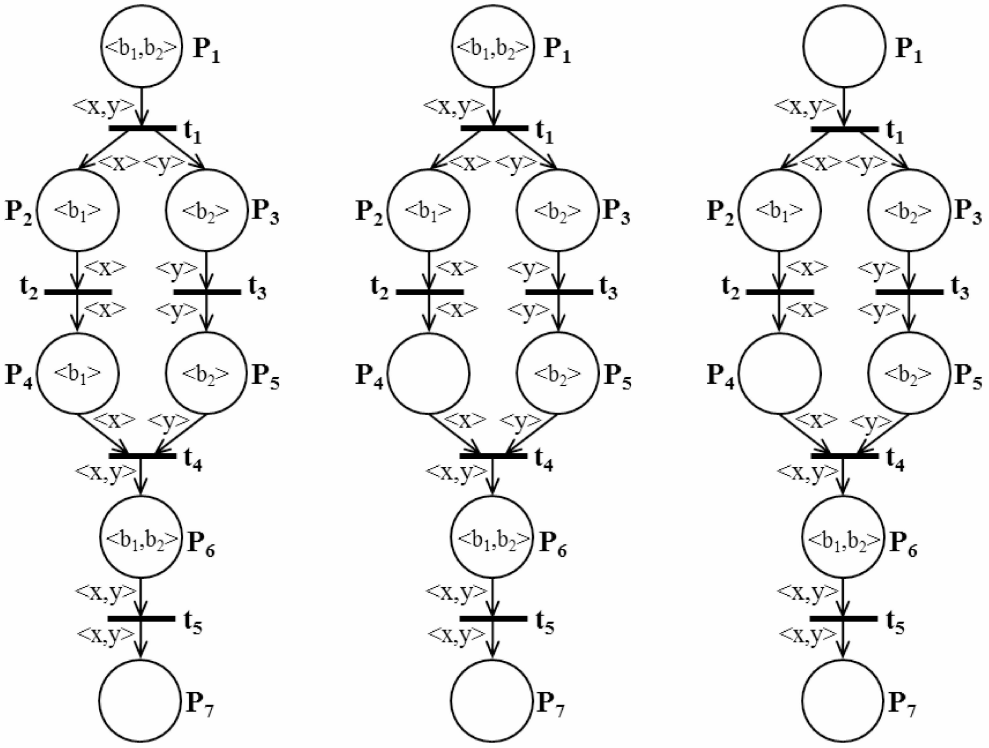
Entrada: Rede de Petri com posicionamento impreciso dos objetos nos lugares, transição t

Saída: Rede de Petri com posicionamento certo dos objetos nos lugares

1: **if** t foi pseudo-disparada anteriormente ($\eta(t) = incerta$) **then**
2: $LL \leftarrow (\bullet t) \cup (t\bullet)$
3: cancelar o disparo de t
4: **else**
5: $LL \leftarrow (\bullet t)$
6: **end if**
7: **while** $LL \neq \phi$ **do**
8: $p \leftarrow$ LL[0]
9: **while** $\exists\ t_i \in (\bullet p) \mid \eta(t_i) = incerta$ **do**
10: $LL \leftarrow LL \cup (\bullet t_i)$
11: finalizar o disparo de t_i
12: **end while**
13: **while** $\exists\ t_j \in (p\bullet) \mid \eta(t_j) = incerta$ **do**
14: $LL \leftarrow LL \cup (t_j\bullet)$
15: cancelar o disparo de t_i
16: **end while**
17: **end while**

disparadas obtendo assim um estado incerto. Considerando que a função de autorização da transição t_2 ($\eta(t_2)$) está avaliada como verdadeira após tais disparos, a sequência de eventos ao aplicar o algoritmo é descrita abaixo:

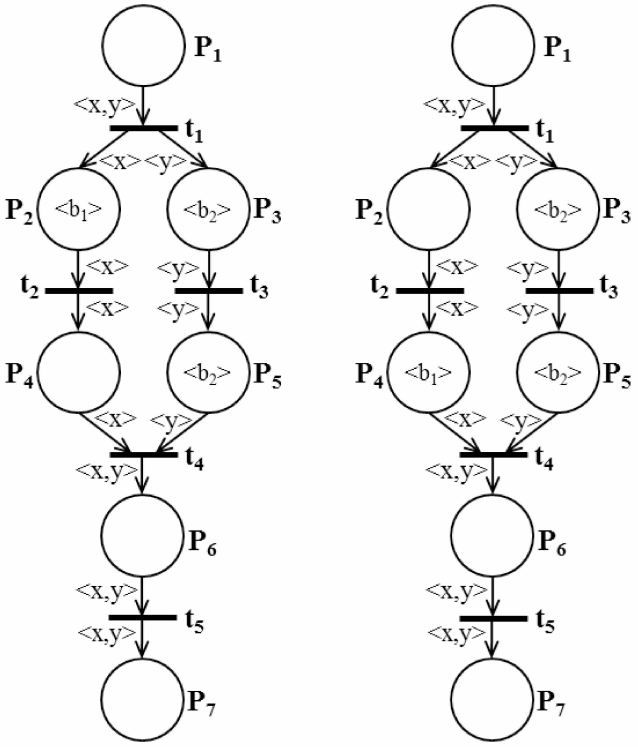
- t_2 foi pseudo-disparada anteriormente ($\eta(t_2) = incerta$), logo:
  - $LL \leftarrow \{P_2\} \cup \{P_4\} = \{P_2,\ P_4\}$;
  - o disparo de t_2 é cancelado, ou seja, a ficha $< b_1 >$ é removida do lugar P_4 ($M(b_1, P_4) = 0$) e mantida no lugar P_2 ($M(b_1, P_2) = 1$) (Figura 22b).
- $LL \neq \{\}$ então $p \leftarrow P_2$, e:
  - a transição t_1 pertencente ao conjunto das transições precedentes de P_2 foi pseudo-disparada ($\eta(t_1) = incerta$), então:
    - $LL \leftarrow \{P_4\} \cup \{P_1\} = \{P_4,\ P_1\}$;
    - o disparo de t_1 é finalizado, ou seja, a ficha $< b_1, b_2 >$ é removida do lugar P_1 ($M(b_1, P_1) = 0$ e $M(b_2, P_1) = 0$) e as fichas $< b_1 >$ e $< b_2 >$ são mantidas, respectivamente, nos lugares P_2 ($M(b_1, P_2) = 1$) e P_3 ($M(b_2, P_3) = 1$) (Figura 22c).
  - não existe nenhuma transição pertencente ao conjunto das transições seguintes de P_2 que foram pseudo-disparadas.



(a) Marcação imprecisa

(b) Disparo de t_2 cancelado

(c) Disparo de t_1 finalizado



(d) Disparo de t_4 cancelado

(e) Disparo certo de t_2

Figura 22 – Aplicação do algoritmo de defuzzificação.

- $LL \neq \{\}$ então $p \leftarrow P_4$, e:
  - não existe nenhuma transição pertencente ao conjunto das transições precedentes de P_4 que foram pseudo-disparadas;
  - a transição t_4 pertencente ao conjunto das transições seguintes de P_4 foi pseudo-disparada ($\eta(t_4) = incerta$), então:
    * $LL \leftarrow \{P_1\} \cup \{P_6\} = \{P_1, P_6\}$;
    * o disparo de t_4 é cancelado, ou seja, a ficha $< b_2 >$ é mantida em P_5 ($M(b_2, P_5) = 1$) e a ficha $< b_1, b_2 >$ é removida do lugar P_6 ($M(b_1, P_6) = 0$ e $M(b_2, P_6) = 0$) (Figura 22d).
- $LL \neq \{\}$ então $p \leftarrow P_1$, e:
  - não existe nenhuma transição pertencente ao conjunto das transições precedentes ou seguintes de P_1 que foram pseudo-disparadas.
- $LL \neq \{\}$ então $p \leftarrow P_6$, e:
  - não existe nenhuma transição pertencente ao conjunto das transições precedentes ou seguintes de P_6 que foram pseudo-disparadas.
- $LL = \{\}$, logo a transição t_2 é disparada com certeza produzindo o objeto $< b_1 >$ no lugar P_4 ($M(b_1, P_4) = 1$) e removendo-a do lugar P_2 ($M(b_1, P_2) = 0$) (Figura 22e).

Após a execução do algoritmo de defuzzificação, o posicionamento dos objetos nos lugares relacionados diretamente ou indiretamente ao evento que torna "$\eta(t_2) = verdadeira$" torna-se certo. Entretanto, pode-se notar que o posicionamento do objeto $< b_2 >$ continua impreciso, isso ocorre porque a informação recebida não permite deduzir se o disparo da transição t_3 foi ou não válido. Para cancelar ou finalizar tal disparo um evento relacionado a uma das transições t_3, t_4 ou t_5 deverá ocorrer, pois as mesmas são diretamente ou indiretamente relacionadas ao objeto b_2.

2.2 WorkFlow net

Um processo de negócio define quais tarefas precisam ser executadas e em qual ordem a execução deve ocorrer. Além disso, de acordo com Aalst (1998a), processos de negócio são baseados em casos, isto é, cada parte do trabalho é executada para um caso específico. Modelar um processo de negócio em termos de uma rede de Petri é relativamente simples: 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, 1998a)

(AALST; HEE, 2004). Assim, uma rede de Petri que modela um processo de negócio é uma *WorkFlow net (WF-net)* que satisfaz as seguintes propriedades (AALST, 1998a):

- tem apenas um lugar de início i (denominado *Start*) e apenas um lugar de término o (denominado *End*), sendo estes dois lugares tratados como lugares especiais;
- o lugar *Start* tem apenas arcos de saída e o lugar *End* apenas arcos de entrada;
- uma ficha em *Start* representa um caso que precisa ser tratado e uma ficha em *End* representa um caso que já foi tratado;
- toda tarefa t (transição) e condição p (lugar) devem estar em um caminho que se encontra entre o lugar *Start* e o lugar *End*.

A definição formal da WF-net é apresentada a seguir (AALST; HEE, 2004):

Definição 18 ***(WorkFlow net)*** *Uma rede de Petri* $WF = (P, T, Pre, Post)$ *é uma WF-net se e somente se:*

- *existe um único lugar de início* $i \in P$ *tal que* $\bullet i = \phi$*;*
- *existe um único lugar de término* $o \in P$ *tal que* $o\bullet = \phi$*;*
- *todo nó* $x \in P \cup T$ *está em um caminho entre os lugares* i *e* o*.*

Uma WF-net tem apenas um único lugar de início (i) e um único lugar de término (o) porque qualquer caso tratado pelo procedimento representado por ela é criado quando o mesmo entra nos WfMS's e é removido quando for completamente tratado pelo mesmo. Em outras palavras, uma WF-net especifica o ciclo de vida de um caso. O terceiro requerimento da Definição 18 é adicionado para evitar atividades e/ou condições que não contribuem para o processamento dos casos (AALST; HEE, 2004).

A fim de indicar explicitamente o início e o fim de cada atividade em execução, duas transições intercaladas por um lugar podem ser usadas para detalhar a (macro) transição do modelo do Aalst (WANG; TEPFENHART; ROSCA, 2009). Como pode ser observado na Figura 23a, o lugar A_1 representa a atividade em execução e as transições B e E representam, respectivamente, o início e o fim da execução desta atividade. Do ponto de vista da análise de alcançabilidade, a Figura 23a pode ser reduzida a uma única transição, como indicado na Figura 23b através da (macro) transição A_1, a qual representa toda a atividade em execução como uma única unidade lógica.

(a) Atividade associada ao lugar (b) Atividade associada à transição

Figura 23 – Representação de uma Atividade.

A verificação de corretude do processo de negócio pode ser realizada por meio de uma WF-net mediante a análise de seu grafo das marcações acessíveis G_A. Alguns erros de modelagem podem ser identificados por meio da análise do modelo, tais como:

- tarefas sem condição de entrada: quando não está especificada a condição necessária para a execução da tarefa;
- tarefas sem condição de saída: quando a tarefa não tem influência no processamento do caso;
- *deadlock*: quando o processamento do caso é bloqueado, não sendo possível ocorrer uma condição que permita a continuação da execução deste caso;
- *livelock*: quando o processo entra em um ciclo no qual a condição de saída nunca ocorre;
- atividades ainda a serem executadas após o caso ter alcançado o estado final;
- fichas que permanecem no modelo do processo após a conclusão do processamento do caso. Por exemplo, quando se tem a geração de uma informação duplicada.

Para ilustrar o mapeamento de um processo por uma WF-net, considera-se o processo de tratamento de reclamações apresentado em (AALST; 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, 2004) p.52.

> Tradução: Uma reclamação é inicialmente gravada. Então, o cliente que efetuou a reclamação e o departamento responsável pela reclamação são contactados. O cliente é questionado para maiores informações. O departamento é informado sobre a reclamação. Estas duas tarefas podem ser executadas em paralelo, isto é, simultaneamente ou em qualquer ordem. Depois disso, os dados são recolhidos e uma decisão é tomada. Dependendo da decisão, ou um pagamento de compensação é efetuado, ou uma carta é enviada. Finalmente, a reclamação é armazenada.

A Figura 24 mostra uma WF-net para este processo. Este modelo tem um erro que se encontra na sincronização das atividades *contact_client* e *contact_department*. Observa-se que, ao concluir a execução de uma destas atividades, a transição t_{bco}, referente ao início da execução da atividade *collect*, é disparada sem esperar a conclusão da execução da outra atividade em questão. Como consequência, uma segunda ficha é gerada, ou seja, ao fim da execução duas fichas estarão presentes no lugar *end* em vez de somente uma.

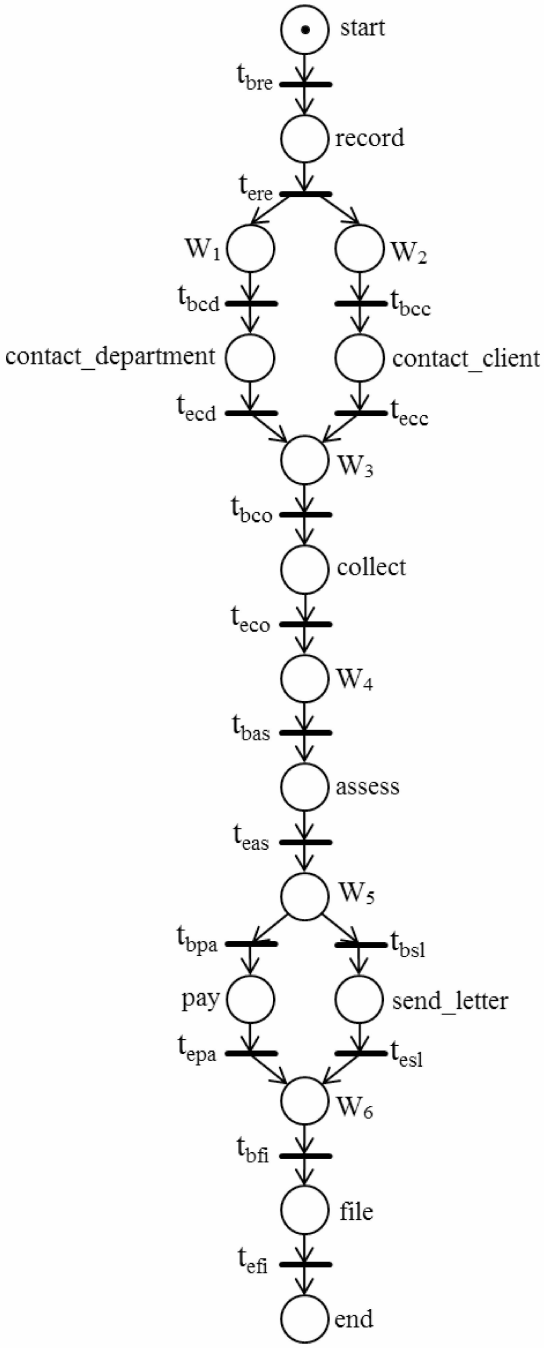


Figura 24 – Modelo com erro de um processo de tratamento de reclamações em WF-net.

De acordo com Aalst e Hee (2004), as seguintes construções básicas para o roteamento de tarefas devem ser consideradas:

- *sequencial*: a forma mais simples de execução de tarefas, onde uma tarefa é executada após a outra, havendo, claramente, dependência entre elas;

- *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;

- *condicional (ou rota seletiva)*: quando há uma escolha entre duas ou mais tarefas;
- *iterativa*: quando é necessário executar uma mesma tarefa múltiplas vezes.

Considerando o processo de tratamento de reclamações, mostrado na Figura 24, as tarefas *contact_client* e *contact_department* são um exemplo de roteamento paralelo, as tarefas *collect* e *assess* são um exemplo de roteamento sequencial e as tarefas *pay* e *send_letter* são um exemplo de roteamento condicional.

Um roteamento iterativo, conforme mostrado em (PASSOS; JULIA, 2009), pode ser substituído por uma tarefa global considerando a definição de blocos bem formados conforme apresentado por (VALETTE, 1979). Assim, uma rota iterativa pode ser substituída por uma tarefa global, como mostra a Figura 25.

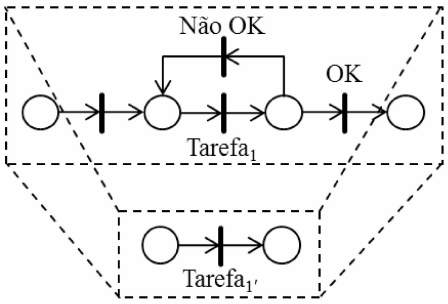


Figura 25 – Substituição de uma rota iterativa por uma tarefa global.

Além dos roteamentos, Aalst e Hee (2004) definem a noção de acionamento. Um acionamento, é uma condição externa que guia a execução de uma transição habilitada. Existem quatro tipos distintos de acionamentos:

- *usuário*: uma transição é acionada por um recurso humano;
- *mensagem*: um evento externo aciona uma transição habilitada;
- *tempo*: uma transição habilitada é acionada por um relógio, isto é, a transição é disparada em um tempo pré-definido;
- *automática*: uma transição é acionada no momento em que é habilitada e não requer interação humana.

A Figura 26 mostra como cada um dos quatro tipos distintos de acionamentos é graficamente associado às transições.

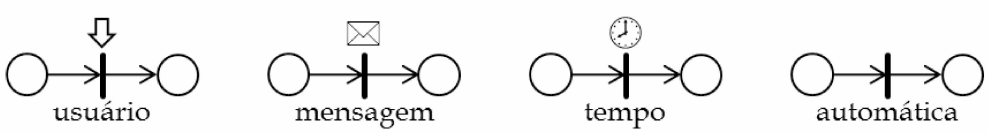


Figura 26 – Tipos de acionamentos definidos por Aalst e Hee (2004).

Nota-se que quando um acionamento do tipo "*usuário*" é considerado, a transição correspondente é acionada por um recurso humano, isto é, há uma alocação de recurso humano associada a esta transição. Nos demais tipos de transição, não há alocação de recursos humanos associada.

O processo de tratamento de reclamações mostrado na Figura 24 consiste de dezesseis transições, das quais seis são automaticamente tratadas (t_{bre}, t_{ere}, t_{bco}, t_{eco}, t_{bfi}, t_{efi}) e dez são acionadas por recursos humanos (t_{bcd}, t_{ecd}, t_{bcc}, t_{ecc}, t_{bas}, t_{eas}, t_{bpa}, t_{epa}, t_{bsl} e t_{esl}).

2.2.1 Soundness

Soundness é um critério de corretude definido para a WF-nets. Uma WF-net é *sound* se, e somente se, os três requisitos a seguir são satisfeitos (AALST; HEE, 2004):

- para cada ficha colocada no lugar de início i, uma (e apenas uma) ficha aparecerá no lugar de término o;
- quando uma ficha aparece no lugar o, todos os outros lugares estão vazios, considerando o caso em questão;
- considerando uma tarefa associada à uma transição t, é possível evoluir da 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 logicamente correto (*sound*) se ele não contém nenhuma tarefa desnecessária e se todo caso iniciado é totalmente concluído em algum momento, não restando nenhuma tarefa pendente (nenhuma ficha remanescente) no modelo. A definição formal do critério de correção *Soundness* no contexto da WF-net, proposto por Aalst em (AALST, 1998a), é apresentada na sequência.

Definição 19 ***(Soundness para WF-net)*** *Um processo de negócio modelado por uma WF-net $WF = (P, T, Pre, Post)$ é sound se, e somente se:*

- *para cada marcação M alcançável a partir da marcação inicial M_0, existe uma sequência de disparos que leva da marcação M para a marcação final M_f. Formalmente:*

$$\forall_M \, (M_0 \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} M_f); \tag{15}$$

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

$$\forall_M \, ((M_0 \xrightarrow{*} M) \wedge (M \geq M_f)) \Rightarrow (M = M_f); \tag{16}$$

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

$$\forall_{t \in T} \, \exists_{M,M'} \, (M_0 \xrightarrow{*} M \xrightarrow{t} M'). \tag{17}$$

A WF-net mostrada na Figura 24 não é *sound*, pois quando uma ficha atinge o lugar de término *End*, ainda há uma ficha remanescente no lugar W_3, considerando, por exemplo, a sequência de disparo $\alpha_1 = \{t_{bre}, t_{ere}, t_{bcc}, t_{ecc}, t_{bcd}, t_{ecd}, t_{bco}, t_{eco}, t_{bas}, t_{eas}, t_{bsl}, t_{esl}, t_{bfi}, t_{efi}\}$.

A fim de relaxar o critério de correção *Soundness* com o propósito de representar uma visão mais pragmática no contexto da correção, Dehnert e Rittgen (2001) propuseram um novo critério, denominado *Relaxed Soundness*, mais fraco que o critério de correção *Soundness* em um sentido formal. A ideia deste critério é que cada transição do modelo do processo pertencerá a pelo menos uma sequência de disparo, a qual finalizará a execução do modelo do processo corretamente, isto é, sem fichas remanescentes. A definição do critério de correção *Relaxed Soundness*, proposto por Dehnert e Rittgen (2001), é apresentada na sequência.

Definição 20 ***(Relaxed Soundness para WF-net)*** *Um processo de negócio modelado por uma WF-net $WF = (P, T, Pre, Post)$ é Relaxed Sound se, e somente se, cada transição está em uma sequência de disparo que inicia na marcação inicial M_0 e finaliza na marcação final M_f. Formalmente:*

$$\forall_{t \in T} \; \exists_{M', M''} \; (M_0 \xrightarrow{*} M' \xrightarrow{t} M'' \xrightarrow{*} M_f). \tag{18}$$

A WF-net mostrada na Figura 24 não é *Relaxed Sound*, pois o fato de haver ainda uma ficha remanescente na rede quando uma ficha atinge o lugar de término *End* é levado em consideração. Por exemplo, considerando todas as possíveis sequências de disparo que são variações de $\alpha_1 = \{t_{bre}, t_{ere}, t_{bcc}, t_{ecc}, t_{bcd}, t_{ecd}, t_{bco}, t_{eco}, t_{bas}, t_{eas}, t_{bsl}, t_{esl}, t_{bfi}, t_{efi}\}$ e $\alpha_2 = \{t_{bre}, t_{ere}, t_{bcc}, t_{ecc}, t_{bcd}, t_{ecd}, t_{bco}, t_{eco}, t_{bas}, t_{eas}, t_{bpa}, t_{epa}, t_{bfi}, t_{efi}\}$ obtidas da WF-net, tem-se que cada transição $t \in T$ está em uma delas. No entanto, quando estas sequências de disparo finalizam, ou seja, o lugar *End* é alcançado, há sempre uma ficha remanescente em algum lugar entre W_3 e $file$.

2.3 WorkFlow net Interorganizacional

Uma WorkFlow net Interorganizacional (IOWF-net) é uma rede de Petri que modela um processo de negócio interorganizacional. Um processo de negócio interorganizacional é um processo com n parceiros de negócio envolvidos, os quais operam de forma independente, porém, com certos pontos de comunicação e sincronização de suas atividades a fim de realizar corretamente o trabalho global (AALST, 1998b). Considerando que cada parceiro tem seu próprio processo de negócio local, uma IOWF-net é composta por modelos em WF-net (referentes ao processo de cada parceiro) e uma estrutura de interação.

A interação entre os processos pode ser realizada de duas maneiras: comunicação síncrona ou assíncrona. A primeira corresponde à fusão de um número de transições forçando a execução simultânea de tarefas específicas entre os modelos em WF-net, e a segunda

corresponde à troca de mensagens assíncronas. Na presente pesquisa as comunicações síncronas não serão consideradas uma vez que considera-se que cada parceiro envolvido controla seu próprio processo. Assim, somente as comunicações assíncronas serão consideradas. Na sequência, a IOWF-net considerada neste trabalho é definida. É relevante destacar que esta é uma adaptação da definição apresentada por Aalst (1998b).

Definição 21 ***(WorkFlow net Interorganizacional)*** *Uma IOWF-net é uma tupla*

$$IOWF\text{-}net = \{WF_1, WF_2, \ldots, WF_q, P_{AC}, AC\} \tag{19}$$

onde:

- $q \in \mathbb{N}$ *é a quantidade de processos modelados por uma WF-net;*
- *para cada* $k \in \{1, \ldots, q\}$*:* WF_k *é uma WF-net com lugar de início* i_k *e lugar de término* o_k*;*
- *para cada* $k, l \in \{1, \ldots, q\}$*: se* $k \neq l$ *então* $(P_k \cup T_k) \cap (P_l \cup T_l) = \emptyset$*;*
- $T^{\blacktriangledown} = \bigcup_{k \in \{1,\ldots,q\}} T_k$, $P^{\blacktriangledown} = \bigcup_{k \in \{1,\ldots,q\}} P_k$, $F^{\blacktriangledown} = \bigcup_{k \in \{1,\ldots,q\}} (P_k \cup T_k) \cap (T_k \cup P_k)$ *(relações entre os elementos dos modelos em WF-net);*
- P_{AC} *é o conjunto de elementos de comunicação assíncrona (lugares de comunicação);*
- $AC \subseteq P_{AC} \times \mathbb{P}(T^{\blacktriangledown}) \times \mathbb{P}(T^{\blacktriangledown})$ *é a relação de comunicação assíncrona*[9]*.*

Cada elemento de comunicação assíncrona, também conhecido como lugar de comunicação, corresponde a um lugar em P_{AC}. A relação AC especifica o conjunto de transições de entrada e o conjunto de transições de saída para cada elemento de comunicação assíncrona (AALST, 1998b). Assim, considerando uma transição t, um lugar de comunicação assíncrona p pode ser visto como um lugar de comunicação de entrada (do inglês *ICP*) ou um lugar de comunicação de saída (do inglês *OCP*) da transição t.

Para ilustrar os conceitos definidos acima, considere o processo de negócio interorganizacional apresentado em Aalst (1998b), o qual modela um processo que precede a apresentação de um artigo em uma conferência:

> This workflow has two loosely coupled workflow processes: (1) the process of an author preparing, submitting and revising a paper, and (2) the process of evaluating and monitoring submissions by the program committee. In this case there are two 'organizations' involved in the interorganizational workflow: the author (AU) and the program committee (PC). The author sends a draft version of the paper to the program committee. The program committee acknowledges the receipt and evaluates the submission. The paper is accepted or rejected by the program committee. In both cases the author is notified. If the paper is rejected, the workflow terminates, otherwise the author can start preparing the final version. After completing the final version, a copy is sent to

[9] $\mathbb{P}(T^{\blacktriangledown})$ é o conjunto de todos os subconjuntos não vazios de $T^{\blacktriangledown}$

> the program committee and the program committee acknowledges the receipt of the final version. If the final version is not received by the program committee by a specified due date, the author is notified that the paper is considered to be too late. A paper which is too late will not be published in the proceedings (AALST, 1998b) p.263.
>
> Tradução: Este processo de negócio interorganizacional possui dois processos de negócio fracamente acoplados: (1) o processo de um autor preparar, submeter e revisar o artigo, e (2) o processo de avaliação e monitoramento das submissões pelo comitê do programa. Neste caso, existem duas organizações envolvidas no processo de negócio interorganizacional: o autor (AU) e o comitê do programa (PC). O autor envia a versão preliminar do artigo para o comitê do programa. O comitê confirma o recebimento e avalia a submissão. O artigo é aceito ou rejeitado pelo comitê. Em ambos os casos o autor é notificado. Se o artigo for rejeitado, o processo termina, caso contrário o autor começa a preparar a versão final. Depois de completar a versão final, uma cópia é enviada ao comitê do programa que confirma o recebimento da versão final. Se a versão final não for recebida pelo comitê até a data final especificada, o autor é notificado que o artigo não pode ser mais considerado. O artigo que é recebido tardiamente não será publicado nos anais.

A Figura 27 mostra o modelo em IOWF-net deste processo. Este modelo em IOWF-net é composto por dois processos *AU* e *PC* onde cada um é modelado por uma WF-net. Cada um destes modelos em WF-net possui um lugar de início e um lugar de término. No caso do modelo *AU*, o lugar de início é *start_flow_author* e o lugar de término é *end_flow_author*. No modelo *PC*, o lugar de início e término são *start_flow_PC* e

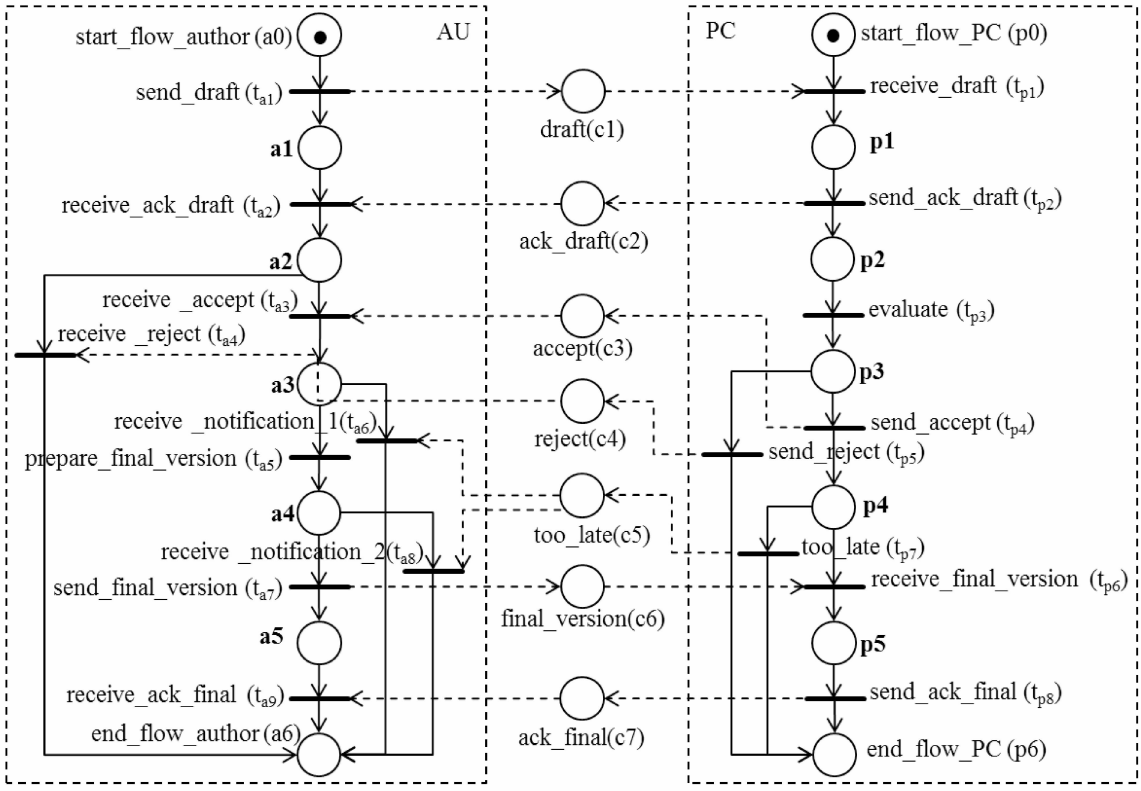


Figura 27 – Modelo de um processo que precede a apresentação de um artigo em uma conferência em IOWF-net (AALST, 1998b).

end_flow_PC, respectivamente. Os lugares *draft*, *ack_draft*, *accept*, *reject*, *too_late*, *final_version* e *ack_final* são os lugares de comunicação. O lugar de comunicação assíncrona *draft*, por exemplo, é o ICP da transição *receive_draft* e o OCP da transição *send_draft*.

2.3.1 Soundness para WorfkFlow net Interorganizacional

No contexto dos processos de negócio interorganizacionais, também é desejável que o critério de correção *Soundness* seja verificado. Como definido anteriormente para a WF-net, para um processo de negócio respeitar tal propriedade, o mesmo deve ser sempre capaz de completar um caso iniciado, ou seja, qualquer caso deve finalizar corretamente e cada tarefa deve necessariamente contribuir para a execução do processo de negócio (AALST, 1998b).

Numa WF-net, para um caso ser finalizado corretamente é necessário que para cada ficha (caso a ser tratado) no lugar de início, uma ficha seja produzida no lugar de término e qualquer outro lugar da rede esteja vazio. Entretanto, conforme mostra Aalst (1998b), o fato de cada processo modelado por uma WF-net ser *sound* não garante que o modelo global seja *sound*, pois erros de comunicação e possíveis situações de travamento introduzidas pelos elementos de comunicação podem aparecer. Além disto, ele também mostra que é possível ter uma IOWF-net globalmente *sound* mas não localmente *sound*.

Com o objetivo de solucionar estes problemas, Aalst (1998b) define que a noção global do critério de correção *Soundness* é baseada na representação de uma IOWF-net através de uma WF-net. Assim, Aalst (1998b) define a *U(IOWF-net)*, isto é, uma IOWF-net transformada em uma WF-net com um único processo. Em uma U(IOWF-net), todas os modelos de processos em WF-net são inseridas em um modelo de processo simples, considerando uma transição de início t_i e uma transição de término t_o (AALST, 1998b). Um lugar de início global i e um lugar de término global o precisam ser adicionados para respeitar a estrutura básica de uma WF-net simples e os elementos de comunicação assíncrona são mapeados em lugares (AALST, 1998b). A transformação da IOWF-net apresentada na Figura 27 em uma U(IOWF-net) pode ser vista na Figura 28, na qual foi adicionada o lugar de início i, as transições t_i e t_o, o lugar de fim o e os arcos (i, t_i), $(t_i, a0)$, $(t_i, p0)$, $(a6, t_o)$, $(p6, t_o)$ e (t_o, o).

O critério de correção Soundness no contexto da IOWF-net é dado pela Definição 22 (AALST, 1998b).

Definição 22 ***(Soundness para WorkFlow net Interorganizacional)*** *Uma IOWF-net é sound se, e somente se, for localmente e globalmente sound. Uma IOWF-net é localmente sound se, e somente se, cada uma de seus processos modelados por uma WF-net é sound. Uma IOWF-net é globalmente sound se, e somente se, a U(IOWF-net) é sound.*

Considerando a IOWF-net mostrada na Figura 27, os modelos de processos *AU* e *PC* em WF-nets são ambos *sound*. Entretanto, sua U(IOWF-net) apresentada na Figura 28 não é *sound*. Portanto, a IOWF-net não satisfaz o critério de correção *Soundness*. Tal critério não é satisfeito devido ao estado de *deadlock* alcançado quando a transição *too_late* da WF-net *PC* e as transições $prepare\_final\_version$ e $send\_final\_version$ da WF-net *AU* são disparadas. Ao realizar estes disparos, as mensagens *too_late* e $final\_version$ se entrelaçam, produzindo uma ficha no lugar $a5$ e duas mensagens que nunca serão recebidas (uma ficha no lugar *too_late* e uma ficha no lugar $final\_version$).

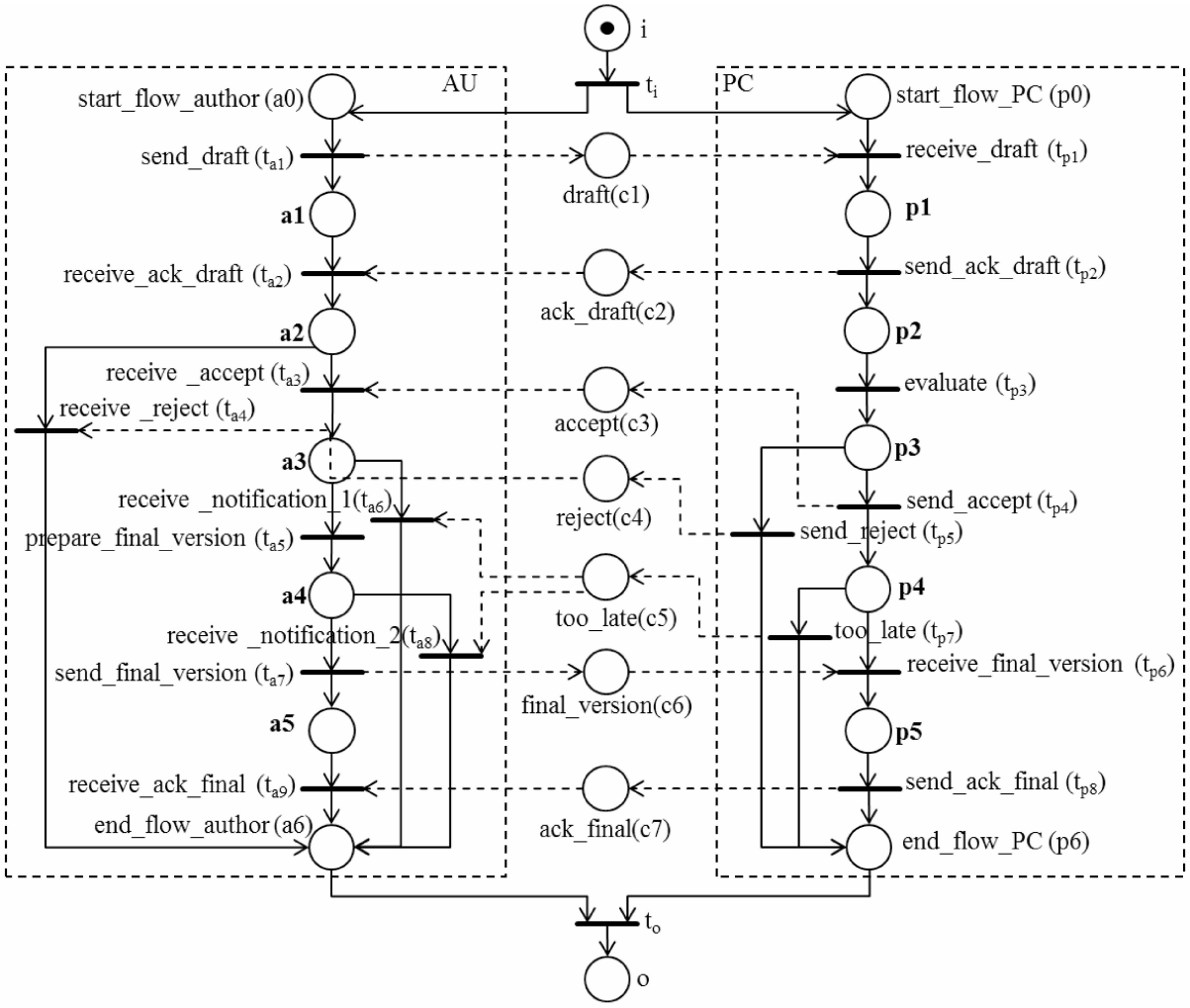


Figura 28 – A U(IOWF-net) para a IOWF-net mostrada na Figura 27 (AALST, 1998b).

2.4 Situações de *Deadlock* baseado em Sifão

O problema de *deadlock* é um assunto exaustivamente estudado em diversas áreas que envolvem comunicação, compartilhamento de informações e recursos (TANENBAUM, 2009). Uma situação de *deadlock* é caracterizada quando o fluxo dos processos é permanentemente impedido devido à falta de materiais, recursos e/ou informações (COFFMAN; ELPHICK; SHOSHANI, 1971).

Uma situação *deadlock* pode ocorrer em virtude da competição entre os processos concorrentes pela utilização de recursos compartilhados ou por falhas de sincronização de comunicação entre esses processos concorrentes (SILVA, 2014). Nos WfMS's, tal situação é causada por um erro de *design* no modelo do processo(MARUTA et al., 1998). A definição de *deadlock* é dada a seguir (BANASZAK; KROGH, 1990):

Definição 23 ***(Deadlock)*** *Um processo de negócio modelado por uma WF-net* $R = (P, T, Pre, Post)$ *está em situação de deadlock para uma dada marcação* M' *se, e somente se, nenhuma transição pode ser disparada. Formalmente,*

$$\forall t \in T, (M' \stackrel{t}{\nrightarrow}). \tag{20}$$

A presença de situações de *deadlock* em processos modelados por uma rede de Petri se deve à existência de estruturas particulares chamadas de *sifões*, também conhecidos como *deadlock* estrutural (BARKAOUI; ABDALLAH, 1995). Como estruturas especiais, os sifões estão relacionados com a vivacidade dos processos modelados por uma rede de Petri e têm sido amplamente utilizados na caracterização e na prevenção das situações de *deadlock* (ZHONG; LI, 2011).

Um sifão é um conjunto de lugares P' tal que o conjunto de transições de entrada de P' está contido no conjunto de transições de saída de P' (MURATA, 1989). Como existem mais saídas de fichas que entradas, o conjunto de lugares P' pode ficar livre de fichas podendo provocar uma situação de *deadlock* (BACALÁ JÚNIOR, 2003). Em particular, um sifão que não contém nenhuma ficha em uma determinada marcação M permanecerá sem fichas para qualquer marcação subsequente à M (TRICAS; MARTÍNEZ, 1995). A definição formal de um sifão é descrita a seguir (DAVID; ALLA, 2010):

Definição 24 ***(Sifão)*** *Seja* $R = (P, T, Pre, Post)$ *uma rede de Petri, um sifão é um conjunto de lugares* P' *não vazio de* P $(P' \subset P)$ *tal que* $\bullet P' \subseteq P' \bullet$.

Definição 25 ***(Sifão Básico)*** *Um sifão* P' *é dito básico se ele não puder ser expresso pela união de outros sifões.*

Definição 26 ***(Sifão Mínimo)*** *Um sifão* P' *é dito mínimo se, e somente se, nenhum outro sifão está contido nele como um subconjunto próprio.*

A fim de evitar o completo esvaziamento de fichas em um sifão, o mesmo deve conter pelo menos uma *trap*. De uma forma simplificada, uma *trap* é um conjunto de lugares P'' tal que o conjunto de transições de saída de P'' esteja contido no conjunto de transições de entrada de P'' (MURATA, 1989). Isto significa que se algum dos lugares da *trap* tem uma ficha, então sempre existirá uma ficha em algum dos lugares da *trap*, ou seja, o disparo das transições podem movimentá-las nos lugares da *trap* mas não podem removê-las. A definição de *trap* é dada a seguir (DAVID; ALLA, 2010):

Definição 27 ***(Trap)*** *Seja $R = (P, T, Pre, Post)$ uma rede de Petri, uma trap é um conjunto de lugares P'' não vazio de P ($P'' \subset P$) tal que $P''\bullet \subseteq \bullet P''$.*

Uma condição necessária para a existência de uma situação de *deadlock* numa rede de Petri é a presença de pelo menos um sifão vazio quando se considera o conjunto das marcações acessíveis (IORDACHE; MOODY; ANTSAKLIS, 2002). Além disto, uma condição necessária e suficiente para a vivacidade de uma rede de Petri marcada é que todos os sifões mínimos da rede devem possuir uma *trap* suficientemente marcada, garantindo assim que nenhuma situação de *deadlock* seja atingível (HACK, 1972).

Diversos algoritmos para detecção automática dos sifões em uma rede de Petri têm sido propostos por diferentes autores. Alguns são baseados em Matriz de Incidência (BOER; MURATA, 1994), desigualdades (EZPELETA; COUVREUR; SILVA, 1993), equações lógicas (KINUYAMA M. E MURATA, 1986), Programação Matemática (CHU; XIE, 1997) ou simples procedimentos de pesquisa de primeira ordem aplicados em diferentes combinações de lugares (JENG; PENG, 1996).

Para exemplificar a detecção de um sifão em uma rede de Petri, o algoritmo baseado em Matriz de Incidência proposto por Boer e Murata (1994) é descrito no Algoritmo 2. Para um melhor entendimento do algoritmo, um teorema e algumas definições apresentados em (BOER; MURATA, 1994) são necessárias. A seguir, a matriz de incidência de sinais (derivado da Matriz de Incidência), uma operação a ser utilizada no algoritmo e um teorema são apresentados.

Definição 28 ***(Matriz de Incidência de Sinais)*** *Para uma rede de Petri com n transições e m lugares, a matriz de incidência de sinais $D = [d_{ij}]$ é uma matriz $n \times m$, cujos valores iniciais são um dos seguintes:*

- $d_{ij} = +$ *se o lugar j é um lugar de saída da transição i;*
- $d_{ij} = -$ *se o lugar j é um lugar de entrada da transição i;*
- $d_{ij} = \pm$ *se o lugar j é lugar de entrada e de saída da transição i;*
- $d_{ij} = 0$ *em outro caso.*

Definição 29 ***(Operação $\oplus$)*** *A adição denotada por $\oplus$ é uma operação binária comutativa no conjunto de quatro elementos, $C_e = \{+, -, 0, \pm\}$, definida como:*

- $+ \oplus - = \pm$;
- $a \oplus a = a$, $\forall a \in C$;
- $\pm \oplus a = \pm$, $\forall a \in C$;
- $0 \oplus a = a$, $\forall a \in C$.

Algoritmo 2 Definição dos Sifões Básicos

Entrada: D: matriz de incidência de sinais, n: quantidade de transições, m: quantidade de lugares

Saída: todos os sifões básicos

1ª ETAPA: identifica os sifões ($LEAFs$)
Passo 1: Para cada coluna D_j ($j = 1, 2, \ldots, m$) na matriz de incidência de sinais D cujo lugar correspondente é chamado de $SEED_j$, faça o nível de recursão $r = 1$ e $X_{jr} = D_j$ e vá para o "Passo 2".
Passo 2: Se X_{jr} tem um valor + na c-ésima linha, procure uma coluna D_a que contenha um valor − ou ± na c-ésima linha. Se encontrar tal coluna, adicione-a ao vetor X_{jr} para obter o vetor $X_{j(r+1)}(X_{jr} \oplus D_a)$ contendo um valor ± na c-ésima linha (isto é, o valor + é neutralizado pelo ±). Repita este procedimento para todas as possíveis colunas de neutralização D_a encontradas. Ao fim deste passo, novos conjuntos $X_{j(r+1)}$ serão produzidos.
Passo 3: Incremente r em 1 e repita o "Passo 2" até que não haja mais nenhum valor + em cada $X_{jr} = D_1 \pm D_2 \pm \ldots \pm D_r$, ou que nenhuma coluna de neutralização seja encontrada. Qualquer X_{jr} sem nenhum valor + representa um sifão $\{p_1, p_2, \ldots, p_r\}$. Esses novos sifões ($LEAF_s$ de $PLANT_j$) serão entradas para a 2ª ETAPA do algoritmo.

2ª ETAPA: verifica se os sifões são mínimos
Quando, no "Passo 1", um novo candidato a sifão mínimo ($LEAF'$) é encontrado, duas condições são verificadas: (1) se no atual conjunto de sifões encontrados ($PLANT_j$) a partir de $SEED_j$ existe um conjunto de lugares ($LEAF$) tal que $LEAF'$ está contido nele ($LEAF' \subset LEAF$), então o conjunto de lugares $LEAF$ será sobreposto por $LEAF'$; e (2) se um subconjunto de $LEAF'$ já existe em $PLANT_j$, então $LEAF'$ não é adicionado à $PLANT_j$. Uma vez que todos m $PLANT_s$ ($PLANT_j, j = 1, 2, \ldots, m$) são gerados, a 3ª ETAPA é executada.

3ª ETAPA: verifica a existência de duplicatas
SB é inicializado com os sifões de $PLANT_1$ e cada $LEAF$ no próximo $PLANT_j$, $j = 2, 3, \ldots, m$, será adicionado à SB somente se o mesmo ainda não estiver contido em SB.

Teorema 1 *Um subconjunto de k lugares, $P' = \{p_1, p_2, \ldots, p_k\}$, em uma rede de Petri R é um sifão se, e somente se, a adição da k-ésima coluna do vetor da matriz de incidência de sinais de R ($D_1 \oplus D_2 \oplus \ldots \oplus D_k$) não contém nenhum valor +, onde D_j representa o vetor coluna correspondente ao lugar p_j com $j = 1, 2, \ldots, k$.*

A partir do Teorema 1, pode-se concluir que todos os possíveis sifões em uma rede de Petri R podem ser gerados encontrando-se todas as possíveis combinações de vetores colunas da matriz de incidência de sinais D de R cuja soma não contém nenhum valor +. De outro modo, um conjunto de lugares P' é um sifão se, e somente se, a adição de suas respectivas colunas em D não contém um valor +.

Alguns dos termos usados na descrição do Algoritmo 2 são definidos a seguir:

- *Operação de Neutralização de* $+$ é a operação usada para se obter $\pm$, ou seja, $+\oplus- = \pm$ ou $+ \oplus \pm = \pm$. Um valor $+$ é dito neutralizado pela adição de $-$ ou $\pm$;
- *Lugar-sifão mínimo com relação ao lugar* p é um sifão contendo um dado lugar p e um conjunto mínimo de lugares. (Note que um lugar-sifão mínimo não é necessariamente um sifão);
- $SEED_j$ é o lugar p_j correspondente à coluna D_j escolhida no passo 1 do algoritmo;
- $SPROUT$ é o processo recursivo descrito na 1ª ETAPA do algoritmo;
- $PLANT_j$ é o conjunto de sifões que são encontrados pelo $SPROUT$ a partir de $SEED_j$. O conjunto completo de sifões é obtido em $PLANT_j$ após o passo 3;
- $LEAF$ é um elemento de $PLANT$ que é candidato a um lugar-sifão mínimo.

O Algoritmo 2 usa um vetor X e dois conjuntos P e SB. $X = D_1 \oplus D_2 \oplus \ldots \oplus D_x$ corresponde à adição de x vetores colunas correspondentes aos r lugares em $P = \{p_1, p_2, \ldots, p_r\}$, e SB é um conjunto de sifões básicos gerados durante a execução do algoritmo. Índices e expoentes apropriados serão colocados em X e P durante a geração dos sifões.

O funcionamento do Algoritmo 2 é ilustrado no Anexo A.

2.5 Considerações Finais

Este capítulo apresentou os conceitos necessários para o entendimento do trabalho. Para tanto os conceitos de uma rede de Petri, de uma *WorkFlow net* e de uma *WorkFlow net* interorganizacional foram descritos. Além disto, extensões das redes de Petri, tais como rede Predicado-Transição, rede de Petri colorida, rede de Petri a objeto e rede de Petri possibilística, foram apresentadas. Por fim, a ocorrência e a detecção de situações de *deadlock* estrutural em uma rede de Petri foram explicadas.

Capítulo 3

Trabalhos Correlatos

A ideia inicial de suporte a processos de negócio é relatada na década de 70, quando o primeiro protótipo de um sistema de *workflow* foi desenvolvido por Zisman (1977) em sua tese de doutorado. A partir da década de 90, de acordo com Desel e Erwin (2000), os processos de negócio e seus projetos ganharam importância, de forma que a habilidade de simplificar um processo desse tipo, de modo eficiente e flexível, se torna o fator de sucesso mais difícil de ser alcançado pelas organizações atualmente. Além disto, nas últimas duas décadas, as organizações têm enfrentado um crescente aumento na incerteza acerca do ambiente de trabalho devido à velocidade das mudanças ambientais (URTASUN-ALONSO et al., 2012). Entre as causas dessas mudanças podem-se destacar as inovações tecnológicas, alterações na demanda, variedade das preferências dos clientes, forte concorrência com custos trabalhistas mais baixos, etc. A fim de suportar as incertezas, a flexibilidade tem sido considerada como um dos elementos mais úteis e essenciais para lidar com a continuidade dos sistemas de *workflows* frente às mudanças do ambiente de trabalho (CANTAMESSA; CAPELLO, 2009) (PUNDIR; MISHRA; GANAPATHY, 2013).

O conceito de flexibilidade foi inicialmente aplicado no contexto dos sistemas de manufatura. Com isto, diversos tipos de sistemas de manufatura flexíveis (FMS's) têm sido propostos a fim de contemplar as incertezas relacionadas ao mercado, governos, regulamentos, tecnologia, etc (WARD; DURAY, 2000). Nos processos de negócios, as abordagens propostas que consideram o conceito de flexibilidade se baseiam nos tipos de flexibilidade propostos por Schonenberg et al. (2008b).

Assim sendo, na Seção 3.1, algumas das abordagens propostas referente aos FMS's são apresentadas. Na Seção 3.2, as principais contribuições existentes na literatura abordando o tema de "flexibilidade nos processos de negócios" são expostas. Nas Seções 3.3 e 3.4, as abordagens propostas para contornar/evitar situações de *deadlock* e agregar o conceito de cancelamento ao processo de negócio são descritas. Juntamente a cada abordagem citada, as limitações existentes em cada uma são apresentadas.

3.1 Flexibilidade Aplicada no Contexto dos Sistemas de Manufatura

Os sistemas de manufatura flexíveis (FMS's) diferem entre si de acordo com os diversos tipos de técnicas e ferramentas requeridas para se incorporar a flexibilidade desejada ao sistema (MISHRA; PUNDIR; GANAPATHY, 2014). A fim de aprimorar tanto o desempenho operacional de processos de fabricação industrial, diversos autores têm proposto vários *frameworks* para analisar, implementar e/ou gerenciar os FMS's tais como (ZAIRI; ZOUARI; PITRAC, 2007) (HERLE et al., 2010) (KHALID; YUSOF; SABUDIN, 2012) (URTASUN-ALONSO et al., 2012) (MAHMUDY; MARIAN; LUONG, 2014).

Zairi, Zouari e Pitrac (2007) propuseram um modelo baseado na rede de Petri colorida para especificação, verificação e supervisão dos FMS's. A especificação formal permite a descrição dos níveis da arquitetura e da produção do sistema e uma ferramenta traduz automaticamente esta especificação em um modelo de rede de Petri colorida.

Uma estratégia de controle capaz de adaptar o escalonamento das tarefas de forma a aumentar a flexibilidade e a produtividade do sistema sem que novos equipamentos sejam requeridos foi definida por Herle et al. (2010). Tal estratégia é baseada no paradigma de processamento paralelo combinado com uma otimização do roteamento das peças entre as diferentes estações do sistema de manufatura.

Khalid, Yusof e Sabudin (2012) e Mahmudy, Marian e Luong (2014) propuseram algoritmos genéticos a fim de aprimorarem o escalonamento de tarefas nos FMS's. Khalid, Yusof e Sabudin (2012) levaram em consideração o problema de manutenção, ou seja, certas máquinas durante um determinado período de tempo não podem ser disponibilizadas para a produção. Sob outra perspectiva, Mahmudy, Marian e Luong (2014) consideraram o problema de escolher simultaneamente os subprodutos e as máquinas necessárias para a produção do produto final de forma a otimizar o planejamento da produção.

Singh, Oberoi e Ahuja (2012) utilizaram uma técnica de análise hierárquica do processo, do inglês *Analytical Hierarchy Process (AHP)*, para avaliar a flexibilidade estratégica nos FMS's. Urtasun-Alonso et al. (2012) analisaram o relacionamento entre os FMS's e as práticas do gerenciamento de recursos humanos a fim de verificar as diferenças e os ganhos em performance quando comparado com um sistema de manufatura tradicional.

Outra vertente destas pesquisas é a utilização de técnicas de inteligência artificial (IA), tais como os conceitos de conjuntos nebulosos (ZADEH, 1965), lógica *fuzzy* (ZADEH, 1988) e lógica possibilística (ZADEH, 1999; DUBOIS; PRADE, 2009) quando informações imprecisas dever ser levadas em conta. Chuu (2006), Das e Caprihan (2007) e Mahdavi, Azar e Bagherpour (2009) fizeram uso da lógica *fuzzy* para avaliarem a flexibilidade nos FMS's. Chuu (2006) propuseram um modelo de tomada de decisão auxiliado por um conjunto *fuzzy* com diferentes conjuntos de termos linguísticos. Das e Caprihan (2007), através de uma base de regras *fuzzy*, avaliaram a flexibilidade baseado em uma visão

sistêmica dos parâmetros que afetam os diferentes tipos de flexibilidade. E, Mahdavi, Azar e Bagherpour (2009) empregaram a lógica *fuzzy* nos problemas de escalonamento dinâmico de tarefas de forma a solucionar o problema de seleção do roteamento mais adequado considerando um determinado processamento e as peças envolvidas.

Desde o trabalho pioneiro de (LOONEY, 1988), vários autores, da comunidade de rede de Petri e de IA, têm propostos diferentes tipos de rede de Petri nebulosas (RPN). Sob o mesmo nome, estes modelos são baseados na rede de Petri e em diferentes noções de IA.

Cardoso et al. em (CARDOSO; VALETTE; DUBOIS, 1989) (CARDOSO; VALETTE; DUBOIS, 1999) (CARDOSO, 1999) e Murata et al. em (MURATA, 1996) (MURATA; SUZUKI; SHATZ, 1999) utilizaram a rede de Petri juntamente com a lógica possibilística e conjuntos nebulosos para incluir uma noção de incerteza nos processos da manufatura permitindo uma maior flexibilidade na especificação dos processos de produção e modelos mais compactos.

Em (CARDOSO, 1999) e (CARDOSO; VALETTE; DUBOIS, 1989), a marcação imprecisa, definida por distribuições de possibilidades, e a noção de duração *fuzzy* de atividades dos processos, caracterizada por dados *fuzzy*, foram definidas na rede de Petri a objetos para "relaxar" as relações de sincronização e/ou descrever um conjunto de alternativas possíveis. A noção de marcação imprecisa foi usada para modelar a tarefa de supervisão do ambiente físico e descrever informação incompleta sobre o estado corrente do sistema. A noção de duração *fuzzy* foi associada a cada transição com o intuito de supervisionar o tempo de execução das atividades.

Em (CARDOSO; VALETTE; DUBOIS, 1999), uma rede de Petri possibilística, a qual combina uma rede de Petri a objetos com a lógica possibilística, foi apresentada. Nesse caso, os estados foram indicados por distribuições de possibilidade e a evolução do estado do processo foi vista como uma atualização dessas distribuições por meio dos "tipos de disparo" das transições. A principal característica é permitir um tipo de raciocínio sobre aspectos da incerteza num sistema dinâmico a eventos discretos.

Em (MURATA, 1996) e (MURATA; SUZUKI; SHATZ, 1999), a noção explícita de tempo foi incluída em uma rede de Petri. A principal característica foi a introdução de quatro funções teóricas do conjunto fuzzy relacionado ao tempo. Elas são: *fuzzy timestamp*, *fuzzy enabling time*, *fuzzy occurrence time* e *fuzzy delay*, todas capturando a incerteza temporal.

A limitação destes trabalhos se encontra no fato deles tratarem essencialmente dos sistemas de manufatura, os quais contêm características próprias em relação aos sistemas de gerenciamento de processos de negócio. Algumas características próprias relativas aos sistemas de manufatura são: os recursos são em sua maioria máquinas; a variação do tempo de execução de uma atividade é relativamente baixa; máquinas podem ter réplicas idênticas, ou seja, sua substituição não impacta na execução do processo; e a flexibilidade aplicada aos sistemas de manufatura se concentra principalmente no problema de

escalonamento de tarefas.

3.2 Flexibilidade Aplicada no Contexto dos Processos de Negócios

Levando em conta as abordagens propostas na manufatura para permitir uma flexibilização de certas características representadas pelo modelo do processo, outros pesquisadores têm proposto diferentes abordagens considerando o conceito de flexibilidade que poderia/deveria existir nos modelos dos processos de negócios. A revisão de algumas das abordagens relevantes que considera um certo nível de flexibilidade nos modelos dos processos é apresentada a seguir.

Em (CUGOLA et al., 1995) e (CUGOLA et al., 1996), um modelo baseado em lógica temporal e máquina de estados finitos para capturar e tolerar desvios no processo durante sua execução foi definido. Para os autores, um processo é correto se todas as restrições dadas pelo conjunto das máquinas de estados são verificadas. Em particular, dois tipos de transição foram criados: as normais e as externas, as quais dependem de requisições do usuário para indicar um comportamento anormal. O problema com tal abordagem é que o modelo do processo foi dado através de uma forma declarativa em vez de um grafo representando todo o processo o qual permitiria analisar certas propriedades do mesmo como, por exemplo, a propriedade *Soundness* no caso da WF-net. Outro problema é a necessidade de modelar explicitamente os cenários alternativos correspondentes aos comportamentos anormais. A consequência é geralmente o aumento da complexidade do conjunto de restrições e do modelo do processo.

Khomyakov e Bider (2001) propuseram uma abordagem baseada numa visão orientada a estado. Considerado como uma trajetória no espaço de todos os estados possíveis (fluxo de estados), o processo é composto por um conjunto de regras de planejamento classificadas em três categorias - obrigações, recomendações e proibições - responsáveis pela distribuição das atividades e pelo controle do fluxo de execução. O estado final desejado é alcançado de acordo com as mudanças introduzidas por cada atividade.

Em (CHARFI; MEZINI, 2007), a linguagem AO4BPEL foi proposta com o objetivo de fornecer configurabilidade através da decomposição do modelo do processo de negócio em um núcleo abstrato e regras de negócios modularizadas. A programação orientada a Aspectos, do inglês Aspect-Oriented Programming (AOP), foi empregada com o propósito de desviar as instâncias do modelo do processo de acordo com os resultados das regras de avaliação. No entanto, os *pointcuts*, onde as regras são intercaladas com a definição do modelo do processo, precisam ser especificados em tempo de *design*.

Em (KUMAR; YAO, 2012), o modelo de um processo em execução - incluindo o fluxo de controle, recursos e dados - foi definido durante a instanciação do mesmo através da aplicação das regras de negócios à um modelo de processo genérico. A flexibilidade de

tal caso é limitada ao tempo de *design*, dado que nenhuma alteração pode ser realizada durante a sua execução.

Em (BARROS; DECKER, 2006), o controle da instância do modelo do processo é encaminhado a diferentes atores, os quais podem adicionar subprocessos e refinar as restrições. Já Charoy, Guabtni e Faura (2006), a fim de lidarem com a questão da coordenação das atividades de cooperação, apresentam um conjunto de requisitos - incluindo flexibilidade e dinamicidade - para os WfMS's que visam suportar *workflows* interorganizacionais. Em ambas abordagens o modelo do processo é criado em tempo de execução pelos usuários considerando a técnica *late modeling*; porém nenhum controle a cerca das propriedades do modelo é realizado.

Em (PESIC; AALST, 2006) e (PESIC et al., 2007), uma abordagem declarativa baseada em restrições é proposta para permitir que as instâncias do modelo do processo se desviem do comportamento original em tempo de execução através da reconfiguração das restrições e do mapeamento das definições do processo. De modo similar, em (AALST; PESIC; SCHONENBERG, 2009), a execução do modelo do processo, especificado implicitamente por restrições, é realizada de modo que qualquer comportamento é permitido desde que não seja explicitamente proibido. O problema destas abordagens é o fato de que a sequência das atividades é limitada apenas à satisfação das restrições pré-determinadas.

Adams et al. em (ADAMS et al., 2005) e (ADAMS et al., 2006) definiram um repertório extensivo de subprocessos associados à cada tarefa onde, durante o tempo de execução, um deles é selecionado de acordo com o contexto particular da instância em execução. Além disto, novos subprocessos podem ser inseridos durante a execução da instância do modelo do processo de forma a suportar trocas dinâmicas e a evolução do mesmo. Cada subprocesso é representado por uma estrutura nomeada como *worklet*. Estendendo tal abordagem ao tratamento das exceções, um repertório extensivo de subprocessos chamados de *exlets* é definido em (ADAMS, 2007) e (ADAMS et al., 2007). Tanto a proposta dos *worklets* quanto a dos *exlets* são implementadas na nova linguagem de modelagem de processos de negócio chamada YAWL, proposta em (ADAMS, 2010) e (AALST; HOFSTEDE, 2005). os *worklets* são suportados através das chamados "*ripple-down rules*" (SCHEFFER, 1996), as quais são baseadas nas informações referente ao contexto e orientam o usuário no momento de selecionar um subprocesso durante a execução do modelo do processo. Uma limitação deste método é a explosão de estados que pode ocorrer quando um processo de automatização é gerado a partir de restrições. Além disto, a propriedade *Soundness* pode não ser garantida.

Zazworka et al. em (ZAZWORKA; BASILI; SHULL, 2009) e (ZAZWORKA et al., 2010) definiram uma abordagem para detectar, durante o desenvolvimento do modelo do processo, os desvios e as inconsistências ocasionadas entre o planejado e o executado. De modo semelhante, mas considerando o modelo do processo em execução, Mohammed, Redouane e Bernard (2007) propuseram uma abordagem para detectar os desvios através

da coexistência de dois modelos. Permanentemente comparados e analisados para detectar os desvios, o primeiro modelo corresponde ao comportamento esperado do processo e o segundo é construído dinamicamente através da observação das ações dos atores humanos. A limitação destas metodologias é que ambas apenas realizam a detecção dos desvios e/ou inconsistências e nenhum tipo de tratamento para as possíveis ocorrências dos mesmos é definido.

Thompson e Torabi (2009) propuseram um tipo de modelo essencialmente baseado em regras para detectar inconsistências (estados inconsistentes com o processo) e aceitar possíveis desvios (transições inconsistentes entre as atividades). A limitação dessa metodologia é relacionada ao modelo do processo que pode ser visto como um simples conjunto de restrições sem uma estrutura real do modelo do processo que pode ser analisado a partir do ponto de vista das propriedades básicas.

Outra tendência destas pesquisas é verificar se o comportamento observado registrado durante a execução de uma instância do modelo do processo, em um arquivo de *log*[1] de eventos, corresponde ao comportamento esperado. As abordagens propostas se diferem em relação ao tipo de não conformidade analisada, o qual pode ser, por exemplo, o tempo gasto (GIBLIN; MüLLER; PFITZMANN, 2006) (THULLNER et al., 2011) (GOMEZ-LOPEZ; GASCA; RINDERLE-MA, 2013), os dados processados (HALLE; VILLEMAIRE, 2008) (BORREGO; BARBA, 2014), os recursos utilizados (SANTOS et al., 2012), o fluxo de controle (AALST et al., 2006) (ROZINAT; AALST, 2008) (MUNOZ-GAMA, 2010) ou tempo/recurso/fluxo (BASIN et al., 2011) (SEBAHI, 2012) (LEONI; AALST, 2013) (LY et al., 2015). O problema destas abordagens é que todas realizam apenas a verificação sem nenhum tipo de tratamento em tempo de execução para evitar ou corrigir as inconsistências detectadas.

Uma alternativa para lidar com a flexibilidade nos processos de negócios são as abordagens baseadas em raciocínio com incertezas. Como exemplificado anteriormente na manufatura, a utilização de tal raciocínio permite uma representação do conhecimento mais abrangente de acordo com o ambiente de execução real. Cîmpan e Oquendo (2000) por meio de conjuntos *fuzzy* e distribuições de possibilidade propuseram uma abordagem para detectar inconsistências. As informações precisas bem como as imprecisas são tratadas pelo uso da lógica *fuzzy*. Logo, a teoria dos conjuntos *fuzzy* é usada para representar tais informações enquanto a teoria possibilística é usada no raciocínio sobre a informações imprecisas através das regras *fuzzy*. Uma limitação desta abordagem é que eles não realizam nenhum tipo de tratamento quando uma inconsistência é detectada.

[1] Um *log*, em termos gerais, é um registro de acontecimentos ou de atividades. Em se tratando de computadores, é um registro gerado por um serviço ou aplicativo específico (às vezes pelo próprio sistema operacional).

3.3 Situações de *Deadlock* Estruturais

Observando as limitações das abordagens anteriormente citadas relacionadas à flexibilidade nos processos de negócios, uma das mais manifestadas refere-se a falta de um modelo estruturado que permita um estudo que assegura que o processo modelado sempre apresente boas propriedades. Entretanto, mesmo que o modelo do processo permita assegurar tais propriedades, de acordo com Fahland et al. (2011), mais de 50% dos processos de negócio em execução não respeitam a propriedade *Soundness*, a qual garante a ausência das situações de *deadlock*. Observa-se que uma situação de *deadlock*, do ponto de vista da execução de um modelo de processo, corresponde a uma situação indesejada obtida após a execução de uma sequência de eventos responsável por conduzir o estado do modelo do processo a uma situação de inconsistência global. Assim sendo, diversas abordagens foram propostas definindo novos métodos de análise, durante a fase de modelagem, ou métodos para detecção, durante a execução do modelo do processo, para identificar e evitar estados de *deadlock* ocasionados, por exemplo, pela estrutura do modelo, pela sincronização entre processos ou pela política de alocação de recursos necessários para a execução do mesmo.

Uma grande parte das pesquisas devotadas ao problema de *deadlock*, seja nos processos de manufatura ou nos de negócio, têm-se centrado nas situações de *deadlock* ocasionadas pela alocação de recursos nos sistemas de: comunicação (TANG et al., 2012) (MOHANTY; KUMARA, 2013), *workflow* (WANG; LU, 2013b) (WANG; LU, 2013a), manufatura flexível (LIU; LI; ZHOU, 2013) (BARUWA; PIERA; GUASCH, 2015), entre outros. Além disto, uma variedade de políticas de controle para evitar *deadlocks* nos sistemas de manufatura automatizados baseadas na rede de Petri têm sido propostas (EZPELETA; COLOM; MARTÍNEZ, 1995) (HUANG et al., 2001) (LI; ZHOU, 2004) (UZAM; ZHOU, 2007) (AHMAD; HUANG; WANG, 2011) (CHEN; LI, 2011) (CHEN; LI; ZHOU, 2012) (HUANG; PAN; ZHOU, 2012) (LI et al., 2012) (LUO et al., 2015). A partir do ponto de vista técnico, a maioria destas políticas de controle propostas para solucionar os problemas de *deadlocks* são desenvolvidas via análise do espaço de estados ou da análise estrutural da rede de Petri. Ao se considerar a análise do espaço de estados, as políticas de controle levam em geral à soluções com uma alta complexidade computacional e ao problema de explosão do número de estados a serem processados. Entretanto, ao evitar o problema de explosão de estados, através da análise estrutural, possíveis estados proibidos podem ser omitidos durante a realização da mesma (LIU et al., 2013).

Uma situação de *deadlock* em uma rede de Petri é caracterizada pela presença de sifões livres de fichas. Considerando isto, diversos algoritmos para detectá-los bem como métodos eficientes para a sintetização de supervisores, os quais forçam à marcação do sifão a nunca se tornar completamente vazia, têm sido propostos (BARKAOUI; ABDALLAH, 1995) (CHU; XIE, 1997) (MARUTA et al., 1998) (SADIQ; ORLOWSKA, 2000) (IORDACHE; MOODY; ANTSAKLIS, 2002) (AWAD; PUHLMANN, 2008) (CORDONE; PIRODDI, 2013) (LIU; LI; ZHOU, 2013) (CHEN; LI; ZHOU, 2014) (HU; ZHOU, 2015)

(LIU et al., 2015). Todos estes trabalhos são baseados em um tipo de transformação do modelo do processo, de modo que, nenhum deles pode ser aplicado durante a fase de execução.

Em se tratando de trabalhos com *workflows* interorganizacionais, Aalst (1998b), Silva et al. (2013) e Passos e Julia (2014) propuseram, cada um, um modo de tratamento dos problemas de *deadlocks* presentes nos modelos de processos. Aalst (1998b), por meio de alterações manuais, desvia o fluxo das atividades do modelo do processo para uma sequência que finalize a execução do mesmo sem qualquer ocorrência de situações indesejadas. Silva et al. (2013) alteram o modelo do processo inserindo lugares responsáveis por controlar os sifões que se esvaziem. E Passos e Julia (2014) definem um método baseado na análise das árvores de prova da Lógica Linear para detectar os cenários livre de *deadlock*.

Como limitação destas abordagens, pode-se observar que no trabalho de Aalst (AALST, 1998b) não utiliza um procedimento sistemático. Silva et al. (2013) alteram o modelo do processo, inserindo um lugar de controle. E Passos e Julia (2014) apenas detectam os cenários livres de *deadlock*.

Ao se observar as limitações das abordagens propostas, quando uma situação de *deadlock* não pode ser completamente removida na fase de *design*, torna-se interessante flexibilizar o modelo de forma que, durante a execução do mesmo, roteiros alternativos possam ser levados em consideração. Desta forma, as situações de *deadlock* podem ser contornadas permitindo manter o modelo do processo em funcionamento de modo a assegurar as boas propriedades.

3.4 Noção de Cancelamento aplicada aos Processos de Negócios

Mesmo que o modelo do processo tenha uma estrutura que permita a verificação de que o mesmo apresenta boas propriedades e, consequentemente, as possíveis situações de *deadlock* podem ser evitadas, quando se considera a noção de cancelamento de atividade nos WfMS's, simples questões como alcançabilidade podem ser indecidíveis e, por consequência, assegurar as boas propriedades se torna inviável (DUFOURD; JANČAR; SCHNOEBELEN, 1999). Neste contexto, o cancelamento de atividade pode ser requisitado mediante um pedido por parte do cliente (por exemplo, cancelamento de uma compra) ou casos de exceções (como por exemplo, informação insuficiente para processar a compra). Em geral, um cancelamento resulta em duas possíveis consequências: desativação de algumas das atividades escalonadas da execução prevista ou finalização das atividades que estão atualmente em execução (WYNN et al., 2009).

Um cancelamento, como no caso das situações de *deadlock*, resultam em desvios nas sequências de ações especificadas e previstas nos modelos de processos. Caso possível, quando um cancelamento ocorre, a consistência do modelo do processo deve ser mantida.

Assim sendo, um comportamento flexível do modelo do processo corresponde à capacidade de lidar não apenas com situações de desvios simples e/ou ocasionais, mas com situações de mudanças extremas em relação às sequências de ações planejadas no modelo.

A rede *reset* proposta inicialmente por Araki e Kasami (1976a), com o objetivo de incorporar explicitamente a uma rede de Petri o conceito de cancelamento através do uso dos arcos *reset*, têm uma aplicação natural na modelagem dos processos de negócios e nos WfMS's (WYNN et al., 2009). Os arcos *reset* conectam lugares à uma transição t_r com o objetivo de remover todas as fichas destes lugares quando a transição dispara.

Nessa rede, os possíveis cancelamentos de atividades precisam ser modelados explicitamente. Entretanto, a introdução dos arcos *reset* aumenta a complexidade da rede quando comparada à rede de Petri definida por Murata (1989), pois: (1) como todas as fichas, e não apenas uma, são removidas ao disparar a transição t_r, os invariantes de lugar não são mantidos para essa rede; (2) se um lugar não contém fichas no momento do disparo da transição com arcos *reset*, a ação de remover as fichas se torna ineficaz para tal lugar; e (3) um arco *reset* pode afetar vários lugares em toda a rede (ou seja, o seu efeito é global), ao contrário dos arcos "normais" de uma transição, os quais só podem influenciar os lugares conectados a ele (isto é, o seu efeito é local) (WYNN et al., 2009).

Embora o cancelamento via rede *reset* possa ser útil na modelagem de processos, ele complica até certo ponto a verificação dos modelos de *workflow* (VERBEEK; AALST; HOFSTEDE, 2007). Pois, uma rede *reset* com mais de dois arcos *reset* perde a decidibilidade da propriedade de alcançabilidade e limitabilidade, consequentemente, da propriedade *Soundness* (ARAKI; KASAMI, 1976b) (DUFOURD; FINKEL; SCHNOEBELEN, 1998) (AALST; PESIC; SCHONENBERG, 2009) (HOFSTEDE et al., 2010).

Levando em consideração o problema dos modelos com arcos *reset*, Wynn et al. (2009) propuseram um conjunto de regras de redução para a WF-net com arcos *reset* baseadas nas regras de redução já definidas para rede de Petri sem arcos *reset*. Normalmente, uma regra de redução diminui o número de elementos de um modelo abstraindo um conjunto de transições e/ou lugares num único elemento (macro-transição ou macro-lugar) enquanto preserva algumas das propriedades do modelo (VERBEEK et al., 2010). Entretanto, um arco *reset* pode nunca ser totalmente abstraído da rede *reset*, ou seja, se uma rede contém arcos *reset* não é possível obter uma outra reduzida sem qualquer arco *reset* e, se a quantidade dos mesmos for superior a dois, a indecidibilidade das propriedades permanecem (WYNN et al., 2009). Além disto, reduções do modelo podem também introduzir transições, as quais não correspondem aos eventos reais. Observa-se que está longe de ser trivial expressar o comportamento desejado sem arcos *reset* dado que, para remover tais arcos, todas as possíveis marcações devem ser consideradas, o que pode tornar o modelo do processo completamente ilegível e intratável (AALST; PESIC; SCHONENBERG, 2009).

3.5 Considerações Finais do Capítulo

Ao se observar as abordagens descritas anteriormente, nota-se que a flexibilidade de um modo geral, tanto do ponto de vista da manufatura quanto dos processos de negócios, trata da capacidade de se desviar, até um certo ponto, dos roteiros predefinidos (sequências de ações/atividades/tarefas/operações) sem que, durante a execução do modelo do processo, uma situação de total inconsistência seja alcançada. Estes desvios têm como objetivo adaptar o modelo do processo às situações imprevistas que dificilmente são modeladas de modo completamente explícitas dado que, por normalmente serem numerosas, um problema de explosão combinatória de possibilidades em relação à quantidade de roteiros alternativos a serem considerados pode ser alcançado.

Levando em consideração as abordagens citadas, a grande maioria das soluções desenvolvidas ou detecta os desvios, ou tentam, na fase de análise, considerar um número maior de roteiros de execução alternativos. Entretanto, a maior parte das abordagens não tem modelos de processos estruturados de forma a permitir uma análise formal das propriedades do mesmo. Além disto, desvios devido a situações de *deadlock* e cancelamento, nem sempre podem ser evitados assegurando as propriedades do processo.

Considerando as limitações citadas anteriormente, nos Capítulos 4 e 5 foram definidas, respectivamente, a WF-net possibilística e a IOWF-net possibilística com o objetivo de permitir o correto tratamento das situações de *deadlock* e cancelamento de forma a assegurar as propriedades do processo. Além destas situações, certos desvios em relação ao modelo do processo serão permitidos sem que um estado de inconsistência seja alcançado.

Capítulo 4

WorkFlow net Possibilística

Este capítulo redefine e, posteriormente, aplica a WF-net possibilística em problemas específicos relacionados aos processos de negócio. Em (REZENDE, 2013), a WF-net possibilística foi definida com o intuito de modelar a recuperação do estado de modelo do processo na fase de execução quando possíveis eventos não são recebidos. No caso, a ação associada a uma atividade é executada somente se o disparo realizado é certo, ou seja, somente se a informação relacionada ao disparo for precisa. Entretanto, para contemplar todas as mudanças de comportamento do modelo do processo de acordo com a qualidade das informações recebidas durante a sua execução, a definição da WF-net possibilística deve ser alterada com o objetivo de considerar uma ação para cada possível mudança.

Para que uma WF-net possibilística considere o padrão de mudança "*Parallelize Process Fragments*" [1] definido por Weber, Reichert e Rinderle-Ma (2008) e que todas as atividades escalonadas para execução tenham a ação correspondente ao disparo certo executada, o algoritmo de defuzzificação utilizado na fase de recuperação do estado do modelo do processo em (REZENDE, 2013) é modificado. Além disto, a noção de cancelamento, apontada por Aalst et al. (2003) como um importante mecanismo nos WfMS's, quando a execução de algumas atividades pode levar a finalização de outras em determinadas circunstâncias é incorporado à WF-net possibilística através da definição de uma região de cancelamento.

A estruturação deste capítulo é dada da seguinte forma: a WF-net possibilística é redefinida na Seção 4.1. Na Seção 4.2, a WF-net possibilística é aplicada aos processos de negócio em execução com roteiros puramente sequenciais com o objetivo de permitir uma paralelização parcial na ordenação das atividades. Neste caso, a execução de novas atividades pode ser iniciada mesmo que as suas precedentes estejam parcialmente executadas, ou seja, não concluídas. Ao fazer isto, possíveis desvios nas práticas de trabalho relacionados à ordenação das atividades são suportados de forma a manter consistente o estado corrente do modelo na fase de monitoramento com o estado real do processo em

[1] "*Parallelize Process Fragments*": paralelizar fragmentos do modelo do processo puramente sequencial em tempo de execução (WEBER; REICHERT; RINDERLE-MA, 2008).

execução. Por fim, na Seção 4.3, um procedimento baseado em regiões de cancelamento e nos disparos incertos é definido a fim de incorporar a noção de cancelamento à WF-net possibilística de forma a considerar o cancelamento da execução de certas atividades e, em alguns casos, do processo. Tal procedimento preserva a decidibilidade das propriedades do modelo tais como alcançabilidade e, consequentemente, *Soundness*.

4.1 Definições da WF-net Possibilística

Uma WF-net Possibilística é baseada na combinação da estrutura de roteamento da WF-net com a marcação imprecisa e o disparo incerto de uma rede de Petri possibilística. Esta combinação produz um tipo de modelo que é capaz de lidar com os problemas de desvio (não respeito de uma sequência de eventos prevista no modelo de especificação do processo (THOMPSON; TORABI, 2009)) na fase de execução dos modelos dos processos de negócio. Neste modelo, as transições representam as alterações de estado do processo e cada ocorrência de um evento durante a execução do mesmo é associado às transições como uma variável booleana[2]. Tal variável é vista essencialmente como um valor externo correspondente a uma mensagem recebida a partir de uma atividade (ou enviada a uma atividade). Os valores internos dos atributos das fichas também são consideradas para habilitar transições. A definição da WF-net possibilística é dada a seguir:

Definição 30 ***(WorkFlow net Possibilística)*** *Uma WorkFlow net Possibilística é uma nônupla*

$$WP =< C_{aso}, P, T, V_f, Pre, Post, A_{tc}, A_{ta}, M_0 > \tag{21}$$

onde:

- C_{aso} *representa a classe de objeto "Caso", onde um conjunto de atributos é definido e organizado em uma hierarquia;*
- T *é um conjunto finito de transições;*
- P *é um conjunto finito de lugares todos do tipo "Caso", onde:*
  - *existe um único lugar de início* $i \in P$ *tal que* $\bullet i = \phi$*;*
  - *existe um único lugar de término* $o \in P$ *tal que* $o\bullet = \phi$*;*
  - *todo nó* $x \in P \cup T$ *está em um caminho de entre os lugares* i *e* o*.*
- V_f *é um conjunto finito de variáveis formais do tipo "Caso";*
- $Pre : P \times T \rightarrow P(V_f^*)$ *é a função lugar precedente que, a cada arco de entrada de uma transição* $t \in T$*, faz corresponder uma soma formal de elementos de* V_f*;*

[2] Um tipo de dado primitivo que possui dois valores, que podem ser considerados como 0 ou 1 (falso ou verdadeiro).

- $Post : P \times T \rightarrow P(V^*)$ *é a função lugar seguinte que, a cada arco de saída de uma transição* $t \in T$*, faz corresponder uma soma formal de elementos de* V_f*;*

- A_{tc} *é uma aplicação que associa uma função de autorização* (η) *a cada transição* $t \in T$*, a qual pode envolver tanto eventos externos quanto um conjunto de atributos dos objetos por meio das variáveis formais* V_f *associadas aos arcos de entrada de* t*. Tal função se comporta como uma condição extra no disparo de* t *e é estruturada da seguinte forma:*

$$\eta_{x_1,...,x_n} : T \longrightarrow \{verdadeira, incerta, falsa\} \tag{22}$$

  onde:

  - T *é um conjunto finito de transições;*
  - $x_1, ..., x_n$ *são as variáveis formais associadas aos arcos de entrada de* t*;*
  - $verdadeira$*,* $incerta$ *e* $falsa$ *são as possíveis interpretações obtidas a partir da análise da função de autorização* (η) *durante a execução do modelo do processo. Elas são representadas, respectivamente, pelos símbolos* V*,* U *e* F *no modelo do processo.*

- A_{ta} *é uma aplicação que associa um conjunto de ações a cada transição* $t \in T$*, a qual envolve os atributos das variáveis formais* V_f *associadas aos arcos de entrada e saída de* t*. Este conjunto, composto por quatro tipos de ações especificadas na Tabela 1, permite modificar os atributos dos objetos de acordo com o tipo do disparo e é estruturada da seguinte forma:*

$$v.atr \leftarrow EXPR(x_1, x_2, \ldots, x_n) \tag{23}$$

  onde:

  - v *é uma variável formal associada ao arco de saída de* t*;*
  - atr *é um dos atributos pertencente à* v*;*
  - $EXPR$ *é uma expressão cujas variáveis usadas na mesma são as variáveis associadas ao(s) arco(s) de entrada de* t*.*

- M_0 *é a marcação inicial que associa, ao lugar* i*, uma soma formal dos objetos do tipo "Caso" (n-uplas de instâncias da classe "Caso").*

A fim de tratar os desvios, a noção de incerteza é inserida. Esta noção exprime o fato de que a existência de um objeto é conhecida, mas a sua localização é imprecisa (isto é, a localização do objeto é descrita em termos de um conjunto de lugares). A partir disso, a marcação da WF-net possibilística pode ser precisa ou imprecisa permitindo assim a existência de dois tipos de disparos: certo ou incerto (*pseudo-disparo*). As definições dos tipos de marcação e de disparo são apresentadas a seguir:

Definição 31 ***(Tipos de Marcação)*** *A localização dos objetos em uma WF-net Possibilística permite a representação de uma:*

- *Marcação precisa: cada objeto do tipo Caso está localizado em apenas um lugar, ou seja, se a possibilidade do objeto b estar no lugar p é 1 ($\pi_b(p) = 1$), então a possibilidade do objeto b estar nos outros lugares do modelo diferentes de p é 0 ($\forall p_i \neq p$, $\pi_b(p_i) = 0$);*

- *Marcação imprecisa: um mesmo objeto do tipo Caso está em dois ou mais lugares com possibilidade igual a 1 ($\pi_b(p_i) = 1$ e $\exists p_j \neq p_i \mid \pi_b(p_j) = 1$).*

Definição 32 ***(Tipos de Disparo)*** *O disparo de uma transição $t \in T$ numa WF-net possibilística pode ser de dois tipos:*

- *Disparo Certo: corresponde ao disparo de uma transição t conforme a definição original da rede de Petri. Assim sendo, o término do disparo, neste caso a finalização, coincide com o início do mesmo. Neste tipo de disparo a localização de todas as instâncias dos objetos envolvidos deve ser precisa. Como consequência, a nova marcação do sistema também é precisa e obtida a partir da remoção das instâncias dos objetos do tipo Caso dos lugares de entrada de t e da adição de novas instâncias de objetos do tipo Caso nos lugares de saída de t;*

- *Disparo Incerto (ou pseudo-disparo): neste tipo de disparo considera-se apenas o início do disparo uma vez que não existe informação suficiente para confirmar se os eventos esperados associados à transição t realmente ocorreram ou não, ou seja, as instâncias dos objetos do tipo Caso são adicionados aos lugares de saída de t mas não são removidos dos lugares de entrada da mesma.*

Na WF-net possibilística, o disparo de uma transição depende, além da marcação nos lugares de entrada da transição, da função de autorização η. Assim sendo, as regras de disparo que definem o comportamento de uma WF-net possibilística são descritas abaixo:

Definição 33 ***(Regras de Disparo)*** *Seja t uma transição e $s_1, ..., s_n$ uma possível substituição dos valores de atributos dos objetos $b_1, ..., b_n$ nas variáveis formais $x_1, ..., x_n$ associadas aos arcos de entrada de t, as regras de disparo são:*

- *se t não está habilitada mas sua interpretação é verdadeira, então um alarme é ativado dado que isto corresponde a uma situação proibida;*

- *se t está habilitada por uma marcação precisa e sua interpretação é verdadeira, então t é disparada com certeza, isto é, as instâncias dos objetos são removidos do lugares de entrada de t, a ação A^c_{ta} associada a t é executada e novas instâncias dos objetos são produzidas nos lugares de saída de t;*

- *se t está habilitada por uma marcação precisa ou imprecisa e sua interpretação é incerta, então t é disparada com incerteza, isto é, as instâncias dos objetos não são removidos do lugares de entrada de t, a ação A^i_{ta} associada a t é executada e novas instâncias dos objetos são produzidas nos lugares de saída de t;*

- *se t está habilitada por uma marcação imprecisa e sua interpretação é verdadeira, então um algoritmo de defuzzificação é chamado para computar uma nova distribuição de possibilidade para as instâncias dos objetos pertencentes à rede. Tal algoritmo é responsável por finalizar ou cancelar os disparos incertos relacionados à transição t. Neste caso, quando um disparo incerto de uma transição t' é finalizado, a ação A^f_{ta} é executada e, quando é cancelado, a ação A^l_{ta} que é executada. Após a execução do algoritmo, se t estiver habilitada por uma marcação precisa, então t será disparada com certeza.*

Observando as regras de disparo, nota-se que as quatro ações que compõem a aplicação A_{ta} são executadas dependendo do estado do modelo do processo. A Tabela 1 especifica, para cada ação, o tipo do disparo o qual a mesma é associada e o estado atual do modelo do processo no momento da sua execução. Por exemplo, a ação A^c_{ta} é executa se, e somente se, o disparo da transição for certo, isto é, a função de autorização é avaliada como verdadeira e a marcação é precisa. Ao fazer isto, possíveis tratamentos de dados podem ser realizados como, por exemplo, um *rollback*[3] nas ações já realizadas quando o pseudo-disparo de uma transição é cancelado.

Tabela 1 – Especificação das quatro ações que compõem a aplicação A_{ta}.

Ação	Tipo do Disparo	Estado do modelo do processo
A^c_{ta}	Disparo Certo	marcação precisa e $\eta(t)$ = verdadeiro
A^i_{ta}	Disparo Incerto	$\eta(t)$ = incerta
A^f_{ta}	Disparo Incerto Finalizado	algoritmo de defuzzificação em execução
A^l_{ta}	Disparo Incerto Cancelado	algoritmo de defuzzificação em execução

O diagrama da Figura 29 apresenta o algoritmo do "jogador" de WF-net possibilística. Este algoritmo descreve o motor de inferência responsável pela execução do modelo em WF-net possibilística, isto é, responsável por deslocar as fichas de modo a respeitar as regras de disparo das transições descritas na Definição 33.

Para elaborar um modelo de um processo de negócio considerando uma WF-net possibilística, sete etapas devem ser realizadas. Tais etapas, definidas a seguir, têm particularidades as quais são especificadas no momento da definição do modelo do processo de acordo com o tipo de problema a ser tratado.

[3] *rollback* é um comando que desfaz os efeitos dos comandos de uma transação

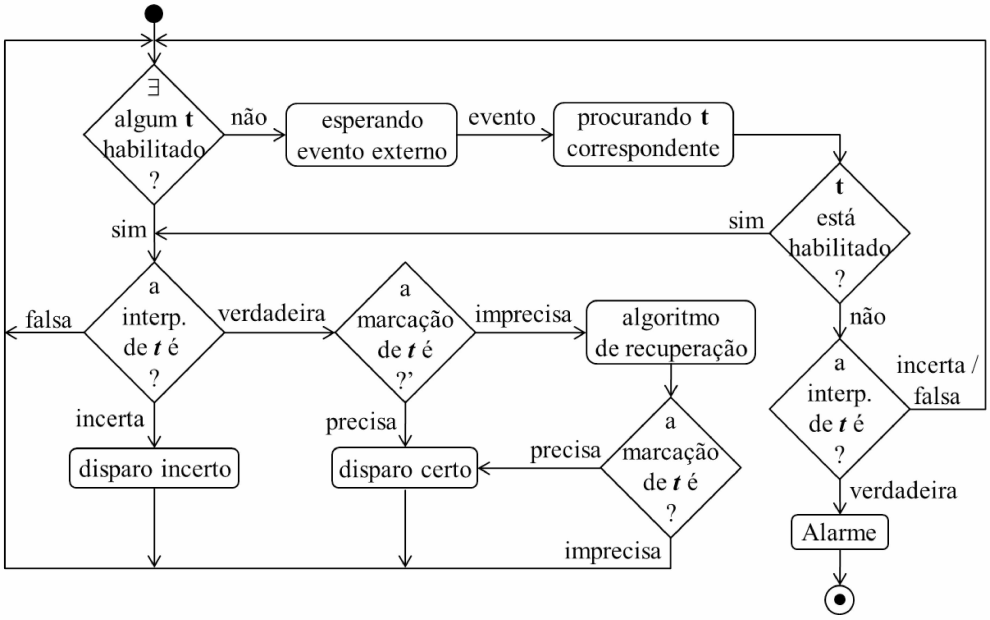


Figura 29 – Algoritmo do "jogador" de WF-net possibilística.

Definição 34 ***(Etapas de Modelagem)*** *As etapas necessárias para modelar um processo de negócio considerando uma WF-net possibilística são:*

1. *definir os lugares P, as transições T e arcos de entrada e saída de cada transição $t \in T$;*
2. *definir os atributos e métodos da classe C_{aso};*
3. *definir e associar as variáveis formais aos arcos de entrada e saída de cada transição $t \in T$;*
4. *definir o subconjunto de transições T^* que terão as funções de autorização produzindo como resultado às possíveis interpretações: $verdadeira$, $incerta$ ou $falsa$;*
5. *definir as funções de autorização. Isto é, para cada transição $t \in T$ faça:*
   a) *definir a(s) condição(ões) necessária(s) para que a função de autorização associada a t possa ser avaliada como $verdadeira$ durante a execução do modelo;*
   b) *se $t \in T^*$, definir a(s) condição(ões) necessária(s) para que a função de autorização associada a t possa ser avaliada como $incerta$ durante a execução do modelo;*
   c) *definir a(s) condição(ões) necessária(s) para que a função de autorização associada a t possa ser avaliada como $falsa$ durante a execução do modelo;*
6. *definir o conjunto de ações A^c_{ta}, A^i_{ta}, A^f_{ta} e A^l_{ta} para cada transição $t \in T$;*
7. *executar o modelo do processo considerando o "jogador" de WF-net possibilística mostrado na Figura 29 e o algoritmo de recuperação definido para o problema. Em*

geral, o algoritmo considerado é o mesmo aplicado à rede de Petri possibilística definida por Cardoso (1999), o qual foi descrito na subseção 2.1.4.5.1. Entretanto, dependendo da característica do problema a ser tratado, algumas alterações no mesmo podem ser requeridas.

4.2 Paralelização Parcial de Processos Sequenciais

A ordenação explícita das atividades nos *workflows* tradicionais desconsidera quaisquer outras situações, isto é, torna rígida a execução do processo de modo que interferências manuais se tornam necessárias para adequá-los às mudanças nas práticas de trabalho durante a execução real. Com o intuito de incorporar a flexibilidade de *design* no modelo a fim de evitar tais interferências, a WF-net possibilística é usada para modelar o processo de forma a considerar o padrão de mudança "*Parallelize Process Fragments*" definido por Weber, Reichert e Rinderle-Ma (2008) permitindo, assim, uma certa paralelização na ordem de execução das atividades. Assim sendo, as possíveis mudanças nas práticas de trabalho relacionadas à ordenação das atividades são suportadas de forma a assegurar a consistência do estado corrente do modelo na fase de execução com o estado real do processo em execução.

Com o objetivo de obrigar a execução das ações A^c_{ta} associadas às atividades, o algoritmo de recuperação utilizado em (REZENDE, 2013) e descrito no Algoritmo 1 é modificado. Esta modificação é devido à necessidade do modelo do processo retornar ao estado certo sem que qualquer disparo incerto seja finalizado, ou seja, apenas cancelamentos são permitidos dado que a finalização tem como característica a não execução da ação A^c_{ta} associada à transição pseudo-disparada.

O algoritmo de defuzzificação modificado é descrito no Algoritmo 3. A transição t indicada na lista de "*Entrada*" refere-se à transição habilitada por uma marcação imprecisa com a função de autorização avaliada como *verdadeira*, isto é, uma informação recebida de modo que a função de autorização associada a t seja avaliada como verdadeira possibilitando, assim, a dedução do estado real do modelo do processo. Por fim, a variável LL, usada no decorrer do algoritmo, refere-se a uma lista de lugares.

Considerando as etapas de modelagem especificadas na Definição 34, as etapas 4, 6 e 7 são detalhadas a fim de especificar as particularidades do problema. Na Etapa 4, o subconjunto de transições T^* é composto por um subconjunto de T ($T^* \subset T$) ou por todas as transições $t \in T$, isto é, $T^* \subseteq T$. A interpretação incerta definida para cada $t \in T^*$ através da função de autorização é a responsável por permitir a paralelização parcial do modelo.

Na etapa 6, a ação A^f_{ta} é desconsiderada para este problema, dado que nenhum disparo incerto é finalizado, apenas cancelado. Assim sendo, o conjunto de ações é composto apenas pelas ações A^c_{ta}, A^i_{ta}, e A^l_{ta}. Por fim, na etapa 7, o algoritmo de defuzzificação a

ser considerado é o Algoritmo 3. Considerando isto e o "jogador" de WF-net possibilística mostrado na Figura 29, a execução de um modelo de processo representado por uma WF-net possibilística tem o seguinte comportamento:

- se a função de autorização associada à transição t é avaliada como *verdadeira* e t está habilitada por uma marcação precisa, t é disparada com certeza e a ação A^c_{ta} associada a t é executada;

- se a função de autorização associada à transição t é avaliada como *incerta*, t é pseudo-disparada e a ação A^i_{ta} associada a t é executada;

- se a função de autorização associada à transição t é avaliada como *verdadeira* e t está habilitada por uma marcação imprecisa, o algoritmo de recuperação é chamado para cancelar todos os disparos incertos realizados nas transições posteriores a t e consequentemente executar todas as ações A^l_{ta} associadas às transições que tiveram o disparo incerto cancelado. Caso o estado do modelo do processo seja certo, ou seja, a marcação é precisa, a transição t poderá ser disparada com certeza e a ação A^c_{ta} associada a t será executada; caso contrário, a incerteza a cerca do estado do modelo do processo será mantida até que uma nova informação permita recuperar o estado certo do mesmo.

Note-se que quando a(s) condição(ões) necessária(s) para avaliar uma função de autorização como *incerta* durante a execução do modelo não é(são) definida(s), tal função de autorização nunca poderá ser avaliada como *incerta*.

Algoritmo 3 Defuzzificação Parcial

Entrada: WF-net com posicionamento impreciso dos objetos nos lugares, transição t
Saída: WF-net com um reposicionamento dos objetos nos lugares

1: **if** t foi pseudo-disparada anteriormente **then**
2: $LL \leftarrow (t\bullet)$
3: cancelar o disparo de t
4: **else**
5: $LL \leftarrow \phi$
6: **end if**
7: **while** $LL \neq \phi$ **do**
8: $p \leftarrow$ LL[0]
9: **while** $\exists\ t_j \in (p\bullet) \mid t_j$ foi pseudo-disparada anteriormente **do**
10: $LL \leftarrow LL \cup (t_j\bullet)$
11: cancelar o disparo de t_j
12: **end while**
13: **end while**

Com o propósito de ilustrar uma paralelização parcial durante a execução de um modelo de processo, um processo de encomenda puramente sequencial apresentado em (WESKE, 2007) é utilizado:

> This process features a sequence of activities, where the first activity to store the order is preceded by a start event. After the order is stored, the inventory is checked. This version of the business process rules that the shipment is prepared only after the invoice is sent and the funds are received. Finally, the goods are shipped and the process terminates (WESKE, 2007) p. 113.

> Tradução: Esse processo apresenta uma sequência de atividades, onde a primeira atividade é responsável por armazenar o pedido. Depois que o pedido é armazenado, o estoque é verificado. Esta versão do processo de negócios determina que o pedido é preparado somente depois que a fatura é enviada e o pagamento é recebido. Finalmente, os produtos são enviados e o processo termina.

A Figura 30 mostra um modelo do processo em WF-net possibilística. $< c >$ é um objeto pertencente à classe "Encomenda", bem como as variáveis x e todos os lugares do modelo. Tal classe é composta pelos seguintes atributos:

- $sgBA1$, $sgEA1$, $sgBA2$, $sgEA2$, $sgBA3$, $sgEA3$, $sgBA4$, $sgEA4$, $sgBA5$, $sgEA5$, $sgBA6$, $sgEA6$, $sgBA7$, $sgEA7$: booleano;
- $t_{max_{EA3}}$, $t_{max_{BA4}}$, $t_{max_{EA4}}$, $t_{max_{BA5}}$, $t_{max_{EA5}}$, $t_{max_{BA6}}$, $t_{max_{EA6}}$: constante inteira.

Observa-se que outros possíveis atributos específicos para o processo de encomenda podem ser definidos para este exemplo; entretanto a não representação deles não interfere no entendimento da abordagem. O significado dos atributos declarados anteriormente para a classe "Encomenda" é:

- os atributos $sgBA1$, $sgBA2$, $sgBA3$, $sgBA4$, $sgBA5$, $sgBA6$ e $sgBA7$, quando verdadeiros, representam a ocorrência de um evento indicando, respectivamente, o início das seguintes atividades: *store order*, *check inventory*, *send invoice*, *receive funds*, *prepare shipment*, *ship goods* e *archive*;
- os atributos $sgEA1$, $sgEA2$, $sgEA3$, $sgEA4$, $sgEA5$, $sgEA6$ e $sgEA7$, quando verdadeiros, representam a ocorrência de um evento indicando, respectivamente, o término das seguintes atividades: *store order*, *check inventory*, *send invoice*, *receive funds*, *prepare shipment*, *ship goods* e *archive*;
- os atributos $t_{max_{BA4}}$, $t_{max_{BA5}}$ e $t_{max_{BA6}}$ representam o tempo máximo permitido numa escala de tempo associada ao modelo do processo para iniciar, respectivamente, a execução das seguintes atividades: *receive funds*, *prepare shipment* e *ship goods*;

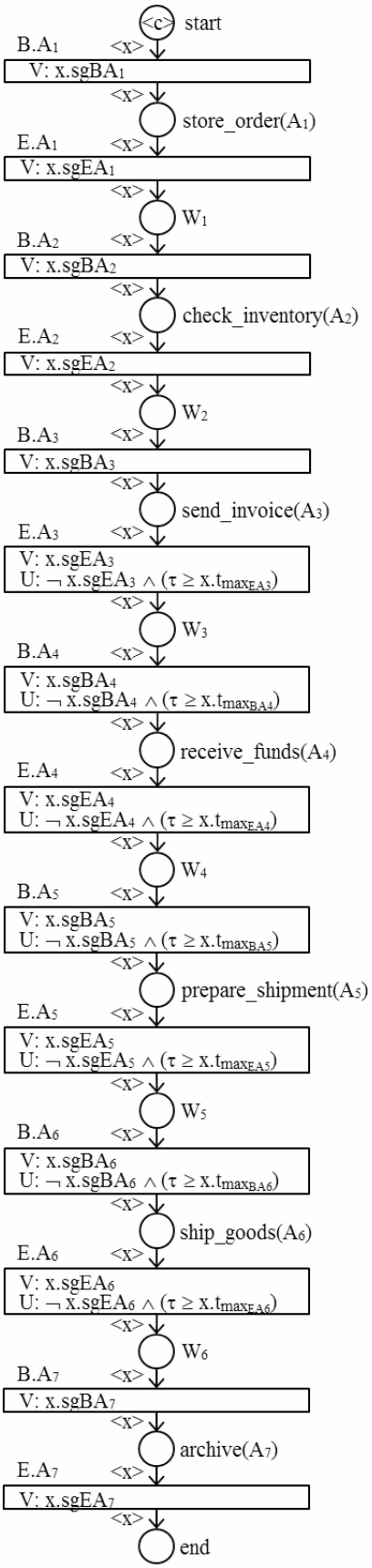


Figura 30 – Modelo de um processo de encomenda em WF-net possibilística.

- os atributos $t_{max_{EA3}}$, $t_{max_{EA4}}$, $t_{max_{EA5}}$ e $t_{max_{EA6}}$ representam o tempo máximo permitido numa escala de tempo associada ao modelo do processo para iniciar, respectivamente, a conclusão das seguintes atividades: *send invoice*, *receive funds*, *prepare shipment* e *ship goods*.

Como definido anteriormente na Seção 4.1, cada transição tem uma interpretação e um conjunto de ações associados a ela. Entretanto, a fim de simplificar o modelo, as ações, as interpretações falsas e, em alguns casos, as interpretações incertas não são descritas.

Com o intuito de exemplificar o uso das diferentes ações associadas às transições, as ações A^c_{ta}, A^i_{ta} e A^l_{ta} são definidas para a transição $E.A_5$ a título de ilustração. Para isto, considere a variável *itens* que é responsável por indicar quais itens ainda faltam para serem separados e a variável *status* que indica quantos por cento da atividade em questão já foi concluída. Abaixo, as três ações associadas à transição $E.A_5$ são descritas.

$$A_{ta} = \begin{cases} A^c_{ta} = exclui(itens) \wedge status = 100\% \\ A^i_{ta} = atualiza(itens, dados) \\ A^l_{ta} = atualiza(itens, dados) \wedge atualiza(status, atividade, porcentagem) \end{cases}$$

A não representação da interpretação *incerta* numa transição t significa que a função de autorização nunca pode ser avaliada como *incerta* para t, apenas como *verdadeira* ou $falsa$. Em contrapartida, a função de autorização será avaliada como $falsa$, mesmo que a interpretação $falsa$ esteja omitida na descrição do modelo, sempre que as interpretações *verdadeira* e *incerta* não são válidas para a transição em questão.

Ao observar o modelo, sob uma política cautelosa, o comportamento esperado do processo é executar todas as atividades de forma sequencial, isto é, uma atividade só poderá iniciar sua execução quando a execução da atividade imediatamente anterior a ela se finalizar. Por exemplo, a atividade *prepare shipment* (A_5) poderá iniciar sua execução somente quando o pagamento da encomenda for confirmado (atividade A_4). Em consequência, os casos baseados neste modelo podem sofrer com longos tempos de processamento, resultando numa insatisfação por parte do cliente.

Levando em consideração o tempo de processamento e, de certa forma, desconsiderando o rígido sequenciamento das atividades, um possível comportamento anormal pode ocorrer. Tal comportamento pode ser caracterizado, por exemplo, pelo início das atividades A_5 e A_6 antes da conclusão da atividade A_4, permitindo assim um menor tempo de processamento. Neste caso, alguns disparos incertos ocorrem, aumentando a imprecisão sobre alguns objetos e, consequentemente, permitindo uma pseudo paralelização durante a execução das atividades especificadas inicialmente de modo puramente sequencial no modelo do processo.

A fim de ilustrar a execução de um possível comportamento anormal, considere as seguintes informações:

- o elemento τ representa o tempo corrente;

- os atributos, os quais representam os tempo máximos permitidos, são inicializados com os respectivos valores:
  - $t_{max_{EA3}} \leftarrow 24$;
  - $t_{max_{BA4}} \leftarrow 25$;
  - $t_{max_{EA4}} \leftarrow 49$;
  - $t_{max_{BA5}} \leftarrow 50$;
  - $t_{max_{EA5}} \leftarrow 74$;
  - $t_{max_{BA6}} \leftarrow 75$;
  - $t_{max_{EA6}} \leftarrow 90$.

- os atributos, os quais representam a chegada de um evento, se tornam verdadeiros nas seguintes datas:
  - $sgBA_1$ quando $\tau = 0$;
  - $sgEA_1$ quando $\tau = 4$;
  - $sgBA_2$ quando $\tau = 7$;
  - $sgEA_2$ quando $\tau = 19$;
  - $sgBA_3$ quando $\tau = 20$;
  - $sgEA_3$ quando $\tau = 23$;
  - $sgBA_4$ quando $\tau = 24$;
  - $sgEA_4$ quando $\tau = 85$;
  - $sgBA_5$ quando $\tau = 86$;
  - $sgEA_5$ quando $\tau = 87$;
  - $sgBA_6$ quando $\tau = 88$;
  - $sgEA_6$ quando $\tau = 89$;
  - $sgBA_7$ quando $\tau = 90$;
  - $sgEA_7$ quando $\tau = 95$.

- as durações mínimas necessárias para executar as seguintes atividades A_1, A_2, A_3, A_4, A_5, A_6 e A_7 são, respectivamente, 3, 8, 3, 15, 15, 8 e 5 unidades de tempo.

- as durações máximas para executar as seguintes atividades A_1, A_2, A_3, A_4, A_5, A_6 e A_7 são, respectivamente, 5, 15, 5, 25, 25, 15 e 10 unidades de tempo.

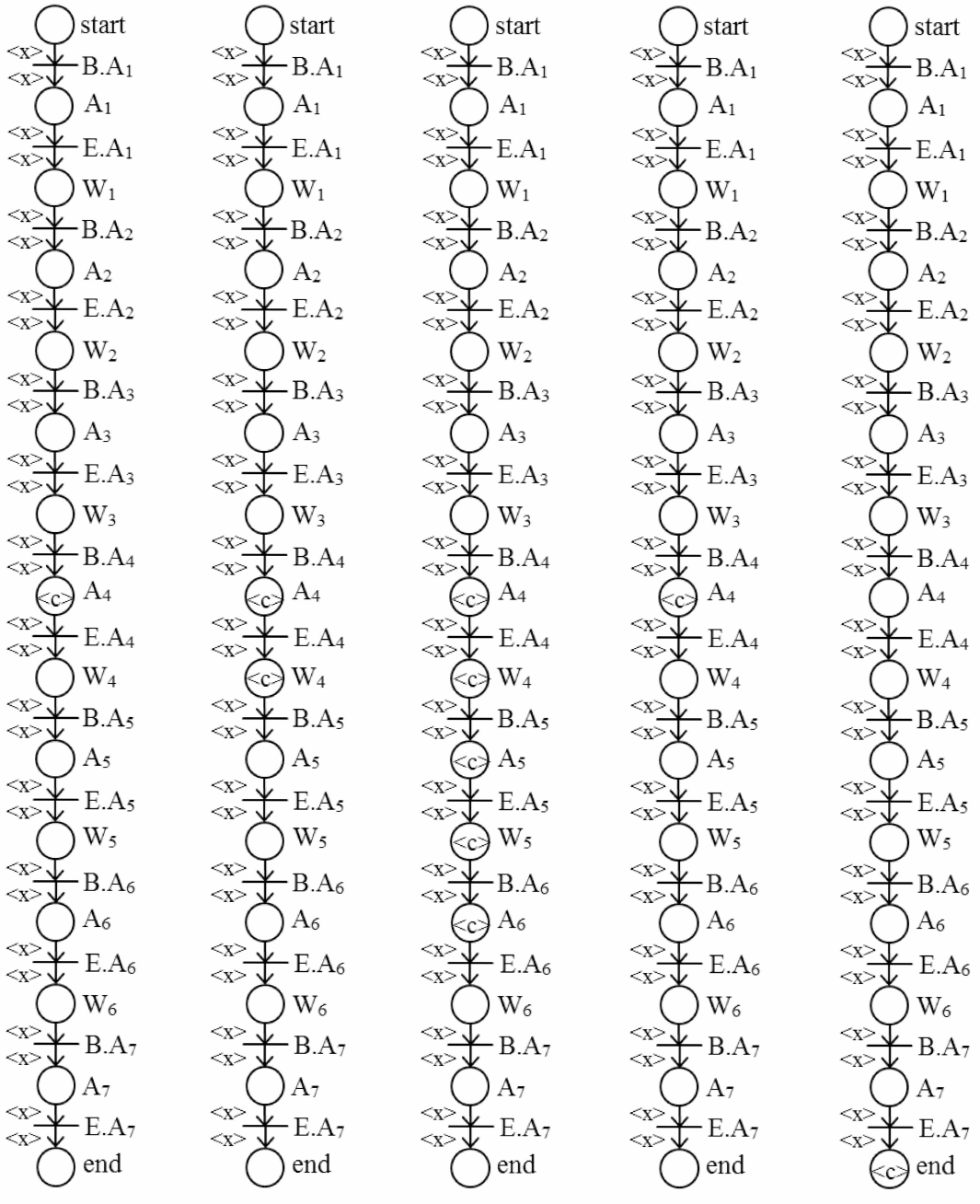
Levando-se em conta as informações acima descritas, o modelo em WF-net possibilística descrito na Figura 30 tem seu comportamento dinâmico simulado de acordo com o "jogador" apresentado na Figura 29, ou seja, disparos incertos e marcações imprecisas são consideradas. A simulação é apresentada em seguida:

- as transições $B.A_1$, $E.A_1$, $B.A_2$, $E.A_2$, $B.A_3$, $E.A_3$ e $B.A_4$ são disparadas com certeza quando o tempo corrente τ é igual aos respectivos valores: $\tau = 0$, $\tau = 4$, $\tau = 7$, $\tau = 19$, $\tau = 20$, $\tau = 23$ e $\tau = 24$. Isto ocorre pois a função de autorização de cada transição mencionada anteriormente é avaliada como *verdadeira* quando o tempo corrente atinge tais valores mencionados, ou seja, a condição relacionada à interpretação V definida em cada função de autorização é validada. Consequentemente, a marcação produzida é sempre precisa e, caso definidas, as ações $A^c_{B.A_1}$, $A^c_{E.A_1}$, $A^c_{B.A_2}$, $A^c_{E.A_2}$, $A^c_{B.A_3}$, $A^c_{E.A_3}$ e $A^c_{B.A_4}$ são executadas. O estado do modelo do processo resultante após o disparo desta sequência de transições é mostrado na Figura 31a (com $\tau = 24$);

- no tempo corrente $\tau = 49$, a confirmação sobre o pagamento da encomenda ainda não foi verificada, ou seja, o atributo $sgEA_4$ continua igual a $falso$. Levando em consideração que a transição $E.A_4$ está habilitada por uma marcação precisa e que

a condição ($\neg\, c.sgEA_4 \wedge \tau \geqslant 49$) relacionada à interpretação U definida na função de autorização (η) associada a $E.A_4$ é validada, isto é, $\eta_{<c>}(E.A_4) = incerta$, então $E.A_4$ é pseudo-disparada e, caso definida, a ação $A^i_{E.A_4}$ é executada. Neste disparo, uma nova instância do objeto $< c >$ é produzida no lugar W_4 sem que o objeto $< c >$ localizado em A_4 seja removido (Figura 31b);

- quando o tempo corrente alcança os seguintes valores $\tau = 50$, $\tau = 74$ e $\tau = 75$, as transições $B.A_5$, $E.A_5$ e $B.A_6$ são pseudo-disparadas dado que elas estão habilitadas por uma marcação imprecisa e a função de autorização de cada transição



(a) $\tau = 24$ (b) $\tau = 49$ (c) $\tau = 75$ (d) $\tau = 85$ (e) $\tau = 95$

Figura 31 – Resultados da simulação - Paralelização Parcial.

mencionada anteriormente é avaliada como *incerta*, ou seja, a condição relacionada à interpretação U definida em cada função de autorização é validada. Além disto, caso definidas, as ações $A^i_{B.A_5}$, $A^i_{E.A_5}$ e $A^i_{B.A_6}$ são executadas. Esta sequência de disparos incertos ocorre porque, até o tempo corrente, a confirmação sobre o pagamento da encomenda ainda não foi verificada e, na prática, para não paralisar o processamento da encomenda, as atividades seguintes são geralmente iniciadas. O estado do modelo do processo resultante após o disparo desta sequência de transições é mostrado na Figura 31c;

- no tempo corrente $\tau = 85$, a confirmação sobre o pagamento da encomenda é recebida, ou seja, o atributo $sgEA_4 = verdade$. Consequentemente, a condição $(c.sgEA_4)$ relacionada à interpretação V definida na função de autorização (η) associada à transição $E.A_4$ é validada, isto é, $\eta_{<c>}(E.A_4) = verdadeira$. Levando em consideração que a transição $E.A_4$ está habilitada por uma marcação imprecisa, dado que as atividades A_5 e A_6 já foram iniciadas a fim de evitar longos tempos de processamento, o algoritmo de recuperação descrito no Algoritmo 3 é chamado para cancelar os disparos incertos realizados nas transições seguintes a $E.A_4$. Durante a execução do algoritmo de defuzzificação, caso definidas, as ações $A^l_{E.A_4}$, $A^l_{B.A_5}$, $A^l_{E.A5}$ e $A^l_{B.A_6}$ associadas às transições que tiveram o disparo incerto cancelado são executadas. Após a execução do algoritmo de recuperação, a transição $E.A_4$ pode ser disparada com certeza, dado que está habilitada por uma marcação precisa (Figura 31d) e a função de autorização associada a ela é avaliada como *verdadeira*. É importante notar que se a marcação não for precisa após o processamento do Algoritmo 3, a transição $E.A_4$ não pode ser disparada com certeza e a instância do modelo do processo deve esperar por um novo evento, o qual permita que o estado do mesmo retorne a uma marcação precisa;

- Finalmente, quando o tempo corrente τ é igual aos respectivos valores $\tau = 86$, $\tau = 87$, $\tau = 88$, $\tau = 89$, $\tau = 90$ e $\tau = 95$, as transições $B.A_5$, $E.A_5$, $B.A_6$, $E.A_6$, $B.A_7$ e $E.A_7$ são disparadas com certeza e, caso definidas, as ações $A^c_{B.A_5}$, $A^c_{E.A_5}$, $A^c_{B.A_6}$, $A^c_{E.A_6}$, $A^c_{B.A_7}$ e $A^c_{E.A_7}$ são executadas. Isto ocorre pois a marcação é precisa e a função de autorização de cada transição mencionada anteriormente é avaliada como *verdadeira*, ou seja, a condição relacionada à interpretação V associada a cada função de autorização é validada. O estado do modelo do processo resultante após o disparo desta sequência de transições é mostrado na Figura 31e.

A distribuição de possibilidade relacionada às instâncias do objeto $< c >$ em função do tempo relativas à simulação anteriormente descrita é mostrada na Figura 32 (as linhas grossas representam uma possibilidade igual a 1(um) e as linhas tracejadas uma possibilidade igual a 0(zero)). Note que as atividades A_5 e A_6, após a execução do algoritmo de recuperação, são concluídas logo em sequência pois as mesmas já foram pseudo-iniciadas

anteriormente pelos seus respectivos responsáveis. Além disso, observando as distribuições de possibilidades, é fácil verificar que as atividades A_4, A_5 e A_6 foram pseudo-paralelizadas durante o período de tempo $[49, 85]$, mesmo que no modelo do processo elas foram descritas de modo puramente sequencial.

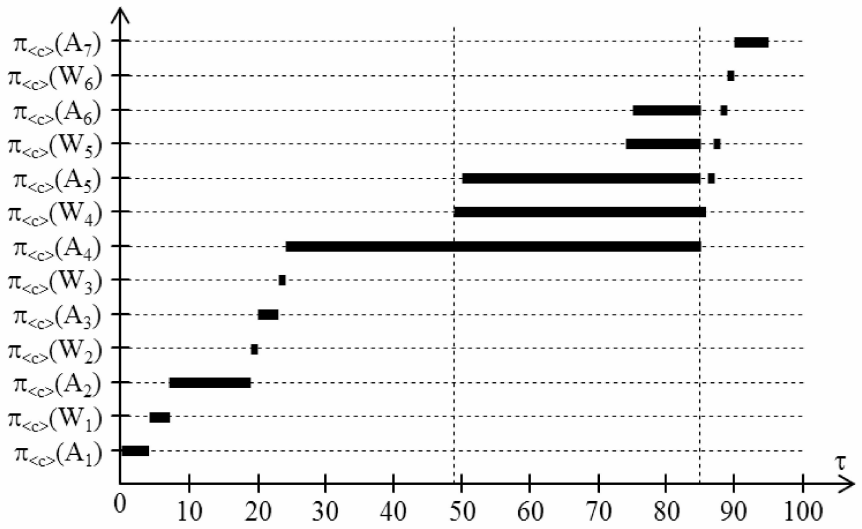
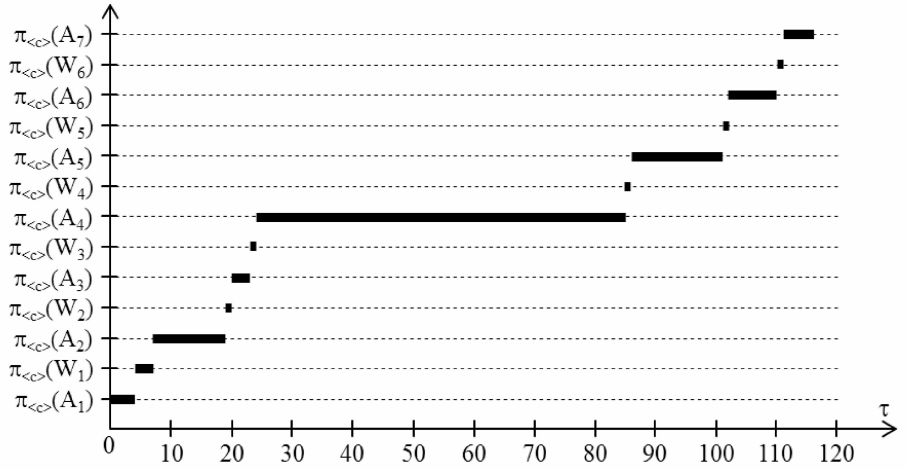


Figura 32 – Distribuições de possibilidade relacionadas aos locais do objeto $< c >$ considerando os disparos incertos.

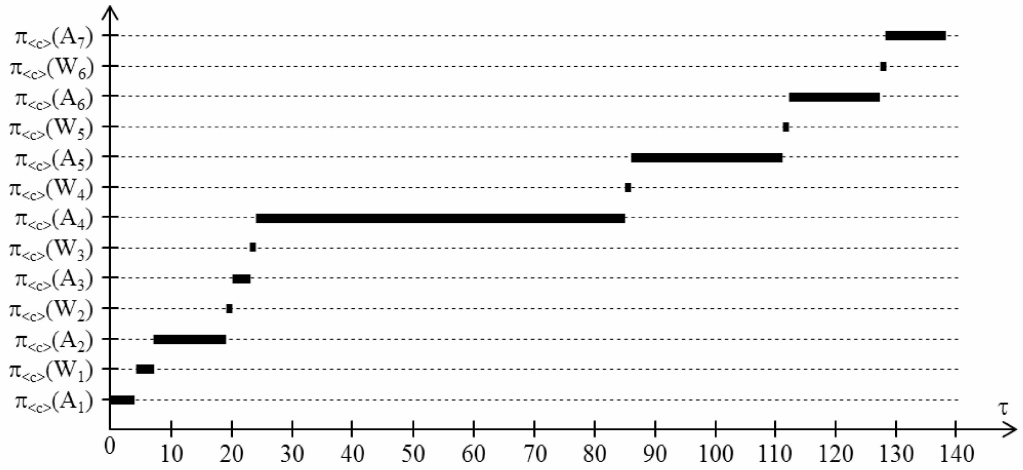
Com o propósito de mostrar a diferença do tempo de processamento gasto quando o modelo do processo é executado de modo puramente sequencial, o mesmo foi executado considerando, num primeiro momento, o tempo de processamento mínimo permitido para executar as atividades A_5, A_6 e A_7 após o recebimento da confirmação sobre o pagamento da encomenda (atividade A_4), e num segundo momento, considerando os tempos máximos permitidos. As Figuras 33a e 33b representam as distribuições de possibilidade relacionadas às instâncias do objeto $< c >$ em função do tempo considerando, respectivamente, as durações de execução mínima e máxima. Ao comparar a Figura 32 com as Figuras 33a e 33b, pode-se perceber que o tempo de processamento gasto ao se utilizar os disparos incertos é 18% menor quando se compara com a Figura 33a e 31% menor quando se compara com a Figura 33b.

Tal cenário tem como base a proposta do padrão de mudança "*Parallelize Process Fragments*" proposto por Weber, Reichert e Rinderle-Ma (2008) para prover mais flexibilidade durante a execução do modelo do processo, isto é, paralelizar fragmentos que originalmente eram puramente sequenciais. A paralelização parcial proposta nesta seção ocorre sem que seja necessário modificar a estrutura do modelo tal como definido pela flexibilidade por desvio.

Quando se considera as práticas de trabalhos na área administrativa, esta paralelização de roteiros sequenciais permite suportar sequências de eventos não previstas inicialmente no modelo do processo quando os recursos envolvidos não seguem rigorosamente as sequências predeterminadas. Desta forma, a flexibilidade incorporada no modelo de pro-



(a) Puramente sequencial com duração de execução mínima das atividades A_5, A_6 e A_7



(b) Puramente sequencial com duração de execução máxima das atividades A_5, A_6 e A_7

Figura 33 – Distribuições de possibilidade relacionadas aos locais do objeto $< c >$.

cesso permite uma melhoria na produtividade do sistema e, além disto, evita os estados de inconsistência que poderiam acontecer quando os recursos envolvidos não respeitam a estrutura do modelo do processo.

4.3 Cancelamento de WorkFlow net Sound

Uma noção importante nos WfMS's é o cancelamento. Tal noção, presente em dois dos vinte padrões de *design* definidos por Aalst et al. (2003), captura a interferência de uma atividade na execução das outras atividades em certas circunstâncias. A fim de incorporá-la nos modelos dos processos de negócio, a WF-net possibilística é usada para considerá-la de forma a manter as boas propriedades do modelo tais como alcançabilidade e, consequentemente, *Soundness*.

Após a execução das Etapas 1, 2 e 3 descritas na Definição 34, uma análise no modelo até então obtido deve ser realizada a fim de se verificar se o mesmo respeita ou não a propriedade *Soundness*. Caso a propriedade *Soundness* não seja respeitada, a noção de cancelamento não poderá ser incorporada no modelo e, consequentemente, o processo de negócio com a noção de cancelamento não poderá ser representado por meio de uma WF-net possibilística. Caso contrário, as outras quatro etapas restantes podem ser realizadas.

As noções de cancelamento de atividade e de caso podem ser generalizadas para a noção de cancelamento de região, ou seja, uma região arbitrária do modelo pode ser submetida a uma ação de cancelamento. Assim sendo, as etapas 4, 5 e 7 descritas na Definição 34 são detalhadas a fim de especificar as particularidades do problema.

O subconjunto de transições T^* definido na Etapa 4 é composto por um subconjunto de T ($T^* \subset T$) e é obtido através das seguintes regras:

- se uma atividade pertence a uma região de cancelamento, a transição de início (t_{ini}) e a transição de fim (t_{fim}) relacionadas a esta atividade serão adicionadas no conjunto T^* ($T^* \leftarrow T^* \cup \{t_{ini}, t_{fim}\}$);
- se uma atividade não pertence a uma região de cancelamento, a transição de início (t_{ini}) e a transição de fim (t_{fim}) relacionadas a esta atividade não serão adicionadas no conjunto T^* ($T^* \leftarrow T^*$), ou seja, as funções de autorização associadas à t_{ini} e à t_{fim} nunca poderão ser avaliadas como *incerta*;

Na Etapa 5, as condições relacionadas à interpretação *verdadeira* se referem às condições de início ou de fim da atividade especificada na transição t. As condições relacionadas à interpretação *incerta* se referem ao evento de cancelamento. Note-se que se o modelo do processo contém mais de uma região de cancelamento, esta condição será uma disjunção dos eventos de cancelamento relacionados a cada região.

Por fim, na etapa 7, o algoritmo de defuzzificação a ser considerado é o Algoritmo 1. Considerando isto e o "jogador" de WF-net possibilística mostrado na Figura 29, a execução de um modelo de processo em WF-net possibilística tem o seguinte comportamento:

1. se a função de autorização associada à transição t é avaliada como *verdadeira* e t está habilitada por uma marcação precisa, t é disparada com certeza e a ação A^c_{ta} associada a t é executada;
2. se a função de autorização associada à transição t é avaliada como *incerta*, t é pseudo-disparada e a ação A^i_{ta} associada a t é executada;
3. se a função de autorização associada à transição t é avaliada como *verdadeira* e t está habilitada por uma marcação imprecisa, o algoritmo de recuperação é chamado para cancelar ou finalizar os disparos incertos realizados. Quando um disparo incerto é finalizado, a ação A^f_{ta} é executada e, quando é cancelado, a ação A^l_{ta} é executada. Os disparos incertos finalizados referem-se às atividades que deveriam ter

sido executadas mas, devido à solicitação de cancelamento, devem ser canceladas. Em contra-partida, os disparos incertos cancelados referem-se a uma tentativa de encontrar a sequência de atividades que conduz o processo ao cancelamento. Após a execução do algoritmo de recuperação, a transição t é disparada com certeza e a ação A^c_{ta} associada a t é executada. Tal transição corresponde ao cancelamento da execução do processo ou de parte do processo.

Para ilustrar a execução de um modelo de processo com uma região de cancelamento, uma versão simplificada de um processo de solicitação de cartão de crédito apresentado em (WYNN et al., 2009) é utilizada:

> The process starts when an applicant submits a credit card application (with the proposed amount). Upon receiving an application (ra), a credit clerk checks whether the submitted application is complete (cc). If not, the clerk requests additional information from the applicant (rmi) and waits (WT) until this information is received (ri) before proceeding. At the same time, a timer is set (to) so that if a certain period elapses before requested information is received, another request for information is sent again. For a complete application, the clerk first checks the requested loan amount (cla). It is then followed by additional checks to validate the applicant's income and credit history. Different checks are performed depending on whether the requested loan is large (pcl) or small (pcs). The validated application is then passed on to a manager to make a decision (md). In the case of an acceptance, the credit card approval activity can start (sa). The applicant is notified of the decision (na) and, at the same time, he/she is asked for his/her preference on any extra features (wef). The applicant can choose extra features such as rewards program or secondary cardholders (cf) before a credit card is produced and delivered (dcc). This indicates the completion of the approval activity (ca) and the process ends. For a rejected application, the applicant is notified of the rejection (nr) and the process ends. An interesting feature of this process is that an applicant can request to cancel an ongoing application (ON) at any time after it was received (ra) and before the manager makes a decision (md) (WYNN et al., 2009) p. 6.

> Tradução: O processo inicia quando um requerente envia uma solicitação de cartão de crédito (com o limite de crédito proposto). Ao receber uma solicitação (ra), um funcionário verifica se a solicitação enviada está completa (cc). Caso não esteja, o funcionário solicita informações adicionais ao requerente (rmi) e espera (WT) até que as informações requisitadas sejam recebidas (ri) para depois prosseguir. Enquanto espera a chegada das informações adicionais, um temporizador é definido (to) de modo que, se um determinado período de tempo expirar antes da recepção das informações solicitadas, outro pedido de informações é novamente enviado. Se a solicitação estiver completa, então o funcionário verifica primeiramente o limite de crédito solicitado (cla). Após isto, verificações adicionais são efetuadas para validar o histórico de renda e de crédito do candidato. Verificações diferentes são realizadas dependendo se o limite de crédito solicitado é alto (pcl) ou baixo (pcs). Depois que a solicitação for validada, a mesma é então transmitida à um gerente para que ele tome uma decisão (md). No caso de uma aceitação, a atividade relacionada à aprovação do cartão de crédito é então iniciada (sa). O requerente é notificado da decisão (na) e, ao mesmo tempo, é solicitado a sua preferência em quaisquer características extras

> (wef). O requerente pode escolher recursos extras, como programa de recompensas ou titulares de cartões de crédito secundários (cf) antes do cartão de crédito ser produzido e entregue (dcc). Ao entregar o cartão de crédito ao requerente, a atividade de aprovação é concluída (ca) e o processo finaliza. Em contra-partida, se a solicitação for rejeitada, o requerente é notificado da rejeição (nr) e o processo finaliza. Uma característica interessante deste processo é que um requerente pode solicitar o cancelamento de uma solicitação de cartão de crédito em curso (ON) a qualquer momento após o recebimento da solicitação (ra) e antes do gerente tomar uma decisão (md).

O modelo representado por uma WF-net possibilística com objetos na Figura 34 descreve a versão simplificada do processo de solicitação de cartão de crédito. O símbolo $< c >$ corresponde a um objeto pertencente à classe "Crédito", bem como as variáveis x e y e todos os lugares do modelo. A fim de simplificar o modelo, as ações associadas às transições não serão descritas. A classe "Crédito" é composta pelos seguintes atributos:

- $sgBRA$, $sgBCC$, $sgBRMI$, $sgBRI$, $sgBTO$, $sgBCLA$, $sgBPCS$, $sgBPCL$, $sgBMD$, $sgBNR$, $sgBSA$, $sgBNA$, $sgBWEF$, $sgBDCC$, $sgBCF$, $sgBCA$: booleano;
- $sgERA$, $sgECC$, $sgERMI$, $sgERI$, $sgETO$, $sgECLA$, $sgEPCS$, $sgEPCL$, $sgEMD$, $sgENR$, $sgESA$, $sgENA$, $sgEWEF$, $sgEDCC$, $sgECF$, $sgECA$: booleano;
- *cancel* : booleano.

Observe-se que outros possíveis atributos específicos para o processo de solicitação de cartão de crédito podem ser definidos para este exemplo; entretanto, a não representação deles não interfere no entendimento da abordagem. Os significados dos atributos declarados anteriormente para a classe "Crédito" são:

- os atributos $sgBRA$, $sgBCC$, $sgBRMI$, $sgBRI$, $sgBTO$, $sgBCLA$, $sgBPCS$, $sgBPCL$, $sgBMD$, $sgBNR$, $sgBSA$, $sgBNA$, $sgBWEF$, $sgBDCC$, $sgBCF$ e $sgBCA$, quando verdadeiros, representam a ocorrência de um evento indicando, respectivamente, o início das seguintes atividades: *receive application* (ra), *check for completeness* (cc), *request more info* (rmi), *receive more info* (ri), *time out* (to), *check loan amount* (cla), *perform check for small amount* (pcs), *perform check for large amount* (pcl), *make decision* (md), *notify reject* (nr), *start approval* (sa), *notify acceptance* (na), *want extra features* (wef), *deliver credit card* (dcc), *choose features* (cf) and *complete approval* (ca);
- os atributos $sgERA$, $sgECC$, $sgERMI$, $sgERI$, $sgETO$, $sgECLA$, $sgEPCS$, $sgEPCL$, $sgEMD$, $sgENR$, $sgESA$, $sgENA$, $sgEWEF$, $sgEDCC$, $sgECF$ e $sgECA$, quando verdadeiros, representam a ocorrência de um evento indicando, respectivamente, o término das seguintes tarefas: *receive application* (ra), *check for*

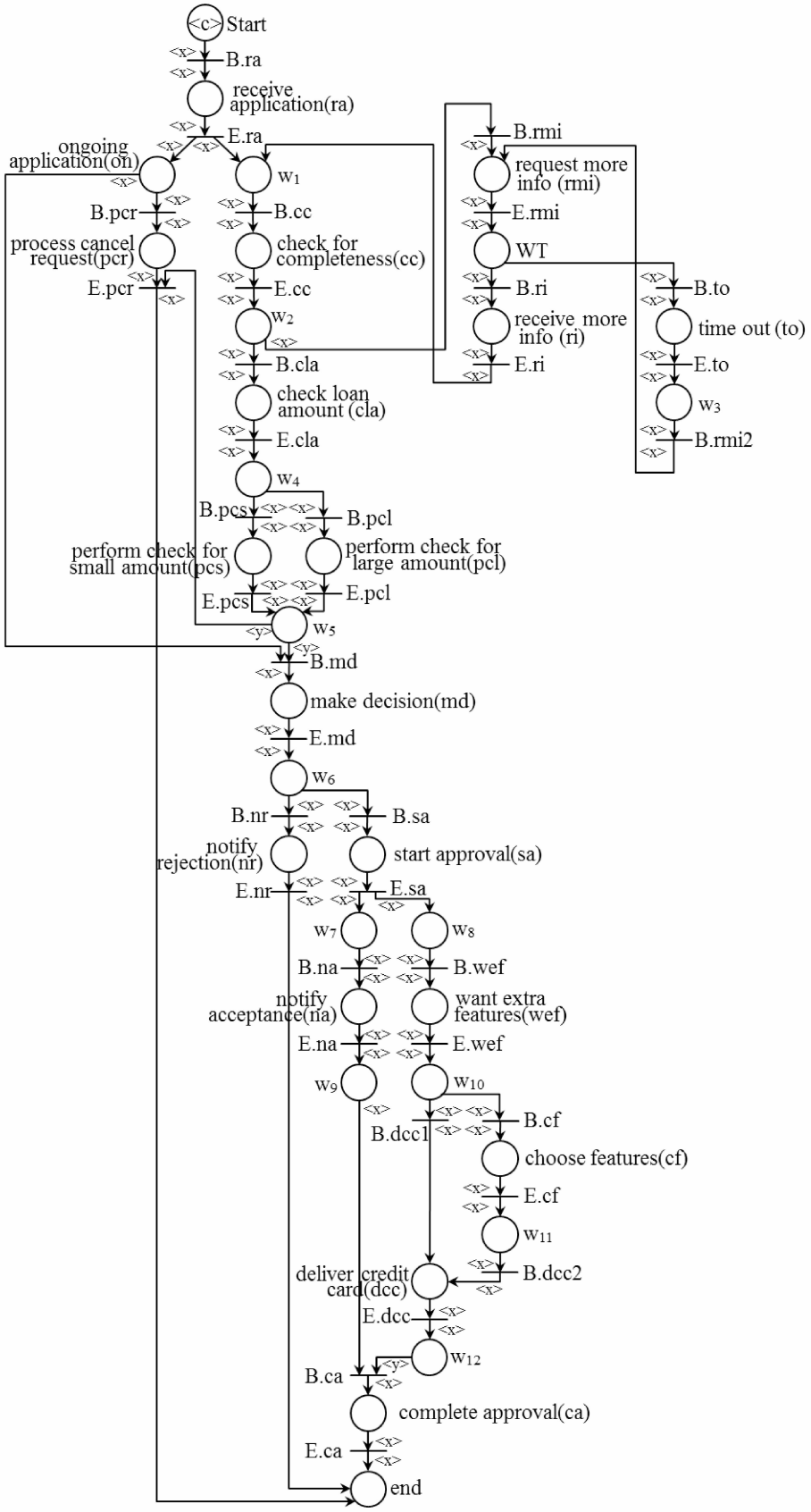


Figura 34 – Modelo de uma versão simplificada de um processo de solicitação de cartão de crédito representado por uma WF-net possibilística.

completeness (cc), *request more info* (rmi), *receive more info* (ri), *time out* (to), *check loan amount* (cla), *perform check for small amount* (pcs), *perform check for large amount* (pcl), *make decision* (md), *notify reject* (nr), *start approval* (sa), *notify acceptance* (na), *want extra features* (wef), *deliver credit card* (dcc), *choose features* (cf) and *complete approval* (ca);

- o atributo *cancel*, quando verdadeiro, representa o recebimento da solicitação de cancelamento do caso por parte do requerente.

A fim de verificar se o modelo representado por uma WF-net possibilística mostrado na Figura 34 respeita a propriedade *Soundness*, o programa aplicativo *Plataform Independent Petri net Editor (PIPE)* (DINGLE; KNOTTENBELT; SUTO, 2009) criado como um projeto de um grupo de alunos de pós-graduação do Departamento de Computação do *Imperial College London* foi utilizado. Como provado por Aalst (1998a), a propriedade *Soundness* corresponde a duas propriedades da rede de Petri com um curto circuito[4]: vivacidade (*liveness*) e limitabilidade (*boundness*). Levando isto em consideração, a WF-net possibilística mostrada na Figura 34 foi editada no aplicativo PIPE considerando a estrutura de uma rede de Petri com um curto-circuito. Após a edição, a análise do espaço de estados disponível no aplicativo foi realizada. O resultado da análise é mostrado na Figura 35. Observando o resultado, pode-se concluir que a WF-net possibilística é *sound* dado que o modelo é livre de *deadlock* (*deadlock* = *false*) e limitado (*bounded* = *safe* = *true*), ou seja, tanto a propriedade *liveness* quanto a propriedade *boundedness* são garantidas no modelo.

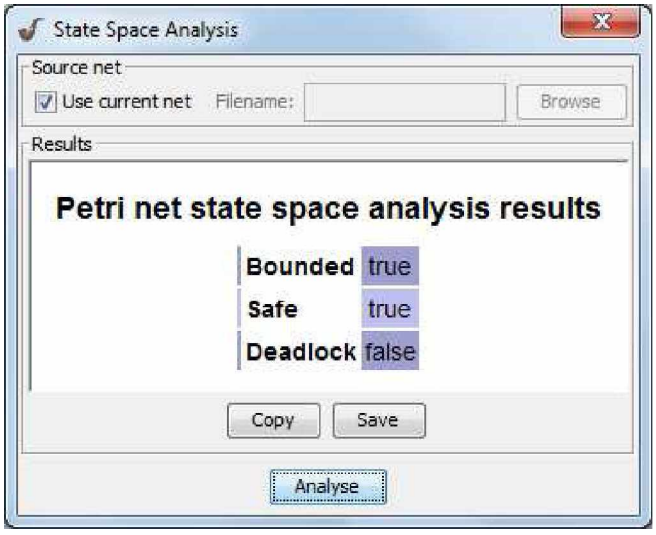


Figura 35 – Tela do software PIPE com o resultado da análise do espaço de estados da rede de Petri derivada da WF-net possibilística mostrada na Figura 34.

[4] Uma rede de Petri com um curto circuito é caracterizada através da presença de uma transição t^* que contém como lugar de entrada o lugar o e, como lugar de saída, o lugar i

Sabendo que o requerente pode requisitar o cancelamento do processo antes que o gerente tome uma decisão e após o recebimento do pedido, a região de cancelamento presente no modelo é composta pelas seguintes atividades: *cc*, *rmi*, *ri*, *to*, *cla*, *pcs* e *pcl*. Logo, as funções de autorização associadas às transições de início e fim relacionadas a estas atividades produzirão como resultado as possíveis interpretações: *verdadeira*, *incerta* ou $falsa$. As outras funções de autorização relacionadas ao restante das atividades (*ra*, *pcr*, *md*, *nr*, *sa*, *na*, wef, *dcc*, cf e *ca*) produzirão como resultado as possíveis interpretações: *verdadeira* ou $falsa$. Note-se que quando uma solicitação de cancelamento é requisitada, a atividade responsável por finalizar a execução do processo será a atividade *pcr*.

A Tabela 2 mostra as funções de autorização associadas às transições pertencentes ao modelo em WF-net possibilística apresentado na Figura 34. As condições necessárias para que uma função de autorização seja avaliada como *verdadeira* ou como *incerta* durante a execução do modelo são representadas, respectivamente, pelas colunas com as seguintes subscrições: "*verdadeira se*" e "*incerta se*". A interpretação $falsa$ não é representada, porém, uma função de autorização será avaliada como $falsa$ sempre que as condições associadas às interpretações *verdadeira* e *incerta* são inválidas. Por fim, se uma função de autorização não contém uma interpretação *incerta* associada a ela, o espaço reservado para ela na Tabela 2 estará vazio.

Tabela 2 – Funções de autorização associadas às transições pertencentes ao modelo em WF-net possibilística apresentado na Figura 34.

	verdadeira se	***incerta se***
$\eta_x(B.ra) =$	$x.begRA$	
$\eta_x(E.ra) =$	$x.endRA$	
$\eta_x(B.pcr) =$	$x.cancel$	
$\eta_{xy}(E.pcr) =$	$x.cancel$	
$\eta_x(B.cc) =$	$x.begCC \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.cc) =$	$x.endCC \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.rmi1) =$	$x.begRMI \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.rmi2) =$	$x.begRMI \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.rmi) =$	$x.endRMI \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.ri) =$	$x.begRI \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.ri) =$	$x.endRI \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.to) =$	$x.begTO \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.to) =$	$x.endTO \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.cla) =$	$x.begCLA \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.cla) =$	$x.endCLA \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.pcs) =$	$x.begPCS \wedge \neg x.cancel$	$x.cancel$

$\eta_x(E.pcs) =$	$x.endPCS \wedge \neg x.cancel$	$x.cancel$
$\eta_x(B.pcl) =$	$x.begPCL \wedge \neg x.cancel$	$x.cancel$
$\eta_x(E.pcl) =$	$x.endPCL \wedge \neg x.cancel$	$x.cancel$
$\eta_{xy}(B.md) =$	$x.begMD \wedge \neg x.cancel$	
$\eta_x(E.md) =$	$x.endMD$	
$\eta_x(B.nr) =$	$x.begNR$	
$\eta_x(E.nr) =$	$x.endNR$	
$\eta_x(B.sa) =$	$x.begSA$	
$\eta_x(E.sa) =$	$x.endSA$	
$\eta_x(B.na) =$	$x.begNA$	
$\eta_x(E.na) =$	$x.endNA$	
$\eta_x(B.wef) =$	$x.begWEF$	
$\eta_x(E.wef) =$	$x.endWEF$	
$\eta_x(B.dcc1) =$	$x.begDCC$	
$\eta_x(B.dcc2) =$	$x.begDCC$	
$\eta_x(E.dcc) =$	$x.endDCC$	
$\eta_x(B.cf) =$	$x.begCF$	
$\eta_x(E.cf) =$	$x.endCF$	
$\eta_{xy}(B.ca) =$	$x.begCA$	
$\eta_x(E.ca) =$	$x.endCA$	

As funções de autorização impõem restrições extras nos disparos das transições. As transições, as quais contêm uma interpretação *incerta* associada a elas, serão disparadas com incerteza apenas quando o requerente solicitar pelo cancelamento do caso, ou seja, quando o atributo *cancel* for *verdadeiro*. Caso contrário, todos os disparos serão certos. Através dos disparos incertos e, consequentemente, das marcações imprecisas, uma sequência das atividades que deveriam ser executadas caso o cancelamento não fosse requerido será definida para que a execução do processo seja finalizada corretamente independentemente do seu estado atual.

Considerando o modelo do processo representado na Figura 34, um objeto $< c >$ no lugar *md* indica que o pedido foi validado e enviado a um gerente para o mesmo tomar uma decisão, impossibilitando assim qualquer pedido de cancelamento por parte do requerente. No entanto, um objeto $< c >$ no lugar *pcr* indica que um pedido de cancelamento foi recebido e, consequentemente, todas as atividades que pertencem à região de cancelamento devem ser canceladas ou interrompidas caso estejam em execução.

No modelo apresentado, se o requerente não solicitar o cancelamento do pedido, todos os disparos serão certos e todas as marcações precisas. Em contrapartida, se o cancelamento for solicitado, ou seja, o atributo *cancel* for verdadeiro, alguns disparos incertos

ocorrerão produzindo, assim, uma marcação imprecisa. Neste caso, quando a transição $E.pcr$ estiver habilitada por uma marcação imprecisa, o algoritmo de recuperação descrito no Algoritmo 1 será chamado para retornar ao estado certo do modelo do processo dado que a função de autorização associada à transição $E.pcr$ é avaliada como verdadeira. Após a execução do algoritmo, o processo é cancelado corretamente através do disparo certo de $E.pcr$.

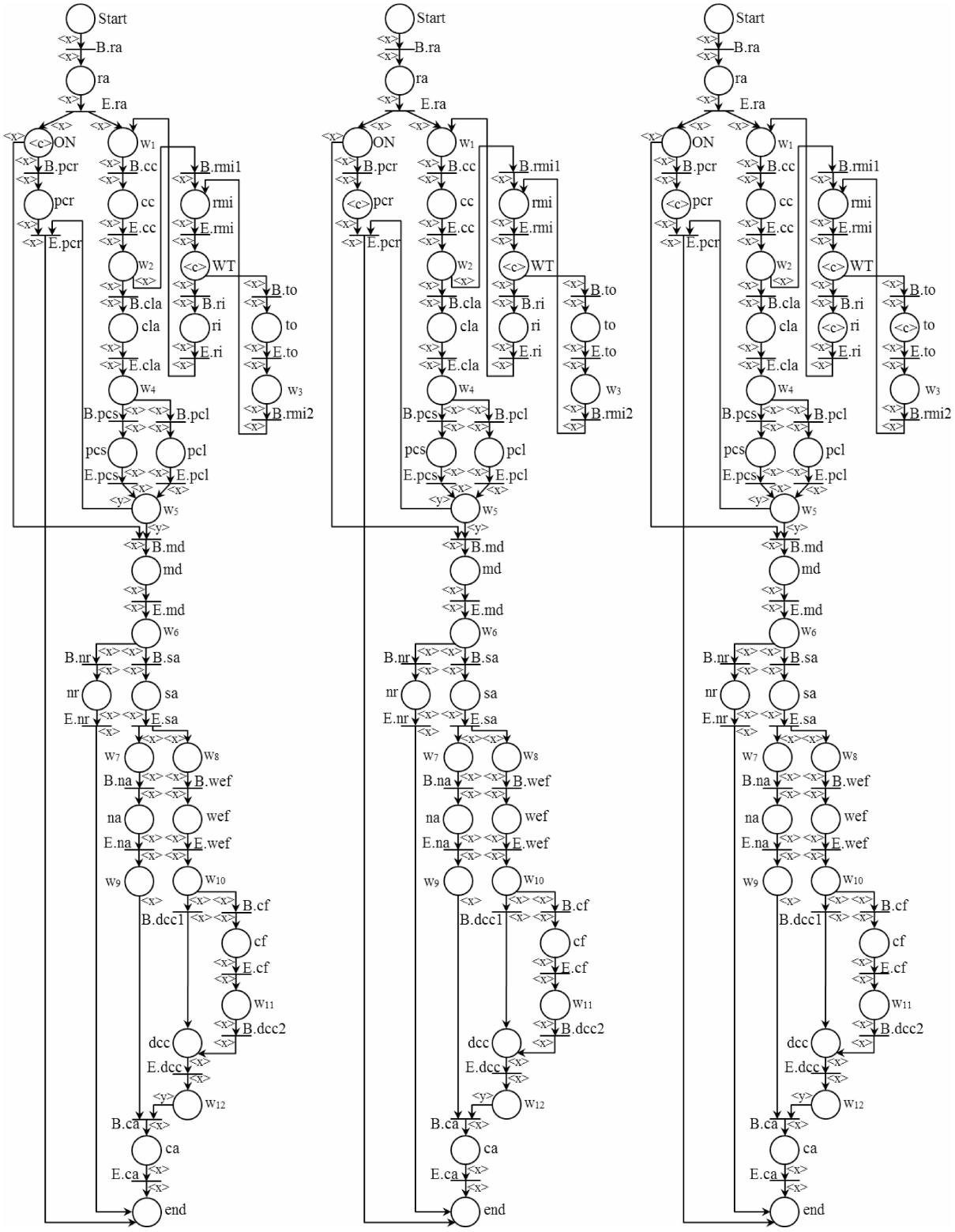
A fim de ilustrar a execução de um possível pedido de cancelamento, considere que as atividades ra, cc e rmi já foram executadas, isto é, as transições $B.ra$, $E.ra$, $B.cc$, $E.cc$, $B.rmi$ e $E.rmi$ já foram disparadas com certeza (Figura 36a). Se um pedido de cancelamento for solicitado pelo requerente neste momento, ou seja, o atributo *cancel* se torna verdadeiro, tem-se o seguinte:

- a transição $B.pcr$ está habilitada por uma marcação precisa e a função de autorização associada a ela é avaliada como *verdadeira* ($\eta_{<c>}(B.pcr) = verdadeira$); então a transição $B.pcr$ é disparada com certeza, isto é, o objeto $< c >$ é removido do lugar ON, um novo objeto $< c >$ é produzido no lugar pcr e a ação $A^c_{B.pcr}$ associada a $B.pcr$, caso definida, é executada (Figura 36b);

- as funções de autorização associadas às transições $B.cc$, $E.cc$, $B.cla$, $E.cla$, $B.pcs$, $E.pcs$, $B.pcl$, $E.pcl$, $B.rmi1$, $E.rmi$, $B.ri$, $E.ri$, $B.to$, $E.to$ e $B.rmi2$ são todas avaliadas como incertas. Entretanto, apenas as transições $B.ri$ e $B.to$ podem ser disparadas dado que são as únicas habilitadas pelo objeto $< c >$. Logo, tanto $B.ri$ quanto $B.to$ são disparadas com incerteza e, consequentemente, cópias do objeto $< c >$ são produzidas, respectivamente, nos lugares ri e to (Figura 36c). Note que o objeto $< c >$ não é removido do lugar WT e as ações $A^i_{B.ri}$ e $A^i_{B.to}$ associadas respectivamente a $B.ri$ e a $B.to$, caso definidas, são executadas;

- a partir da marcação imprecisa obtida no passo anterior, as transições $E.ri$ e $E.to$ estão habilitadas e as funções de autorização associadas a elas são avaliadas como incertas ($\eta_{<c>}(E.ri) = \eta_{<c>}(E.to) = incerta$); logo ambas são disparadas com incerteza, cópias do objeto $< c >$ são produzidas nos lugares w_1 e w_3 respectivamente e, caso definidas, as ações $A^i_{E.ri}$ e $A^i_{E.to}$ associadas respectivamente a $E.ri$ e a $E.to$ são executadas;

- considerando a evolução da marcação, as transições $B.cc$, $B.rmi2$, $E.cc$, $B.cla$, $E.cla$, $B.pcs$, $B.pcl$ e $E.pcs$ serão disparadas com incerteza dado que as funções de autorização associadas a elas são avaliadas como *incerta* ($\eta_{<c>}(B.cc) = \eta_{<c>}(B.rmi2) = \eta_{<c>}(E.cc) = \eta_{<c>}(B.cla) = \eta_{<c>}(E.cla) = \eta_{<c>}(B.pcs) = \eta_{<c>}(B.pcl) = \eta_{<c>}(E.pcs) = incerta$). Consequentemente, cópias do objeto $< c >$ serão produzidas nos respectivos lugares cc, rmi, w_2, cla, w_4, pcs, pcl e w_5 e, caso definidas, as ações $A^i_{B.cc}$, $A^i_{B.rmi2}$, $A^i_{E.cc}$, $A^i_{B.cla}$, $A^i_{E.cla}$, $A^i_{B.pcs}$, $A^i_{B.pcl}$ e $A^i_{E.pcs}$ são executa-

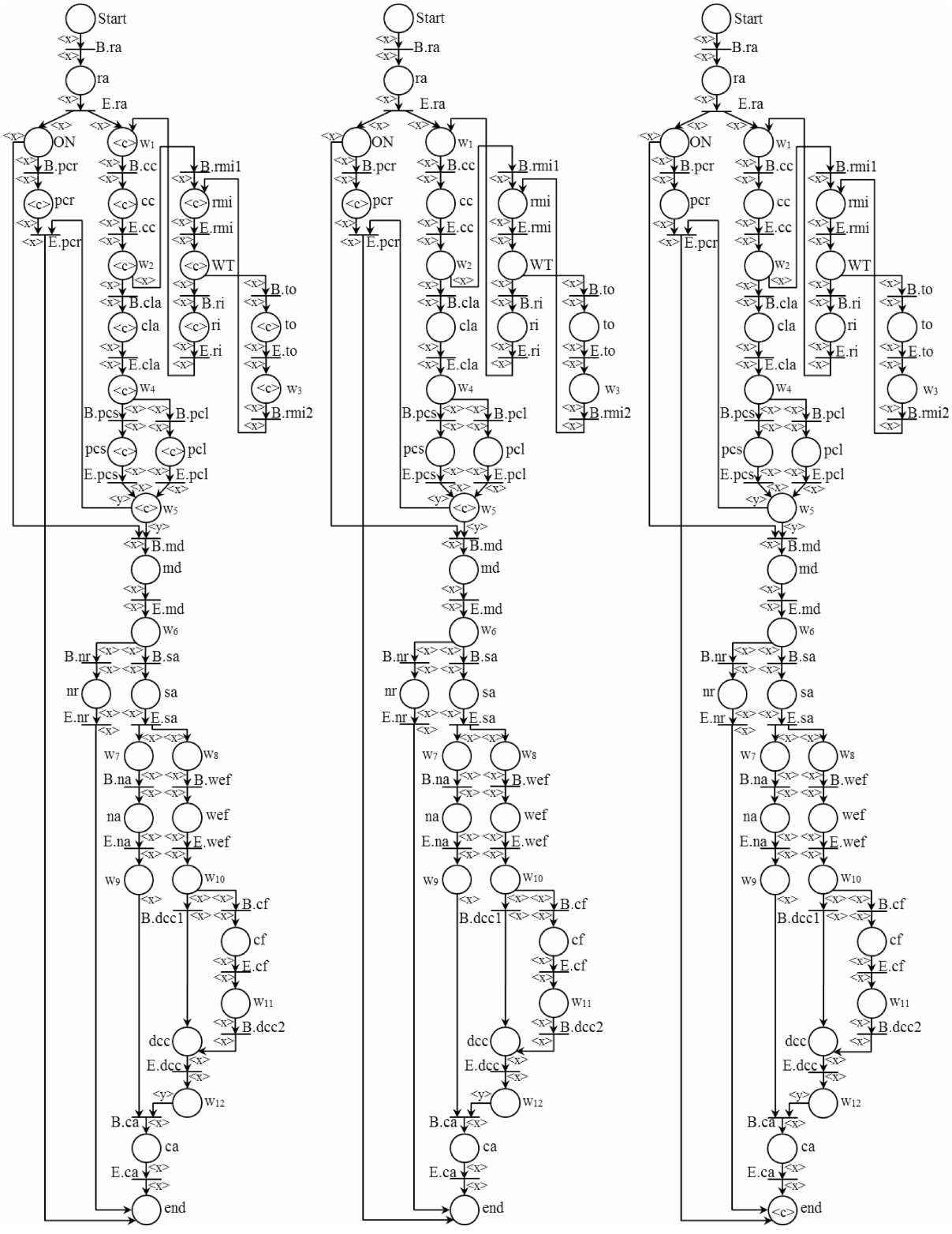


(a) Marcação inicial considerada pela simulação.

(b) Marcação precisa obtida após o disparo certo da transição $B.pcr$.

(c) Marcação imprecisa obtida após o disparo incerto das transições $B.ri$ e $B.to$.

Figura 36 – Resultados da simulação - Regiões de Cancelamento (Parte 1).



(a) Marcação imprecisa obtida após todos os disparos incertos.

(b) Marcação precisa obtida após a execução do algoritmo de recuperação

(c) Processo cancelado.

Figura 37 – Resultados da simulação - Regiões de Cancelamento (Parte 2).

das. O estado do modelo do processo resultante após o disparo desta sequência de transições é mostrado na Figura 37a;

- as transições $B.rmi1$, $E.rmi$ e $E.pcl$ estão habilitadas por uma marcação imprecisa e as funções de autorização associadas a elas são avaliadas como incertas ($\eta_{<c>}(B.rmi1) = \eta_{<c>}(E.rmi) = \eta_{<c>}(E.pcl) = incerta$). Entretanto, tais transições não podem ser disparadas com incerteza devido à existência do objeto $< c >$ nos seus lugares de saída (rmi, WT e w_5, respectivamente) como foi explicado na subseção 2.1.4.5.1;

- por fim, a transição $E.pcr$ está habilitada por uma marcação imprecisa dado a marcação atual mostrada na Figura 37a e, além disto, sua função de autorização é avaliada como verdadeira ($\eta_{<cc>}(E.pcr) = verdadeira$). Esta situação ocorre devido à solicitação do cancelamento do processo por parte do requerente e a obrigação de cancelar ou de interromper a execução de todas as atividades após tal solicitação. A fim de retornar para a marcação precisa para que se possa concluir o cancelamento do processo, o algoritmo de recuperação descrito na subseção 2.1.4.5.1 é executado. Tal algoritmo irá finalizar os disparos incertos das transições $E.pcs$, $B.pcs$, $E.cla$, $B.cla$, $E.cc$, $B.cc$, $E.ri$ e $B.ri$ executando, caso definidas, as ações $A^f_{E.pcs}$, $A^f_{B.pcs}$, $A^f_{E.cla}$, $A^f_{B.cla}$, $A^f_{E.cc}$, $A^f_{B.cc}$, $A^f_{E.ri}$ e $A^f_{B.ri}$. Além disto, o algoritmo de defuzzificação irá cancelar os disparos incertos das transições $B.pcl$, $B.to$, $E.to$ e $B.rmi2$ executando, caso definidas, as ações $A^l_{B.pcl}$, $A^l_{B.to}$, $A^l_{E.to}$ e $A^l_{B.rmi2}$ (Figura 37b). As transições E_{pcl} e B_{rmi1} têm seus lugares de saída marcados devido o disparo incerto das transições E_{pcs} e B_{rmi2} respectivamente. Além destas duas, o lugar de saída da transição E_{rmi} também está marcado devido a marcação obtida antes da solicitação de cancelamento. Tanto a transição E_{rmi} quanto as transições E_{pcl} e B_{rmi1} não foram disparadas devido à preexistência do objeto $< c >$ nos seus lugares de saída. Após a execução do algoritmo de defuzzificação, a transição $E.pcr$ pode ser disparada com certeza dado que a mesma está habilitada por uma marcação precisa e a função de autorização associada a ela é avaliada como *verdadeira*. O disparo certo de $E.pcr$ permite executar, caso definida, a ação $A^c_{E.pcr}$ e concluir a solicitação do cancelamento do processo, ou seja, apenas o local *end* fica marcado (Figura 37c). Note-se que as ações A^i_{ta}, A^f_{ta} e A^l_{ta} são todas relacionadas ao tratamento do pedido de cancelamento.

Tal cenário, através da utilização dos disparos incertos, definiu uma sequência de disparos responsável por produzir o objeto $< c >$ no lugar w_5. Após a execução desta sequência de disparos, a transição $E.pcr$ se tornou habilitada por uma marcação imprecisa e, considerando que a sua função de autorização estava avaliada como *verdadeira*, o algoritmo de recuperação foi chamado para retornar ao estado certo do modelo do pro-

cesso. Após a execução do algoritmo, a transição *E.pcr* pôde ser disparada com certeza permitindo, assim, finalizar a execução do processo com sucesso.

Os disparos incertos e, consequentemente, as marcações imprecisas substituem o uso dos arcos *reset* (ARAKI; KASAMI, 1976a) no modelo do processo quando a noção de cancelamento deve ser considerada. Ao fazer isto, as propriedades, tais como alcançabilidade e limitabilidade (que se tornam indecidíveis em uma rede *reset* com mais de dois arcos *reset*) são mantidas dado que a estrutura do modelo do processo não é modificada.

Considerando que as boas propriedades do modelo são mantidas, ou seja, a decidibilidade da propriedade *Soundness* é garantida, todas as marcações que devem ser consideradas são obtidas através do disparo incerto e da marcação imprecisa. A existência de tais marcações é garantida pela propriedade *Soundness* que afirma a não existência de transições mortas presente no modelo do processo.

Como um pedido de cancelamento pode ser considerado como um desvio/distúrbio significativo no processo, um tratamento adequado deve ser realizado para que o cancelamento das devidas atividades não alcance um estado de inconsistência. Se não houver um tratamento adequado para realizar o cancelamento das atividades em execução, um tipo de intervenção humana deve ser realizado com o intuito de reiniciar manualmente o processo de forma a reverter o estado de inconsistência. Para evitar isso, o "jogador" de WF-net possibilística aqui proposto, assegura a flexibilidade do modelo para executar o pedido de cancelamento de forma a recuperar o estado normal de funcionamento do processo em execução sem qualquer tipo de inconsistência.

4.4 Considerações Finais

Neste capítulo a WF-net possibilística foi redefinida e, posteriormente, aplicada em problemas específicos relacionados aos processos de negócio. Uma WF-net Possibilística é baseada na combinação da estrutura de roteamento de uma WF-net com a marcação imprecisa e o disparo incerto de uma rede de Petri possibilística. Esta combinação produz um tipo de modelo que é capaz de lidar com os problemas de desvio (não respeito de uma sequência de eventos prevista no modelo de especificação do processo (THOMPSON; TORABI, 2009)) na fase de execução dos modelos dos processos de negócio.

A WF-net possibilística é aplicada, na Seção 4.2, aos processos de negócio em execução com roteiros puramente sequenciais com o objetivo de permitir uma paralelização parcial na ordenação das atividades sem que seja necessário modificar a estrutura do modelo tal como definido pela flexibilidade por desvio. Neste caso, a execução de novas atividades pode ser iniciada mesmo que as suas precedentes estejam parcialmente executadas, ou seja, não concluídas. Esta paralelização de roteiros sequenciais permite uma melhoria na produtividade do sistema e evita estados de inconsistência que poderiam acontecer quando os recursos envolvidos não respeitam a estrutura do modelo do processo.

Na Seção 4.3, a WF-net possibilística é aplicada aos processos de negócio com noção de cancelamento de forma a considerar o cancelamento da execução de certas atividades e, em alguns casos, do processo. Levando em consideração que a estrutura do modelo não é modificada, a decidibilidade das propriedades, tais como alcançabilidade e limitabilidade, são mantidas.

Capítulo 5

WorkFlow net Interorganizacional Possibilística

Este capítulo define e, posteriormente, aplica a *WorkFlow net* interorganizacional possibilística em problemas relacionados com desvios em processos de negócio. A Seção 4.1 define a IOWF-net possibilística baseada na WF-net possibilística definida no Capítulo 4. Na Seção 5.1, a IOWF-net possibilística é aplicada aos processos de negócio interorganizacionais com o objetivo de lidar com problemas de comunicação, tais como mensagens perdidas, atrasadas ou inesperadas entre os processos de forma a evitar situações de inconsistências no modelo dos processos. A Seção 5.3 exemplifica situações de *deadlock* causadas, em sua grande maioria, por razões estruturais nos modelos dos processos de negócio que se comunicam/colaboram e que são evitadas através do modelo em IOWF-net possibilística sem que a sua estrutura seja modificada.

5.1 Definição da *WorkFlow net* interorganizacional possibilística

A IOWF-net possibilística é baseada na combinação da estrutura de roteamento e dos protocolos de comunicação de uma IOWF-net com a noção de disparo incerto e marcação imprecisa de uma rede de Petri possibilística. Tal abordagem produz um tipo de modelo capaz de lidar com possíveis falhas de comunicação e existências de problemas estruturais no modelo dos processos durante a execução dos mesmos. Uma variável booleana é associada a cada disparo de transição e tal variável é vista essencialmente como um valor externo correspondente a uma mensagem recebida a partir de uma atividade (ou a partir de outro processo). Os valores internos dos atributos das fichas também podem habilitar transições. A definição da IOWF-net possibilística é dada a seguir:

Definição 35 ***(WorkFlow net Interorganizacional Possibilística)*** *Uma IOWF-net possibilística é uma tupla*

$$R = \{WP_1, WP_2, \ldots, WP_w, P_{AC}, AC\} \tag{24}$$

onde:

1. $w \in N$ *é a quantidade de processos modelados;*
2. *para cada* $k \in \{1, \ldots, w\}$*:* WP_k *representa um modelo de processo em WF-net possibilística com lugar de início* i_k *e lugar de término* o_k*;*
3. *para cada* $k, l \in \{1, \ldots, w\}$*: se* $k \neq l$ *então* $(P_k \cup T_k) \cap (P_l \cup T_l) = \emptyset$*;*
4. $T^{\blacktriangle} = \bigcup_{k \in \{1,\ldots,w\}} T_k$, $P^{\blacktriangle} = \bigcup_{k \in \{1,\ldots,w\}} P_k$, $F^{\blacktriangle} = \bigcup_{k \in \{1,\ldots,w\}} (P_k \cup T_k) \cap (T_k \cup P_k)$ *(relações entre os elementos das* WP_k*);*
5. P_{AC} *é o conjunto de elementos de comunicação assíncrona (lugares de comunicação);*
6. $AC \subseteq P_{AC} \times \mathbb{P}(T^{\blacktriangle}) \times \mathbb{P}(T^{\blacktriangle})$ *é a relação de comunicação assíncrona*[1]*.*

Cada elemento de comunicação assíncrona, também conhecido como lugar de comunicação, corresponde a um lugar em P_{AC} no modelo do processo apenas para fins de análise das propriedades do modelo. Caso contrário, ele é representado em cada modelo de processo (nas WF-nets possibilísticas) durante a sua execução como uma condição a ser considerada por outro processo.

Considerando que a relação AC especifica o conjunto de transições de entrada e o de saída para cada elemento de comunicação assíncrona (AALST, 1998b), a condição representada por um lugar de comunicação p é vista como uma variável booleana associada à função de autorização de cada transição de saída de p ou às ações A^c_{ta} e A^i_{ta} de cada transição de entrada de p. Ao ser associada à função de autorização, esta variável representa a chegada do evento enviado por outro processo e, ao ser associada às ações, ela representa o envio do evento à outro processo.

Observe-se que a variável booleana que fica associada a um lugar de comunicação p aparece tanto na ação A^c_{ta} quanto na ação A^i_{ta} de cada transição de entrada de p. Caso esta variável booleana fosse associada apenas à ação A^c_{ta}, quando um disparo incerto ocorre nenhuma dedução sobre o estado atual do modelo do processo global é possível, impossibilitando assim a continuidade da execução do mesmo.

Para elaborar o modelo de um processo de negócio considerando uma IOWF-net possibilística, 7 etapas devem ser realizadas. Tais etapas, definidas a seguir, têm particularidades de acordo com o tipo de problema a ser tratado, as quais são especificadas no momento da definição do problema.

[1] $\mathbb{P}(T^{\blacktriangle})$ é o conjunto de todos os subconjuntos não vazios de $T^{\blacktriangle}$

Definição 36 ***(Etapas de Modelagem)*** *As etapas necessárias para modelar um processo de negócio interorganizacional em IOWF-net possibilística são:*

1. *definir a quantidade w de processos a serem modelados por uma WF-net possibilística;*

2. *definir as w WF-nets possibilísticas de acordo com as etapas especificadas na Definição 34;*

3. *definir os lugares de comunicação (P_{AC});*

4. *definir a relação de comunicação assíncrona AC;*

5. *seja T^S o conjunto de transições de saída especificado na relação AC($T^S \subset T^{\blacktriangle}$), para cada transição $t \in T^S$, acrescentar as condições necessárias referentes ao(s) lugar(es) de comunicação na função de autorização relacionada a t de tal forma que a mesma possa ser avaliada como verdadeira e/ou incerta durante a execução;*

6. *seja T^E o conjunto de transições de entrada especificado na relação AC ($T^E \subset T^{\blacktriangle}$), para cada transição $t \in T^E$, acrescentar na ação A^c_{ta} e na ação A^i_{ta}, a(s) ação(ões) responsável(is) por substituir o(s) lugar(es) de comunicação(ões) associado(s) a t no modelo do processo quando o mesmo está sendo executado;*

7. *executar cada um dos w modelos que interagem na IOWF-net possibilística separadamente considerando o jogador de WF-net possibilística mostrado na Figura 29 e o algoritmo de recuperação descrito no Algoritmo 1, isto é:*

   a) *se a função de autorização associada à transição t é avaliada como verdadeira e t está habilitada por uma marcação precisa, então t é disparada com certeza e a ação A^c_{ta} associada a t é executada;*

   b) *se a função de autorização associada à transição t é avaliada como incerta, então t é disparada de forma incerta e a ação A^i_{ta} associada a t é executada;*

   c) *se a função de autorização associada à transição t é avaliada como verdadeira e t está habilitada por uma marcação imprecisa, então o algoritmo de recuperação é chamado para finalizar ou cancelar os disparos incertos realizados durante a execução do modelo do processo. Quando um disparo incerto é finalizado, a ação A^f_{ta} é executada e, quando é cancelado, a ação A^l_{ta} é executada. Após a execução do algoritmo de recuperação, a transição t é disparada com certeza e a ação A^c_{ta} associada a t é executada.*

5.2 Falhas de comunicação numa *WorkFlow net* Interorganizacional Sound

Os processos de *workflow* interorganizacionais, devido as suas características de colaboração entre organizações, tendem a enfrentar problemas de comunicação tais como mensagens perdidas, atrasadas ou inesperadas entre os processos, levando eventualmente à situações de *deadlock*. O tratamento usual destes problemas na maioria dos WfMS's é suspender a execução do modelo do processo e reportar o problema ao administrador responsável por realizar uma intervenção manual de forma a resolver o problema para que o modelo do processo possa voltar à sua execução. Entretanto, tal intervenção torna-se cada vez mais onerosa devido à crescente complexidade dos processos e à demanda de recursos humanos especializados com seu alto custo e tempo de resposta lenta (ALONSO et al., 2000). O ideal é que após a identificação de tais problemas, o WfMS seja capaz de executar algum procedimento para evitar o problema ou para uma recuperação que transforme o estado inválido num estado válido (JALOTE, 1989).

Com o intuito de incorporar a flexibilidade por *desvio* no modelo do processo a fim de recuperar o estado do processo após a ocorrência de algum problema de comunicação, a IOWF-net possibilística visa modelar os processos interorganizacionais. Desta forma, espera-se desviar de forma o mais automática possível destes problemas durante a execução do modelo do processo, possibilitando a correta evolução para a maioria dos casos, mesmo em caso de falhas de comunicação entre processos.

Ao se considerar as etapas de modelagem especificadas na Definição 36, na etapa 2, responsável por modelar os processos de cada parceiro de negócio, o conjunto T^*, que se refere às transições que terão as funções de autorização produzindo como resultado as possíveis interpretações *verdadeira*, *incerta* e $falsa$, é composto pelas transições $t \in T^{\blacktriangle}$, isto é, o conjunto T^* é um subconjunto de $T^{\blacktriangle}$ ($T^* \subseteq T^{\blacktriangle}$). Neste caso, a interpretação *incerta* é responsável pelo desvio dos problemas de comunicação, os quais podem levar eventualmente à situações de *deadlock* e, quando não definida, a função de autorização nunca é avaliada como *incerta*.

O processo que precede a apresentação de um artigo em uma conferência descrito na Seção 2.3 é considerado para ilustrar o caso. O modelo em IOWF-net possibilística que representa tal processo é apresentado na Figura 38. Diferentemente do modelo mostrado na Seção 2.3 usado para ilustrar o não respeito a propriedade *Soundness*, este respeita tal propriedade, tanto localmente quanto globalmente como provado em (AALST, 1998b).

Como explicado na Seção 5.1, na execução dos modelos, os lugares de comunicação representam variáveis booleanas associadas às funções de autorização ou às ações A^c_{ta} e A^i_{ta} nas WF-nets possibilísticas dos processos AU e PC. Para tanto, estes modelos são considerados separadamente como apresentado nas Figuras 39a e 39b.

$< a >$ é um objeto pertencente à classe "Artigo", bem como a variável x e todos os

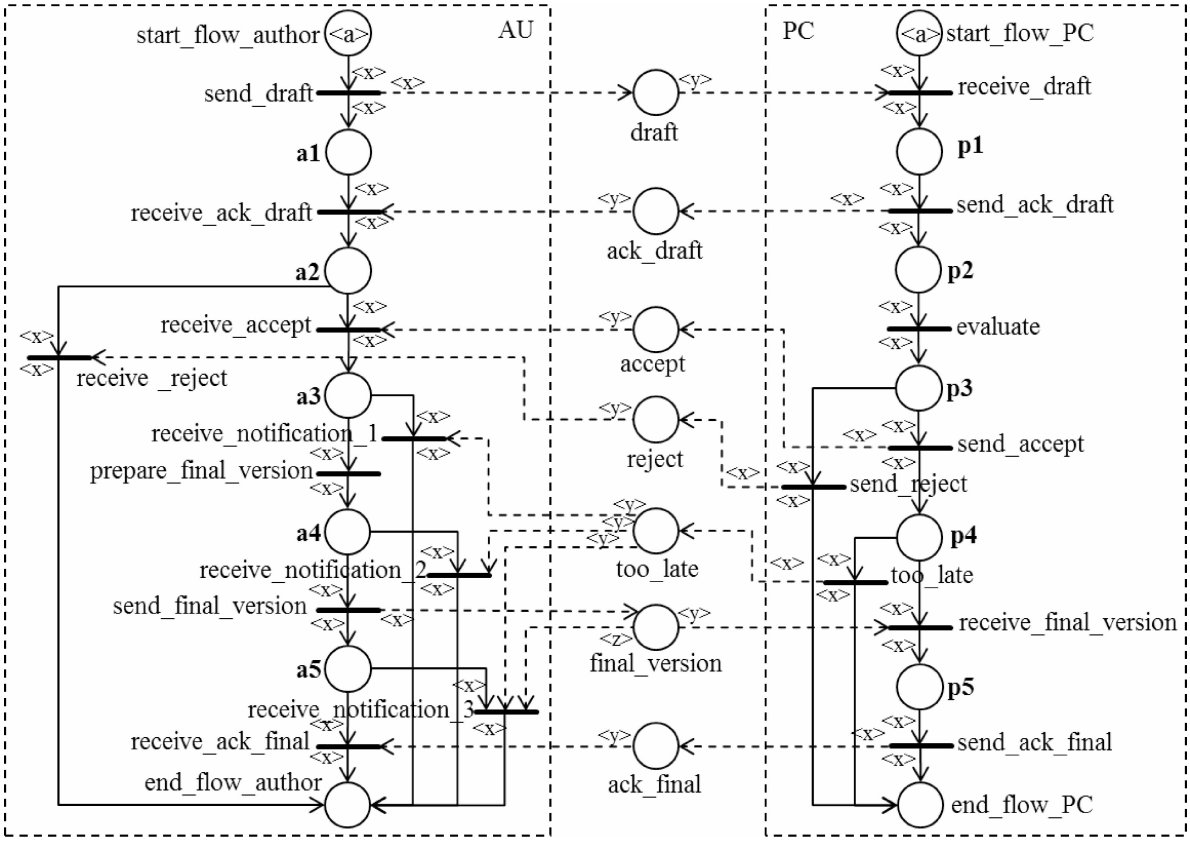
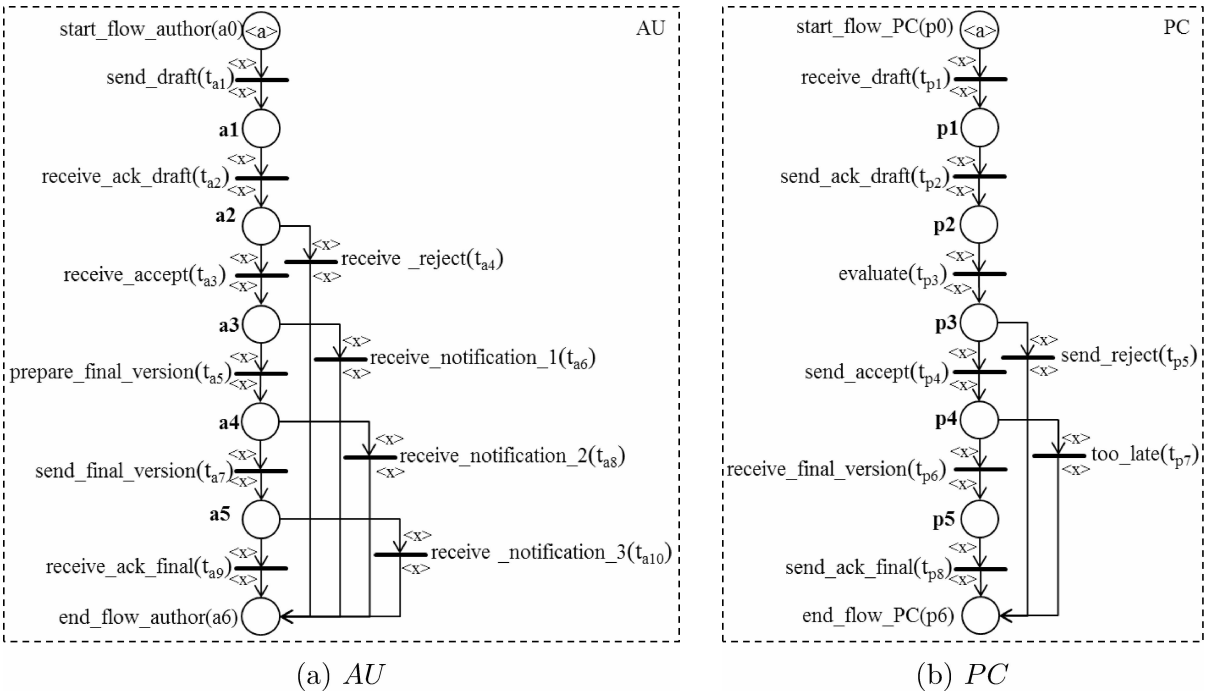


Figura 38 – Modelo de um processo que precede a apresentação de um artigo em uma conferência representado por uma IOWF-net possibilística *sound*.



(a) *AU* (b) *PC*

Figura 39 – WF-nets possibilísticas dos processos *AU* e *PC* referentes à IOWF-net possibilística da Figura 38.

lugares de ambos os modelos. Tal classe é composta pelos seguintes atributos:

- $draft$, $ackDraft$, $accept$, $reject$, $tooLate$, $finalVersion$, $ackFinal$: booleano;
- $endDraft$, $endFVersion$, $sAFinal$: booleano;
- $evaluate$: constante caractere;
- $t_{max_{s2}}$, $t_{max_{s3}}$, $t_{max_{s5}}$, $t_{max_{end}}$: constante inteira.

Observa-se que outros possíveis atributos específicos para o processo que precede a apresentação de um artigo em uma conferência podem ser definidos para este exemplo; entretanto, a não representação deles não interfere no entendimento do caso. O significado dos atributos declarados anteriormente para a classe "Artigo" é:

- o atributo $draft$, quando verdadeiro, sinaliza o envio do esboço de um artigo por parte do autor (AU) ao comitê do programa (PC);
- o atributo $ackDraft$, quando verdadeiro, sinaliza o reconhecimento por parte do PC do recebimento do esboço de um artigo;
- o atributo $accept$, quando verdadeiro, sinaliza o aceite do esboço do artigo por parte do PC;
- o atributo $reject$, quando verdadeiro, sinaliza a recusa do esboço do artigo por parte do PC;
- o atributo $tooLate$, quando verdadeiro, sinaliza o fim do prazo permitido para o envio da versão final do artigo;
- o atributo $finalVersion$, quando verdadeiro, sinaliza o envio da versão final do artigo por parte do AU ao PC;
- o atributo $ackFinal$, quando verdadeiro, sinaliza o reconhecimento por parte do PC do recebimento da versão final do artigo;
- o atributo $endDraft$, quando verdadeiro, sinaliza a conclusão do esboço do artigo por parte do AU;
- o atributo $endFVersion$, quando verdadeiro, sinaliza a conclusão da versão final do artigo por parte do AU;
- o atributo $sAFinal$, quando verdadeiro, sinaliza o recebimento da versão final do artigo por parte do AU ao PC dentro do prazo permitido;
- o atributo $evaluate$ indica o aceite (quando igual ao caractere $'a'$) ou a recusa (quando igual ao caractere $'r'$) do esboço do artigo por parte do PC;

- os atributos $t_{max_{s2}}$ e $t_{max_{s3}}$ são constantes inteiras que representam o tempo máximo permitido numa escala de tempo associada ao modelo do processo para iniciar, respectivamente, a execução das seguintes atividades: *receive_ack_draft* e *receive_accept*;

- o atributo $t_{max_{s5}}$ é uma constante inteira que representa o tempo máximo permitido numa escala de tempo associada ao modelo do processo para iniciar, respectivamente, a execução de uma das seguintes atividades: *receive_notification_1*, *receive_notification_2* ou *receive_notification_3*;

- o atributo $t_{max_{end}}$ é uma constante inteira que representa o tempo máximo permitido numa escala de tempo associada ao modelo do processo para finalizar a execução do mesmo, caso os eventos referente ao aceite e ao rejeite do esboço do artigo não sejam recebidos.

As transições t_{a2}, t_{a3}, t_{a4}, t_{a6}, t_{a8}, t_{a9}, t_{a10}, t_{p1} e t_{p6} pertencem ao conjunto de transições de saída especificado na relação AC ($T^S = \{t_{a2}, t_{a3}, t_{a4}, t_{a6}, t_{a8}, t_{a9}, t_{a10}, t_{p1}, t_{p6}\}$). Logo, as condições necessárias para que as funções de autorização relacionadas a estas transições possam ser avaliadas como *verdadeira* durante a execução do modelo são compostas pelas condições relacionadas aos lugares de comunicação e à execução do modelo do processo local. As condições necessárias para que as funções de autorização relacionadas ao restante das transições (t_{a1}, t_{a5}, t_{a7}, t_{p2}, t_{p3}, t_{p4}, t_{p5}, t_{p7} e t_{p8}) possam ser avaliadas como *verdadeira* durante a execução do modelo são compostas apenas pelas condições relacionadas à execução do modelo do processo local.

A Tabela 3 mostra as funções de autorização associadas às transições pertencentes aos modelos dos processos AU e PC representados pelas WF-nets possibilísticas apresentadas na Figura 39. As colunas com as subscrições "*verdadeira se*" e "*incerta se*" representam, respectivamente, as interpretações *verdadeira* e *incerta* associadas a cada transição através das funções de autorização. A interpretação $falsa$ não é especificada; porém a função de autorização será avaliada como $falsa$ sempre que as condições associadas tanto a interpretação *verdadeira* quanto a interpretação *incerta* sejam inválidas. Por fim, se a interpretação *incerta* não é especificada numa função de autorização, a mesma não pode ser avaliada como *incerta*.

Tabela 3 – Funções de autorização associadas às transições pertencentes aos modelos dos processos AU e PC representados pelas WF-nets possibilísticas apresentadas na Figura 39.

	verdadeira se	***incerta se***
$\eta_x(t_{a1}) =$	$x.endDraft$	
$\eta_x(t_{a2}) =$	$x.ackDraft$	$(\tau \geq t_{max_{s2}}) \wedge \neg x.ackDraft$

$\eta_x(t_{a3}) =$	$x.accept$	$(\tau \geq t_{max_{s3}}) \wedge \neg x.accept$
$\eta_x(t_{a4}) =$	$x.reject \vee \tau \geq t_{max_{end}}$	
$\eta_x(t_{a5}) =$	$x.accept \wedge \neg x.tooLate$	$(\tau > t_{max_{s3}}) \wedge$ $\neg(x.accept \vee x.tooLate)$
$\eta_x(t_{a6}) =$	$x.tooLate$	
$\eta_x(t_{a7}) =$	$x.endFVersion \wedge x.accept \wedge \neg x.tooLate$	
$\eta_x(t_{a8}) =$	$x.tooLate \wedge \neg x.endFVersion$	
$\eta_x(t_{a9}) =$	$x.ackFinal$	
$\eta_x(t_{a10}) =$	$x.tooLate \wedge x.finalVersion$	
$\eta_x(t_{p1}) =$	$x.draft$	
$\eta_x(t_{p2}) =$	$x.draft$	
$\eta_x(t_{p3}) =$	$x.ackDraft$	
$\eta_x(t_{p4}) =$	$x.evaluate = 'a'$	
$\eta_x(t_{p5}) =$	$x.evaluate = 'r'$	
$\eta_x(t_{p6}) =$	$\tau \leq t_{max_{s5}} \wedge x.finalVersion$	
$\eta_x(t_{p7}) =$	$\tau > t_{max_{s5}}$	
$\eta_x(t_{p8}) =$	$x.sAFinal$	

As transições t_{a1}, t_{a7}, t_{p2}, t_{p4}, t_{p5}, t_{p7} e t_{p8} pertencem ao conjunto de transições de entrada especificado na relação AC ($T^E = \{t_{a1}, t_{a7}, t_{p2}, t_{p4}, t_{p5}, t_{p7}, t_{p8}\}$). Logo, as ações A^c_{ta} e A^i_{ta}, definidas para cada transição $t \in T^E$, serão compostas, além das ações relacionadas à execução do modelo do processo local, pelas ações responsáveis por substituir os lugares de comunicação. Nota-se que as ações responsáveis por substituir os lugares de comunicação são inseridas tanto na ação A^c_{ta} quanto na ação A^i_{ta} caso a transição $t \in T^E$. Assim sendo, a Tabela 4 mostra as ações necessárias para o entendimento do modelo. A função $eval()$ presente na ação relacionada à transição t_{p3} é responsável por avaliar o esboço do artigo atualizando o atributo $evaluate$ para 'a' caso o esboço seja aceito ou para 'r' caso seja recusado. Note que outras possíveis ações específicas ao processo local podem ser definidas; entretanto, tal simplificação não interfere no entendimento da abordagem.

Tabela 4 – Ações associadas às transições pertencentes aos modelos dos processos AU e PC representados pelas WF-nets possibilísticas apresentadas na Figura 39.

Processo *AU*	**Processo *PC***
$A^c_{ta1} : x.draft \leftarrow verdadeiro$ $A^i_{ta1} : x.draft \leftarrow verdadeiro$	$A^c_{tp2} : ackDraft \leftarrow verdadeiro$ $A^i_{tp2} : ackDraft \leftarrow verdadeiro$
$A^c_{ta5} : endFVersion \leftarrow verdadeiro$	$A^c_{tp3} : x.eval()$
$A^c_{ta7} : finalVersion \leftarrow verdadeiro$ $A^i_{ta7} : finalVersion \leftarrow verdadeiro$	$A^c_{tp4} : accept \leftarrow verdadeiro$ $A^i_{tp4} : accept \leftarrow verdadeiro$

$A^c_{tp5} : reject \leftarrow verdadeiro$ $A^i_{tp5} : reject \leftarrow verdadeiro$
$A^c_{tp6} : sAFinal \leftarrow verdadeiro$
$A^c_{tp7} : tooLate \leftarrow verdadeiro$ $A^i_{tp7} : tooLate \leftarrow verdadeiro$
$A^c_{tp8} : ackFinal \leftarrow verdadeiro$ $A^i_{tp8} : ackFinal \leftarrow verdadeiro$

Ao observar o modelo do processo AU mostrado na Figura 39a, o comportamento esperado após o envio do esboço de um artigo ao comitê do programa, representado pelo modelo do processo PC (Figura 39b), é de receber duas mensagens a partir de PC antes que o tempo corrente alcance o tempo máximo permitido de espera. A primeira mensagem a ser recebida por AU a partir de PC se refere ao recebimento do esboço ($receive\_ack\_draft$) e a segunda mensagem ao aceite ($receive\_accept$) ou à recusa ($receive\_reject$) do mesmo. O tempo máximo permitido de espera é indicado pelos atributos $t_{max_{s2}}$, $t_{max_{s3}}$ e $t_{max_{end}}$ associados, respectivamente, às transições t_{a2}, t_{a3} e t_{a4}. Se as mensagens esperadas pelo modelo do processo AU são recebidas antes do tempo indicado pelos atributos $t_{max_{s2}}$ e $t_{max_{s3}}$ definidos para o objeto $< a >$, todos os disparos serão certos e todas as marcações precisas. Caso contrário, se o tempo corrente alcançar o valor de um destes dois atributos ($t_{max_{s2}}$ ou $t_{max_{s2}}$) relacionados a uma das atividades do objeto $< a >$ e nenhuma mensagem relacionada a tal atividade for recebida a partir do comitê de programa, alguns disparos incertos ocorrerão aumentando assim a imprecisão sobre a localização do objeto em questão. Observa-se que, a fim de evitar um estado de espera infinito caso nenhuma mensagem que permita a recuperação do estado do modelo do processo após a execução de alguns disparos incertos seja recebida, a transição t_{a4} será disparada com certeza caso o tempo corrente alcançar o valor do atributo $t_{max_{end}}$ associada a ela, permitindo assim a conclusão da execução do modelo do processo.

Um desvio do comportamento esperado pode facilmente ocorrer se, por exemplo, o comitê do programa não notifica o recebimento do esboço do artigo ou atrasa o envio da notificação referente a aceitação ou a recusa do mesmo. O autor pode continuar a preparação do artigo mesmo sem saber se ele foi aceito ou rejeitado (disparo de transição t_{a3} ou t_{a4}).

A fim de ilustrar a execução de um possível comportamento anormal, considere as seguintes informações:

- o elemento τ representa o tempo corrente;
- os atributos no tempo corrente $\tau = 0$ são inicializados com os seguintes valores:
  - $draft = finalVersion \leftarrow falso$;
  - $ackDraft = accept = reject = tooLate = ackFinal \leftarrow falso$;

- $endFVersion = sAFinal \leftarrow falso$;
- $endDraft \leftarrow verdadeiro$;
- $evaluate \leftarrow ''$;
- $t_{max_{s2}} \leftarrow 25$;
- $t_{max_{s3}} \leftarrow 60$;
- $t_{max_{s5}} \leftarrow 95$;
- $t_{max_{end}} \leftarrow 300$.

❑ a função $eval()$ atualiza o atributo $evaluate$ para $'a'$;

❑ devido a uma falha de comunicação, a atualização do valor do atributo ack_draft para $verdadeiro$ realizada na transição t_{p2} não é bem sucedida, ou seja, durante toda a execução do modelo do processo o mesmo é avaliado como $falso$.

Levando em conta as informações acima descritas, os modelos dos processos AU e PC representados pelas WF-nets possibilísticas apresentadas na Figura 39 são simulados de acordo com o "jogador" apresentado na Figura 29, ou seja, disparos incertos e marcações imprecisas são consideradas. A simulação é detalhada abaixo:

❑ no tempo corrente $\tau = 0$, o esboço do artigo está concluído, ou seja, o atributo $endDraft = verdadeiro$ e, consequentemente, a função de autorização associada à transição t_{a1} é avaliada como $verdadeira$ ($\eta_{<a>}(t_{a1}) = verdadeira$). Dado que t_{a1} está habilitada por uma marcação precisa, ela é disparada com certeza (Figura 40a) e a ação A^c_{ta1} é executada, alterando o valor do atributo $draft$ para $verdadeiro$;

❑ no tempo corrente $\tau = 1$, PC é notificado da recepção do esboço de um artigo, ou seja, o atributo $draft = verdadeiro$ e, consequentemente, a função de autorização associada à transição t_{p1} é avaliada como $verdadeira$ ($\eta_{<a>}(t_{p1}) = verdadeira$). Dado que t_{p1} está habilitada por uma marcação precisa, ela é disparada com certeza e, caso fosse definida, a ação A^c_{tp1} seria executada (Figura 40b);

❑ no tempo corrente $\tau = 10$, PC enviará a AU uma notificação referente ao recebimento do esboço do artigo. Considerando que a transição t_{p2} está habilitada por uma marcação precisa e que a função de autorização associada a ela é avaliada como $verdadeira$ ($\eta_{<a>}(t_{p2}) = verdadeira$), t_{p2} é disparada com certeza (Figura 40c) e a ação A^c_{tp2} é executada. Entretanto, como mencionado anteriormente, o valor do atributo $ackDraft$ não será alterado para $verdadeiro$, permanecendo, assim, $falso$ até o fim desta execução;

❑ no tempo corrente $\tau = 25$, devido a uma falha de comunicação entre os modelos dos processos, o atributo $ackDraft$ não foi atualizado para $verdadeiro$. Logo, a transição t_{a2} está habilitada por uma marcação precisa e a sua função de autorização está avaliada como $incerta$ ($\eta_{<a>}(t_{a2}) = incerta$); então t_{a2} é disparada com incerteza,

ou seja, uma cópia do objeto $< a >$ é produzido no lugar $a2$ sem que o objeto $< a >$ seja removido do lugar $a1$ (Figura 40d) e, caso fosse definida, a ação A^i_{ta2} seria executada. O disparo incerto foi realizado para que a falha de comunicação ocorrida possa ser contornada de forma a manter consistente o estado do modelo do processo;

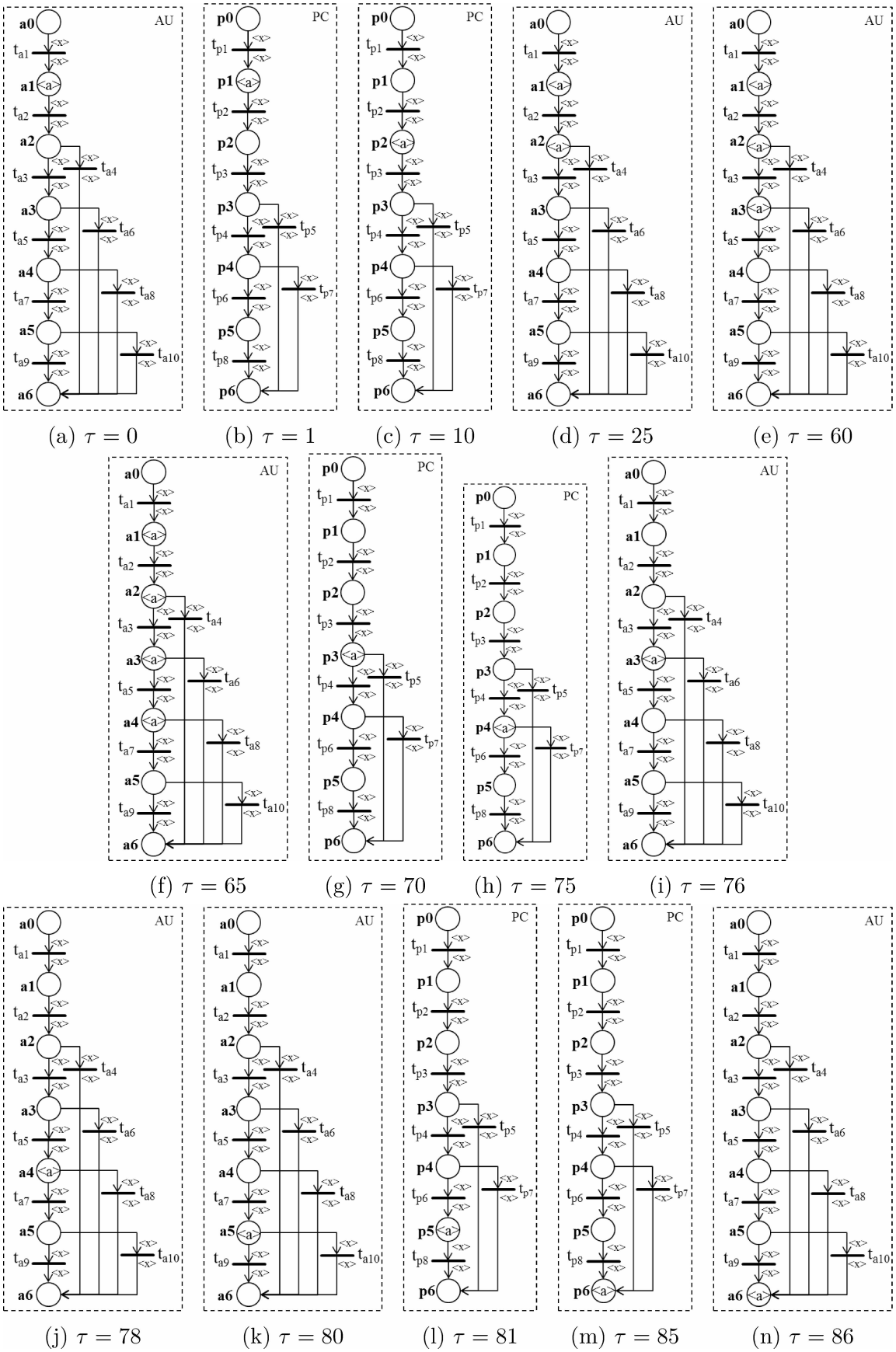
- no tempo corrente $\tau = 60$, nenhuma confirmação sobre o recebimento do esboço do artigo ou sobre o aceite ou a recusa do mesmo foi recebida, ou seja, tanto o atributo $ackDraft$ quanto os atributos $accept$ e $reject$ continuam iguais a $falso$ para o objeto $< a >$. Entretanto, neste momento, a função de autorização associada à transição t_{a3} é avaliada como $incerta$ ($\eta_{<a>}(t_{a3}) = incerta$) e a mesma está habilitada por uma marcação imprecisa. Tal situação permite disparar de forma incerta a transição t_{a3}, produzindo assim uma cópia do objeto $< a >$ no lugar $a3$ sem que o objeto $< a >$ seja removido do lugar $a2$ (Figura 40e). E, caso fosse definida, a ação A^i_{ta3} seria executada;

- no tempo corrente $\tau = 65$, constata-se que nenhuma resposta por parte de PC até então foi recebida, ou seja, os atributos $ackDraft$, $accept$, $reject$ e too_late continuam iguais a $falso$ para o objeto $< a >$. Contudo, neste momento, a transição t_{a5} está habilitada por uma marcação imprecisa e a função de autorização associada a ela se torna $incerta$ ($\eta_{<a>}(t_{a5}) = incerta$). Logo, um disparo incerto de t_{a5} é realizado, produzindo uma cópia do objeto $< a >$ no lugar $a4$ sem removê-lo do lugar de entrada $a3$ (Figura 40f) e, caso fosse definida, a ação A^i_{ta5} seria executada;

- no tempo corrente $\tau = 70$, o esboço do artigo é avaliado pelo PC, ou seja, a transição t_{p3} é disparada com certeza (Figura 40g) considerando que a mesma está habilitada por uma marcação precisa e sua função de autorização está avaliada como $verdadeira$ ($\eta_{<a>}(t_{p3}) = verdadeira$). Neste disparo, a ação A^c_{tp3} é executada, alterando o valor do atributo $evaluate$ para $'a'$ como anteriormente definido;

- no tempo corrente $\tau = 75$, de acordo com a avaliação realizada no disparo anterior, o esboço do artigo foi aceito, ou seja, o atributo $evaluate$ é igual à $'a'$ e, consequentemente, a função de autorização associada à transição t_{p4} é avaliada como $verdadeira$ ($\eta_{<a>}(t_{p4}) = verdadeira$). Dado que a transição t_{p4} está habilitada por uma marcação precisa, ela é disparada com certeza (Figura 40h) e a ação A^c_{tp4} é executada, alterando o valor do atributo $accept$ para $verdadeiro$;

- no tempo corrente $\tau = 76$, a confirmação sobre o aceite do esboço do artigo é recebida, ou seja, o atributo $accept = verdadeiro$ e, consequentemente, a função de autorização associada à transição t_{a3} é avaliada como $verdadeira$ ($\eta_{<a>}(t_{a3}) = verdadeira$). Levando em consideração que a transição t_{a3} está habilitada por uma marcação imprecisa, visto que as transições t_{a2} e t_{a5} foram disparadas de forma



(a) $\tau = 0$ (b) $\tau = 1$ (c) $\tau = 10$ (d) $\tau = 25$ (e) $\tau = 60$

(f) $\tau = 65$ (g) $\tau = 70$ (h) $\tau = 75$ (i) $\tau = 76$

(j) $\tau = 78$ (k) $\tau = 80$ (l) $\tau = 81$ (m) $\tau = 85$ (n) $\tau = 86$

Figura 40 – Resultados da simulação - Falhas de Comunicação.

incerta devido à não confirmação do recebimento do esboço do artigo por parte de PC à AU e, além disto, do atraso na resposta referente ao aceite do mesmo, o algoritmo de recuperação descrito no Algoritmo 1 é chamado para cancelar os disparos incertos de t_{a3} e t_{a5} e finalizar o de t_{a2}. Durante a execução deste algoritmo, caso definidas, as ações A^l_{ta3} e A^l_{ta5} seriam executadas no momento do cancelamento do disparo incerto da transição em questão e a ação A^f_{ta2} no momento da finalização do disparo incerto da transição t_{a2}. Após concluir a execução do algoritmo de recuperação, a transição t_{a3} é disparada com certeza dado que está habilitada por uma marcação precisa e que a função de autorização associada a ela está avaliada como *verdadeira* (Figura 40i).

- no tempo corrente $\tau = 78$, a transição t_{a5} está habilitada por uma marcação precisa e sua função de autorização está avaliada como *verdadeira* ($\eta_{<a>}(t_{a5}) = verdadeira$). Uma vez que o esboço do artigo foi aceito e que o tempo limite para o envio da versão final ainda não foi alcançado, ou seja, o atributo $aceite = verdadeiro$ e o atributo $too\_late = falso$, a transição t_{a5} é disparada com certeza (Figura 40j) e a ação A^c_{ta5} responsável por atualizar o atributo $endFVersion$ para *verdadeiro* é executada;

- no tempo corrente $\tau = 80$, a versão final do artigo é enviada a PC, ou seja, como a transição t_{a7} está habilitada por uma marcação precisa e sua função de autorização está avaliada como *verdadeira* ($\eta_{<a>}(t_{a7}) = verdadeira$), dado que os atributos *accept* e $endFVersion$ são iguais a *verdadeiro* e o atributo *too_late* a $falso$, t_{a7} é disparada com certeza (Figura 40k) e a ação A^c_{ta7} é executada, alterando assim o valor do atributo $finalVersion$ para *verdadeiro*;

- no tempo corrente $\tau = 81$, a versão final do artigo é recebida por PC, ou seja, o atributo $finalVersion = verdadeiro$ e, consequentemente, a função de autorização associada à transição t_{p6} é avaliada como *verdadeira* ($\eta_{<a>}(t_{p6}) = verdadeira$). Como a transição t_{p6} está habilitada por uma marcação precisa, t_{p6} é disparada com certeza (Figura 40l) executando a ação A^c_{tp6} responsável por atualizar o atributo $sAFinal$ para *verdadeiro*;

- no tempo corrente $\tau = 85$, PC notifica AU sobre o recebimento da versão final do artigo, ou seja, como a transição t_{p8} está habilitada por uma marcação precisa e sua função de autorização está avaliada como *verdadeira* ($\eta_{<a>}(t_{p8}) = verdadeira$), t_{p8} é disparada com certeza (Figura 40m) e a ação A^c_{p8} é executada, alterando o valor do atributo $ackFinal$ para *verdadeiro*;

- finalmente, no tempo corrente $\tau = 86$, a confirmação do recebimento da versão final do artigo por parte de PC é recebida por AU, ou seja, o atributo $ackFinal = verdade$ e, consequentemente, a função de autorização associada à transição t_{a9} é

avaliada como *verdadeira* ($\eta_{<a>}(t_{a9}) = verdadeira$). Assim sendo, t_{a9} é disparada com certeza (Figura 40n) e, caso fosse definida, a ação A^c_{ta9} seria executada. Após o disparo certo de t_{a9}, a execução do modelo do processo é finalizada com sucesso.

Se considerar, durante a simulação anteriormente descrita, a IOWF-net definida na Seção 2.3 em vez da IOWF-net possibilística, uma inconsistência é ocasionada no momento em que a função de autorização associada à transição t_{a3} se torna *verdadeira* quando o lugar $a2$ ainda não está marcado e a função de autorização associada à t_{a2} é avaliada como falsa. Assim a transição t_{a3} não é habilitada e consequentemente não é disparada. A Figura 41 destaca tal inconsistência com o objeto $< a >$ localizado no lugar $a1$, nenhum objeto presente no lugar $a2$, a função de autorização $\eta_{<a>}(t_{a2})$ avaliada como falsa e a função de autorização $\eta_{<a>}(t_{a3})$ avaliada como *verdadeira*. Na prática, tal inconsistência pode ser tratada e corrigida por um funcionário, ou seja, o estado corrente do modelo na fase de execução deve ser manualmente ajustado para corrigir a marcação da rede, isto é, o estado do processo.

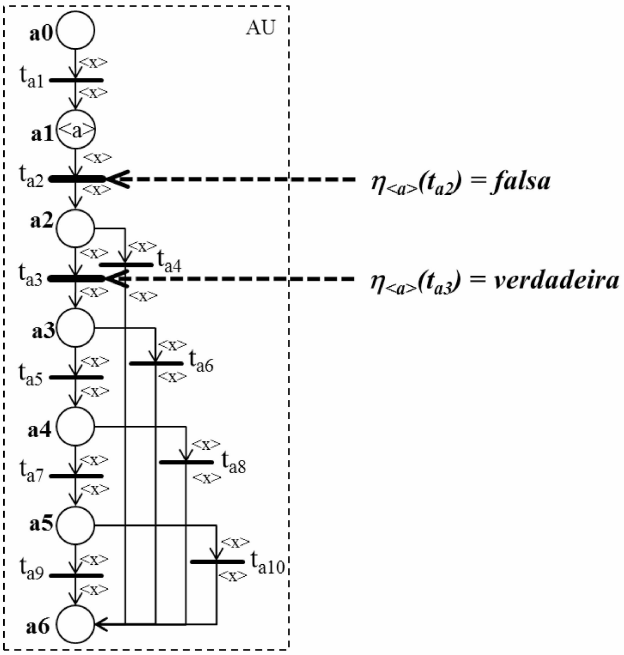


Figura 41 – Possível inconsistência no modelo do processo AU.

Para que um modelo do processo cuja execução seja conduzida por um "jogador" de rede de Petri considere desvios em relação ao modelo definido sem a ocorrência de qualquer tipo de inconsistência, novas transições devem então ser inseridas no modelo para que o mesmo considere todos os possíveis comportamentos anormais. Entretanto uma consequência desta modificação é tornar o modelo correspondente mais complexo e prejudicar sua legibilidade.

A noção de disparo incerto neste problema é considerado com o intuito de incorporar uma certa flexibilidade ao modelo do processo de forma a torná-lo robusto quando não se tem um total controle sobre a geração dos eventos. Desta forma, situações imprevistas

ocasionadas por desvios de comportamento durante a execução do modelo do processo interorganizacional podem ser devidamente tratados.

5.3 Prevenção de *Deadlocks* na *Workflow net* Interorganizacional *Relaxed Sound*

A necessidade de sincronização de processos paralelos pode levar à situações de *deadlock*, às quais, em sua grande maioria, são causadas pela própria estrutura dos processos de negócio. Caso o processo permita a ocorrência de *deadlocks*, a única solução para evitá-los sem que o modelo do processo seja explicitamente modificado é evitar a(s) sequência(s) de disparo responsável(is) por conduzir o estado do modelo ao *deadlock*, e utilizar uma outra sequência que seja capaz de alcançar seu estado final. Assim sendo, a fim de evitar as situações de *deadlock* nos modelos dos processos de negócio interorganizacionais sem que os mesmos sejam alterados, a IOWF-net possibilística através da noção de disparo incerto é usada com o objetivo de se desviar, em tempo de execução, das sequências responsáveis pelos estados de *deadlock*. Observa-se que o modelo do processo deve respeitar a propriedade *Relaxed Soundness* para que se possa garantir que cada transição pertence a pelo menos uma sequência de disparo responsável por conduzir a execução do modelo do processo ao seu estado final.

Para que uma situação de *deadlock* seja evitada, inicialmente uma análise deve ser realizada na estrutura (topologia) do modelo do processo para determinar o sifão mínimo responsável por tal situação. Após encontrá-lo, as funções de autorização, associadas às transições responsáveis diretamente e indiretamente pelo esvaziamento do mesmo e, consequentemente, pelo estado de *deadlock*, são definidas apenas pelas interpretações *incerta* e $falsa$, ou seja, tais funções de autorização nunca poderão ser avaliadas como $verdadeira$. Isto permite um raciocínio para frente (*forward-chaining*), explorando possíveis alternativas de disparo, de forma a evitar o possível esvaziamento do sifão.

Para realizar a análise estrutural do processo de negócio interorganizacional *Relaxed Sound* este é primeiramente modelado por uma IOWF-net. Após obter para cada estado de *deadlock* presente no modelo do processo o sifão corresponde e, consequentemente, as transições responsáveis diretamente e indiretamente por tal estado, uma IOWF-net possibilística é então definida. O procedimento responsável por identificar o(s) sifão(ões) e a(s) transição(ões) é descrito a seguir:

1. determinar o conjunto de sifões mínimos SM presentes no modelo do processo:

    a) tornar a IOWF-net repetitiva ou parcialmente repetitiva, ou seja, adicionar uma transição SR com arcos de entrada saindo dos lugares o e com arcos de saída entrando nos lugares i de cada processo modelado por uma WF-net;

b) utilizar um programa aplicativo para detectar automaticamente o conjunto SM de sifões mínimos;

c) para cada sifão mínimo $sm \in SM$, se sm possui uma *trap* associada a ele, então sm deve ser removido do conjunto SM ($SM \leftarrow SM - \{sm\}$);

2. determinar o conjunto de sifões SM que conduzem à um estado de *deadlock*:

a) gerar o grafo das marcações acessíveis G_A;

b) definir o conjunto de estados de *deadlock* E_d pertencentes à G_A;

c) definir, para cada $ed \in E_d$, o conjunto de sequências de disparo C_d pertencentes à G_A responsáveis por conduzir o estado do modelo do processo à ed. Neste caso, a ordem das transições em uma sequência de disparo $cd \in C_d$ não importa;

d) para cada sifão mínimo $sm \in SM$, faça:

i. definir o conjunto de sequência(s) de disparo C_s responsável(is) por esvaziar o sifão sm. Neste caso, a ordem das transições em uma sequência de disparo $cs \in C_s$ não importa;

ii. verificar se todas as sequências de disparo $cs \in C_s$ estão contidas em alguma sequência de disparo $cd \in C_d$, isto é, $\forall cs \in C_s, \exists cd \in C_d | cs \subseteq cd$. Caso não exista, remover o sifão sm do conjunto SM ($SM \leftarrow SM - \{sm\}$);

3. determinar o conjunto T^D das transições responsáveis diretamente e indiretamente pelo estado de *deadlock*:

a) para cada estado de *deadlock* $ed \in E_d$, definir um sifão $sm \in SM$ que o represente;

b) para cada $t \in T^{\blacktriangle}$, se t é responsável diretamente por esvaziar sm, então t é adicionada ao conjunto T^D ($T^D \leftarrow T^D \cup \{t\}$);

c) para cada $t \in T^{\blacktriangle}$, se t está habilitada por uma marcação imprecisa e o seu disparo produz fichas em pelo menos um lugar que não pertence à sm, então t é adicionada ao conjunto T^D ($T^D \leftarrow T^D \cup \{t\}$);

d) para cada transição $t \in T^D$, se ao disparar t o lugar de saída o do modelo do processo local correspondente é marcado, indicando a sua correta conclusão, então t deve ser removido do conjunto T^D ($T^D \leftarrow T^D - \{t\}$).

Após realizar a análise estrutural, a IOWF-net possibilística referente ao processo interorganizacional é então definida. Considerando isto, as w WF-nets possibilísticas que compõem a IOWF-net possibilística são definidas de acordo com as etapas de modelagem especificadas na Definição 34. Por fim, as etapas 3, 4, 5 e 6, especificadas na definição 36, são realizadas com o objetivo de definir a IOWF-net possibilística.

Tendo em vista que as funções de autorização associadas às transições responsáveis diretamente e indiretamente pelo estado de *deadlock*, representadas pelo conjunto T^D, só podem ser compostas pelas interpretações *incerta* e $falsa$, as w WF-nets possibilísticas são então parcialmente modificadas. Esta modificação ocorre na função de autorização associada a cada transição $t \in T^D$, isto é, para cada transição $t \in T^D$, a interpretação *verdadeira* definida na função de autorização associada a t é removida e as suas condições são inseridas por meio de uma conjunção lógica na interpretação *incerta* associada a t.

Ao fim da definição da IOWF-net possibilística, as w WF-nets possibilísticas podem então ser executadas de acordo com a etapa 7 descrita na Definição 36. Nota-se que os disparos incertos finalizados referem-se a uma sequência de disparo livre de *deadlock* e os cancelados a uma sequência de disparo que pode conduzir a um estado de *deadlock*.

Para ilustrar a execução de um modelo de processo com uma situação de *deadlock*, o processo que precede a apresentação de um artigo em uma conferência apresentado na Seção 2.3 é utilizado. A IOWF-net apresentada na Figura 27 não respeita a propriedade *Soundness* mas sim a *Relaxed Soundness*. Assim sendo, alguns problemas de sincronização podem surgir, permitindo a ocorrência de *deadlocks* estruturais durante a execução do modelo do processo.

O primeiro passo para tornar o modelo do processo livre de *deadlocks* durante a sua execução é determinar os sifões mínimos presentes no mesmo. Entretanto uma condição necessária para encontrar sifões em uma rede de Petri é que a mesma seja repetitiva ou parcialmente repetitiva. Assim sendo, uma nova transição SR com arcos de entrada saindo dos lugares end_flow_author e end_flow_PC e arcos de saída entrando nos lugares $start\_flow\_author$ e $start\_flow\_PC$ é acrescentada à Figura 27. O modelo considerando uma rede de Petri repetitiva equivalente é apresentada na Figura 42.

O foco deste trabalho não é apresentar um novo algoritmo para encontrar sifões, assim o programa aplicativo *PIPE* é utilizado para a detecção automática destas estruturas. Por meio do programa, 24 sifões mínimos são encontrados dos quais 10 podem ser esvaziados, ou seja, 14 sifões possuem uma *trap* associada a eles impossibilitando-os de se esvaziarem.

Os 24 sifões encontrados são representados na Tabela 5. A primeira coluna da tabela refere-se ao identificador do sifão descrito na segunda coluna. A presença do símbolo "•" na terceira coluna da tabela indica que o sifão em questão possui uma *trap* associada.

Tabela 5 – Sifões mínimos e *traps* referentes à Figura 42.

ID	**Siphon**	**Trap**
H_1	$start\_flow\_PC$, p1, p2, p3, p4, p5, p6	•
H_2	a6, reject, too_late, ack_final, $start\_flow\_PC$, p1, p2, p3, p4, p5	•
H_3	a3, a4, a5, a6, $accept$, $reject$, $start\_flow\_PC$, p1, p2, p3	•
H_4	a2, a3, a4, a5, a6, ack_draft, $start\_flow\_PC$, p1	•

H_5	*start_flow_AU*, a1, a2, a3, a4, a5, a6	•
H_6	*start_flow_AU*, a2, a3, a4, a5, a6, *draft*, *ack_draft*, p1	•
H_7	*start_flow_AU*, a3, a4, a5, a6, *draft*, *accept*, *reject*, p1, p2, p3	•
H_8	*start_flow_AU*, a1, a2, a3, a4, a6, *ack_final*, *final_version*, p5	•
H_9	*start_flow_AU*, *draft*, p1, p2, p3, p4, p5, p6	•
H_{10}	*start_flow_AU*, a2, a3, a4, a6, *draft*, *ack_draft*, *final_version*, *ack_final*, p1, p5	•
H_{11}	a3, a4, a6, *accept*, *reject*, *final_version*, *ack_final*, *start_flow_PC*, p1, p2, p3, p5	•
H_{12}	*start_flow_AU*, a3, a4, a6, *draft*, *accept*, *reject*, *final_version*, *ack_final*, p1, p2, p3, p5	•
H_{13}	*start_flow_AU*, a6, *draft*, *reject*, *too_late*, *ack_final*, p1, p2, p3, p4, p5	•
H_{14}	a2, a3, a4, a6, *ack_draft*, *final_version*, *ack_final*, *start_flow_PC*, p1, p5	•
H_{15}	*start_flow_AU*, a3, a4, a6, *draft*, *accept*, *reject*, *ack_final*, p1, p2, p3, p4, p5	
H_{16}	a3, a4, a6, *accept*, *reject*, *ack_final*, *start_flow_PC*, p1, p2, p3, p4, p5	
H_{17}	*start_flow_AU*, a1, a2, a3, a4, a6, *ack_final*, *start_flow_PC*, p1, p2, p3, p4, p5	
H_{18}	*start_flow_AU*, a1, a2, a3, a4, a6, *draft*, *ack_final*, p1, p2, p3, p4, p5	
H_{19}	*start_flow_AU*, a2, a3, a4, a6, *draft*, *ack_draft*, *ack_final*, p1, p2, p3, p4, p5	
H_{20}	a2, a3, a4, a6, *ack_draft*, *ack_final*, *start_flow_PC*, p1, p2, p3, p4, p5	
H_{21}	*start_flow_AU*, a1, a2, a6, *too_late*, *ack_final*, *start_flow_PC*, p1, p2, p3, p4, p5	
H_{22}	*start_flow_AU*, a1, a2, a6, *draft*, *too_late*, *ack_final*, p1, p2, p3, p4, p5	
H_{23}	*start_flow_AU*, a2, a6, *draft*, *ack_draft*, *too_late*, *ack_final*, p1, p2, p3, p4, p5	
H_{24}	a2, a6, *ack_draft*, *too_late*, *ack_final*, *start_flow_PC*, p1, p2, p3, p4, p5	

Embora existam 10 sifões mínimos sem *trap* presentes no modelo, nem todos necessariamente ficam livre de fichas. O esvaziamento dos mesmos também depende do compor-

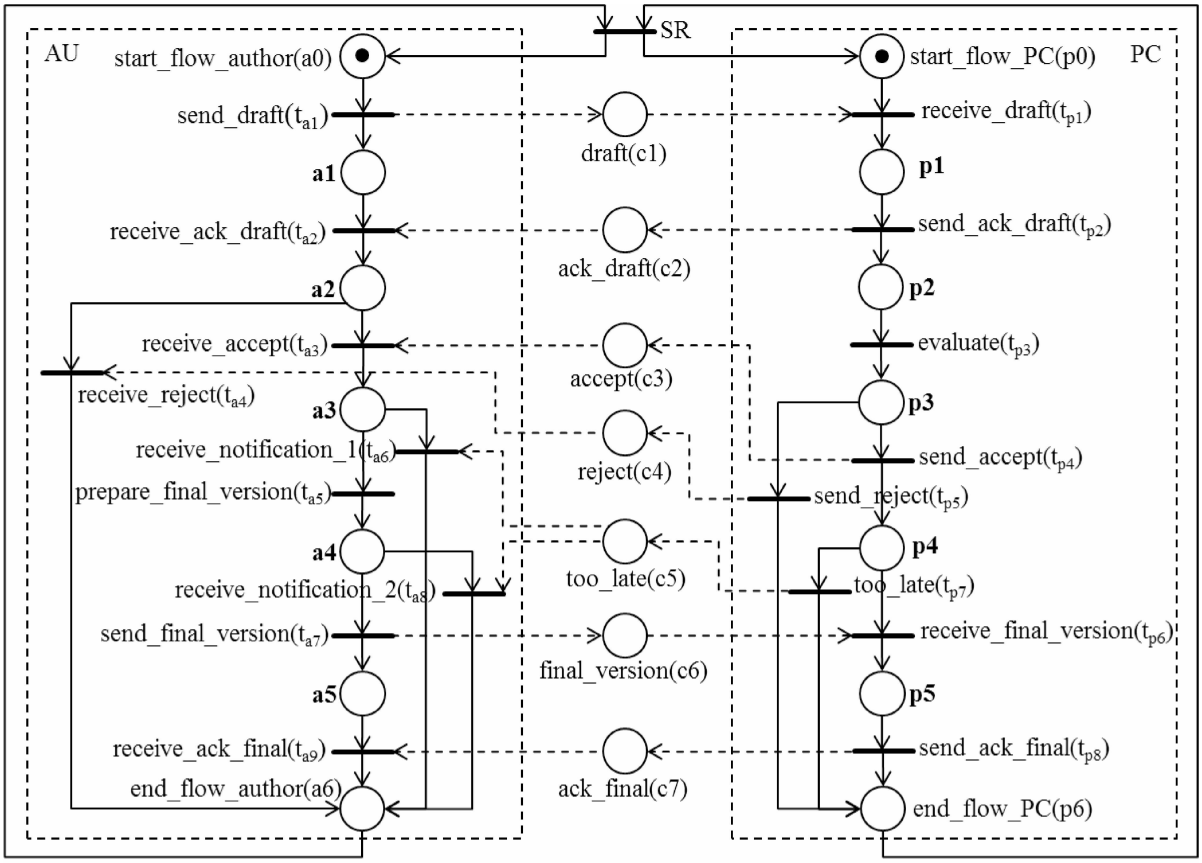


Figura 42 – Modelo representado por uma rede de Petri repetitiva.

tamento global do modelo. Assim sendo, o grafo das marcações acessíveis que permite uma visualização gráfica dos estados alcançáveis a partir do estado inicial é apresentado na Figura 43. Ao observá-lo é possível verificar que existe apenas um estado de *deadlock*, o qual está destacado em cinza. Logo, o conjunto de estados de *deadlock* E_d é composto apenas por um elemento, ou seja, $E_d = \{M_{19}\}$.

O estado de *deadlock* M_{19} é alcançado a partir do estado inicial através de 13 sequências de disparo diferentes. Entretanto, a diferença entre estas 13 sequências advém apenas da ordem de disparo das transições. Logo, ignorando esta ordem de disparo, das 13 sequências encontradas, pode-se considerar apenas uma, ou seja, o conjunto C_d é composto apenas por uma sequência de disparo: $C_d = \{t_{a1}, t_{a2}, t_{a3}, t_{a5}, t_{a7}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}\}$. A partir desta informação, os 10 sifões mínimos sem *trap* serão analisados para verificar se a(s) sequência(s) de disparo que o esvaziam, deixando-o permanentemente livre de fichas, está(ão) contidas em C_d.

As sequências de disparo C_s responsáveis pelo esvaziamento das fichas nos sifões sem *trap* são descritas a seguir:

$$H_{15} :: C_s^{15} = \{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}, t_{a3}, t_{a5}, t_{a7}\};$$
$$H_{16} :: C_s^{16} = \{t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}, t_{a3}, t_{a5}, t_{a7}\};$$
$$H_{17} :: C_s^{17} = [\{t_{a1}, t_{a2}, t_{a3}, t_{a5}, t_{a7}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}\},$$

$$\{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\}];$$

$$H_{18} :: C_s^{18} = [\{t_{a1}, t_{a2}, t_{a3}, t_{a5}, t_{a7}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}\},$$
$$\{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\}];$$

$$H_{19} :: C_s^{19} = [\{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\},$$
$$\{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\}];$$

$$H_{20} :: C_s^{20} = [\{t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p7}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\},$$
$$\{t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{a2}, t_{a3}, t_{a5}, t_{a7}\}];$$

$$H_{21} :: C_s^{21} = \{t_{a1}, t_{a2}, t_{a3}, t_{p1}, t_{p2}, t_{p3}, t_{p5}\};$$

$$H_{22} :: C_s^{22} = \{t_{a1}, t_{a2}, t_{a3}, t_{p1}, t_{p2}, t_{p3}, t_{p5}\};$$

$$H_{23} :: C_s^{23} = \{t_{a1}, t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{p1}, t_{a2}, t_{a3}\};$$

$$H_{24} :: C_s^{24} = \{t_{p1}, t_{p2}, t_{p3}, t_{p5}, t_{a2}, t_{a3}\}$$

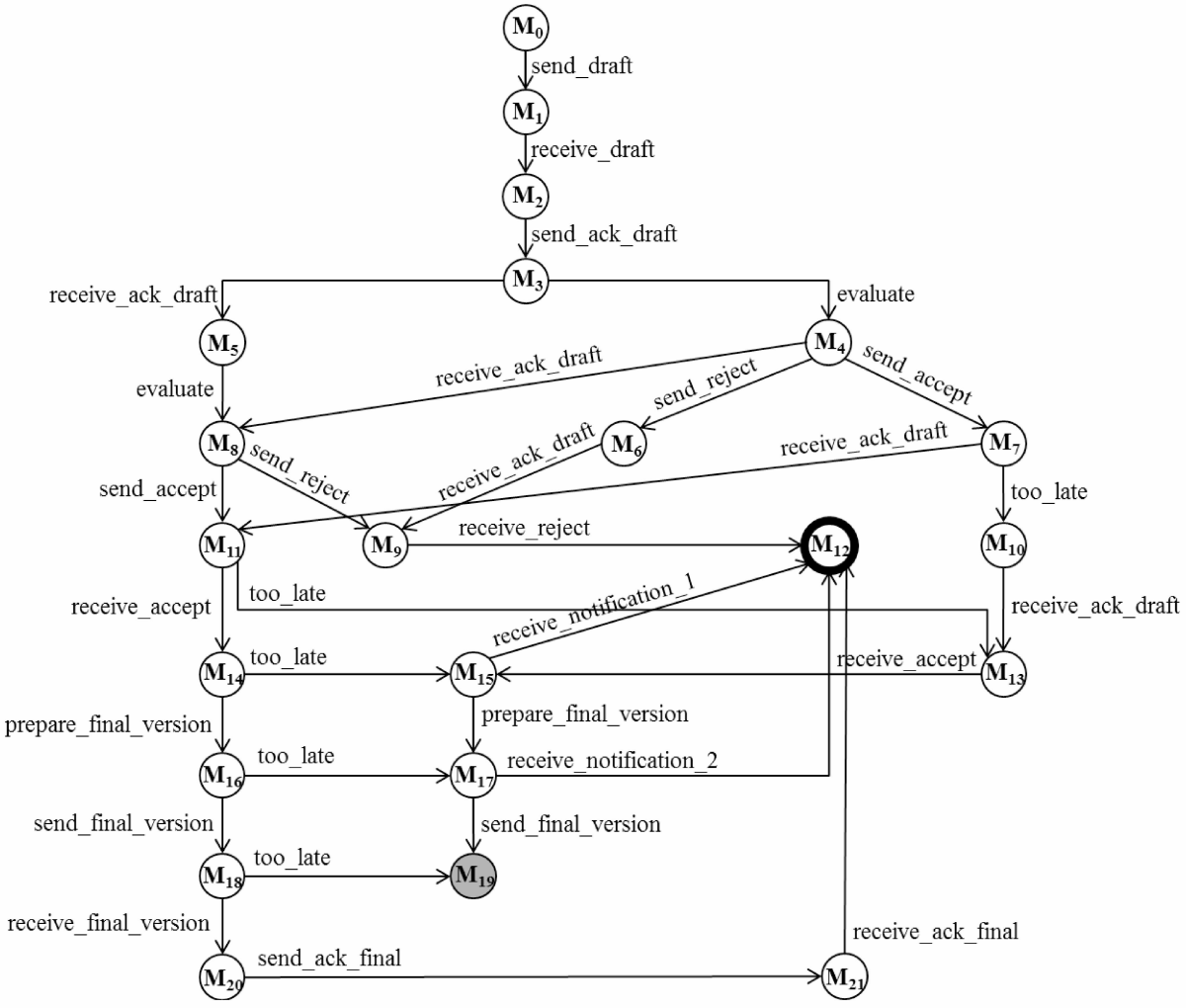
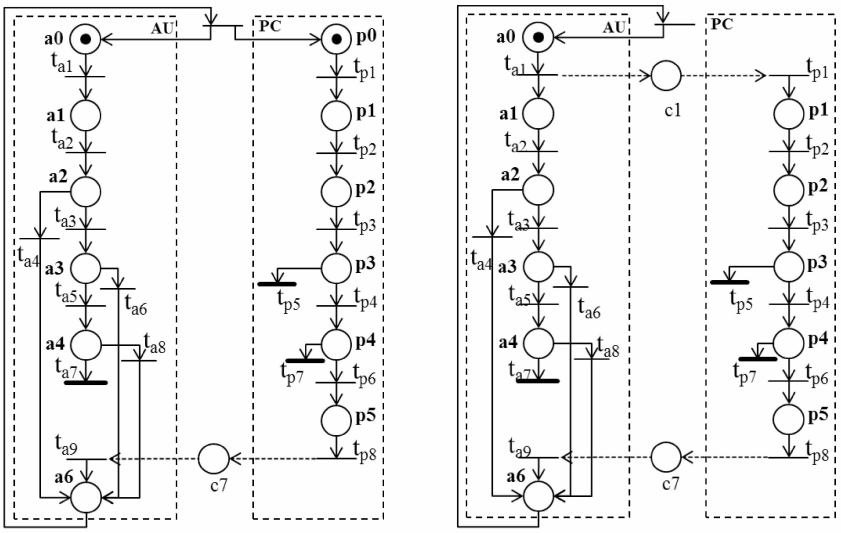


Figura 43 – Grafo das marcações acessíveis da rede de Petri repetitiva da Figura 42.

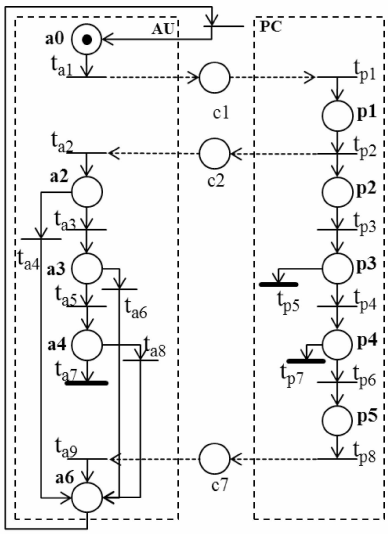
Pode-se notar, ao observar as sequências de disparo, que os sifões H_{17}, H_{18}, H_{19} e H_{20}, representados graficamente na Figura 44, contém mais de uma sequência de disparo que o esvaziam. Entretanto, cada um destes 4 sifões contém uma sequência de disparo que não esta diretamente relacionada ao estado de *deadlock* pelo conjunto E_d, ou seja, a sequência

verificada não está contida em nenhuma sequência pertencente ao conjunto C_d. Logo, os sifões H_{17}, H_{18}, H_{19} e H_{20} serão desconsiderados.

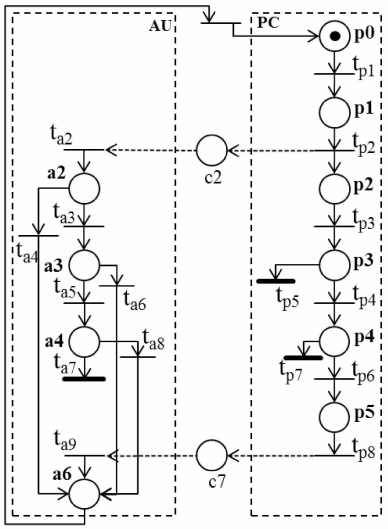


(a) H_{17}

(b) H_{18}



(c) H_{19}



(d) H_{20}

Figura 44 – Sifões mínimos sem *trap* referentes à Figura 42 com pelo menos uma sequência de disparo que não conduz o modelo do processo a um estado de *deadlock*.

Nenhuma das sequências de disparo que esvaziam os sifões H_{21}, H_{22}, H_{23} e H_{24}, representados graficamente na Figura 45, estão diretamente relacionadas ao estado de *deadlock* definido pelo conjunto E_d, ou seja, estas sequências não estão contidas em nenhuma sequência pertencente ao conjunto C_d. Logo os sifões H_{21}, H_{22}, H_{23} e H_{24} também serão desconsiderados.

Por fim, os 2 sifões mínimos sem *trap* restantes, H_{15}, H_{16}, representados graficamente na Figura 46, estão diretamente relacionadas ao estado de *deadlock* definido pelo conjunto E_d, ou seja, estas sequências estão contidas na sequência pertencente ao conjunto C_d.

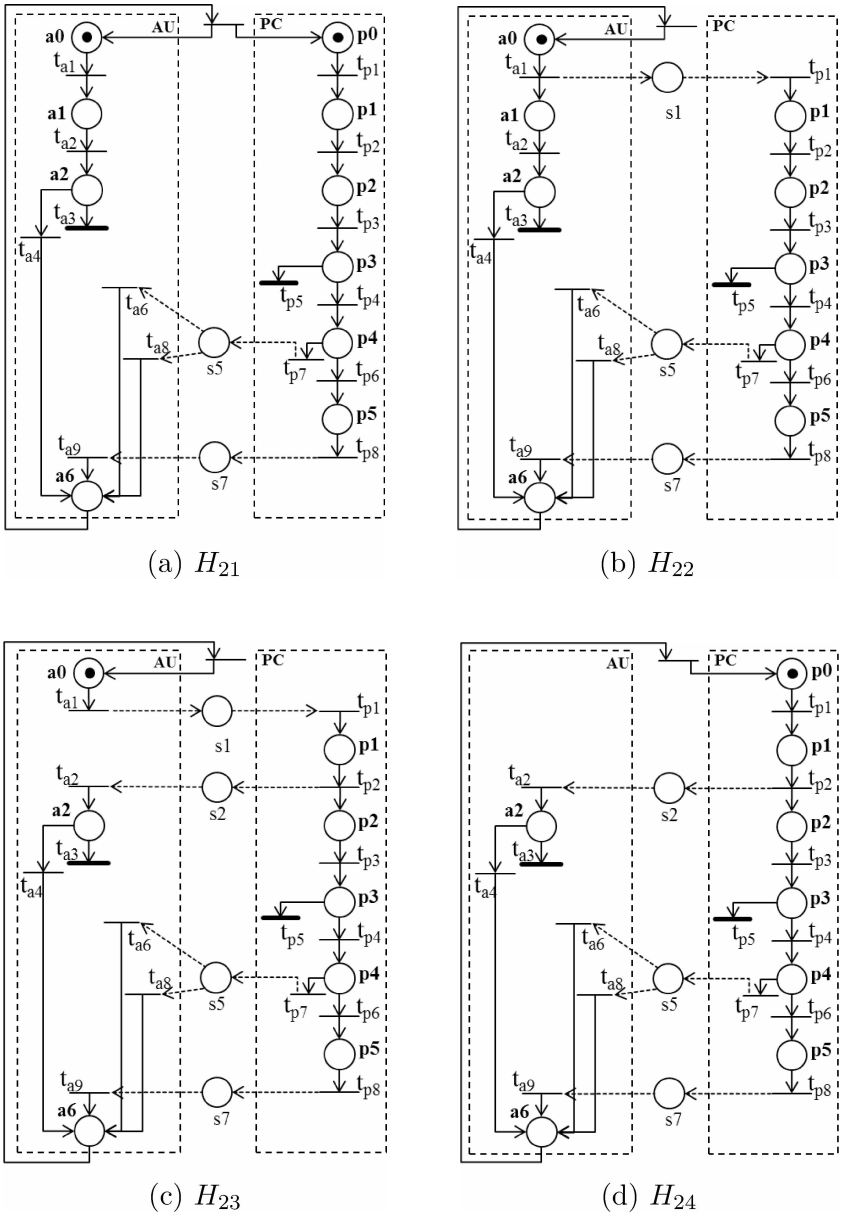
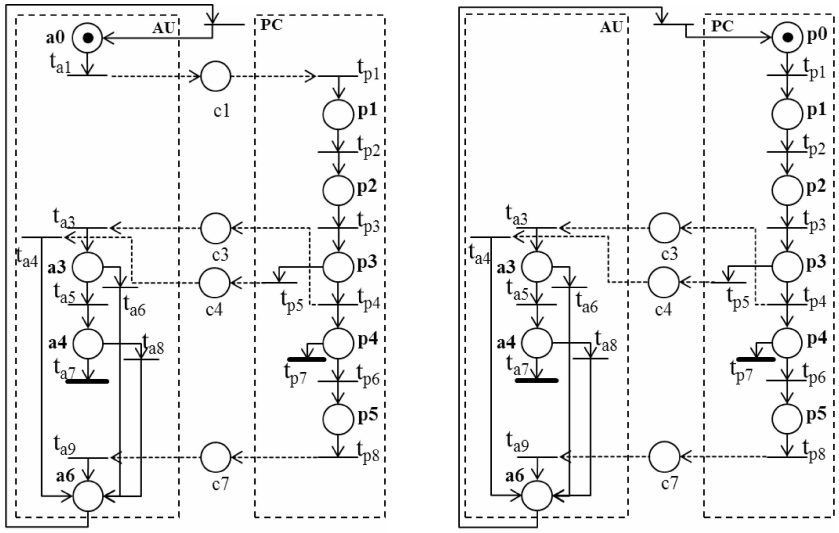


Figura 45 – Sifões mínimos sem *trap* referentes à Figura 42 que não conduzem o modelo do processo a um estado de *deadlock*.

Logo, o conjunto de sifões SM que conduzem o estado do processo a um estado de *deadlock* é composto por ambos sifões, isto é, $SM = \{H_{15}, H_{16}\}$.

Apesar do fato que os sifões H_{15} e H_{16} pertencentes ao conjunto SM caracterizam uma mesma situação de *deadlock*, é possível verificar a existência de uma particularidade entre cada um deles. Tal particularidade é a marcação inicial, dado que no sifão H_{15} a marcação inicial é uma ficha no lugar $a0$ e no sifão H_{16} a marcação inicial é apenas uma ficha no lugar $p0$. Levando em consideração que a diferença entre ambos é relacionada apenas à marcação inicial, pode-se desconsiderar o sifão H_{16} e considerar apenas o sifão H_{15}.

Após determinar o sifão H_{15} como o responsável pela situação de *deadlock* definido pelo conjunto E_d, o conjunto T^D das transições responsáveis diretamente e indiretamente ao



(a) H_{15} (b) H_{16}

Figura 46 – Sifões mínimos sem *trap* referentes à Figura 42 que conduzem o estado do modelo do processo a um estado de *deadlock*.

estado de *deadlock* deve ser então definido. Para tanto, as transições são classificadas como diretamente, indiretamente ou não relacionada ao esvaziamento do sifão H_{15}. As funções de autorização associadas às transições classificadas como diretamente ou indiretamente relacionadas ao esvaziamento do sifão, ou seja, às transições pertencentes ao conjunto T^D, produzem como resultado as possíveis interpretações: *incerta* ou $falsa$. As outras funções de autorização associadas às transições não relacionadas ao esvaziamento do sifão produzem como resultado as possíveis interpretações: *verdadeira* e $falsa$. Considerando isto, cada uma das transições são analisadas a seguir:

- as transições t_{a7} e t_{p7} são diretamente responsáveis pelo esvaziamento do sifão H_{15}; logo elas são inseridas no conjunto T^D ($T^D = \{t_{a7}, t_{p7}\}$);

- a transição t_{a8} é habilitada por uma marcação imprecisa quando a transição t_{a7} é disparada de forma incerta; porém, seu disparo produz fichas apenas nos lugares que pertencem ao sifão H_{15}; logo a transição t_{a8} não é relacionada ao esvaziamento do sifão H_{15} e, consequentemente, ela não é inserida no conjunto T^D;

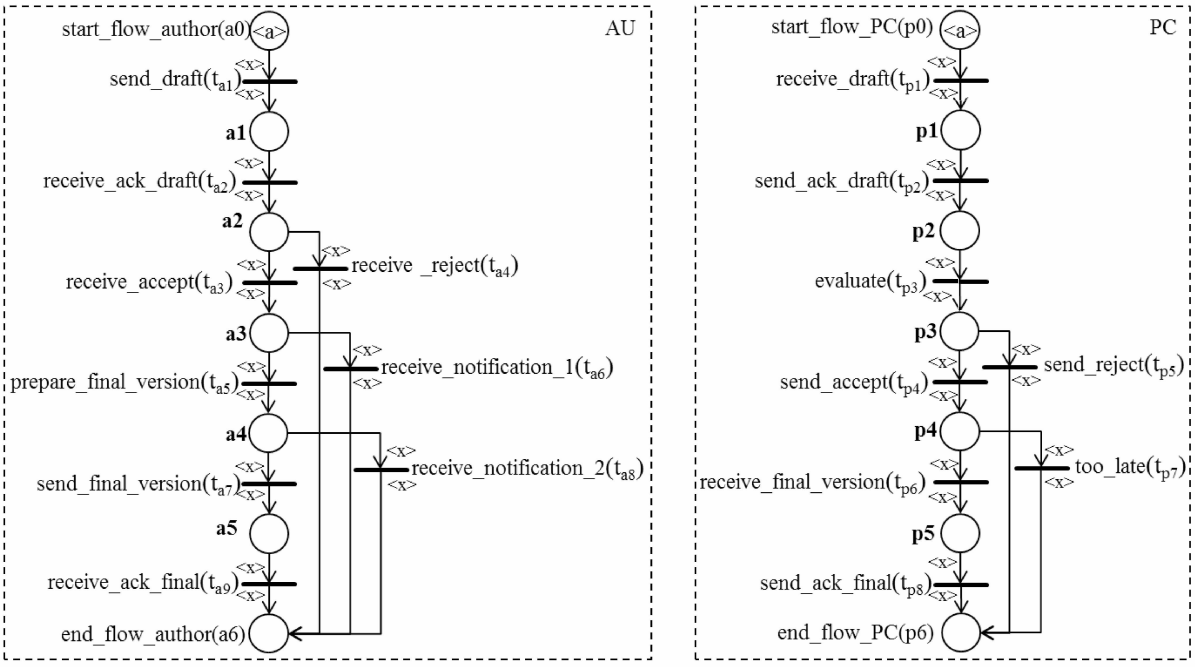
- a transição t_{a9} é habilitada por uma marcação imprecisa quando a transição t_{a7} é disparada de forma incerta; porém, seu disparo produz fichas apenas nos lugares que pertencem ao sifão H_{15}; logo a transição t_{a9} não é relacionada ao esvaziamento do sifão H_{15} e, consequentemente, ela não é inserida no conjunto T^D;

- a transição t_{p6} é habilitada por uma marcação imprecisa quando a transição t_{p7} é disparada de forma incerta; porém, seu disparo produz fichas apenas nos lugares que pertencem ao sifão H_{15}; logo a transição t_{p6} não é relacionada ao esvaziamento do sifão H_{15} e, consequentemente, ela não é inserida no conjunto T^D;

- as transições t_{a1}, t_{a2}, t_{a3}, t_{a4}, t_{a5}, t_{a6}, t_{p1}, t_{p2}, t_{p3}, t_{p4}, t_{p5} e t_{p8} são todas habilitadas por uma marcação precisa considerando a atual classificação; logo elas não são relacionadas ao esvaziamento do sifão H_{15} e, consequentemente, não são inseridas no conjunto T^D;

A transição t_{p7} foi inserida no conjunto T^D, entretanto, ao disparar t_{p7}, o lugar de saída $p6$ do modelo do processo PC é marcado indicando a correta conclusão do mesmo. Considerando que o disparo incerto é utilizado para garantir a existência de pelo menos uma sequência de disparo válida, a qual conduz o modelo do processo ao seu estado final e, que ao disparar a transição t_{p7} o estado final é alcançado, a transição t_{p7} é então removida do conjunto T^D. Desta forma, T^D é composto apenas pela transição t_{a7} ($T^D = \{t_{a7}\}$).

Após definir o conjunto T^D das transições que podem ser eventualmente disparadas de forma incerta a fim de evitar uma possível situação de *deadlock*, os modelos dos processos AU e PC são transformados em dois modelos em WF-net possibilística como ilustrado na Figura 47. Lembre-se que os lugares de comunicação são representados como simples eventos externos associados às transições nas funções de autorização ou nas ações A^c_{ta} e A^i_{ta} dos modelos como variáveis booleanas.



(a) AU (b) PC

Figura 47 – WF-nets possibilísticas referentes aos modelos dos processos AU e PC da IOWF-net representada na Figura 27.

$< a >$ é um objeto pertencente à classe "Artigo", bem como a variável x e todos os lugares relacionados aos modelos dos processos AU e PC. Tal classe é composta pelos seguintes atributos:

- $draft$, $ackDraft$, $accept$, $reject$, $tooLate$, $finalVersion$, $ackFinal$: booleano;

- $endDraft$, $endFVersion$, $sAFinal$: booleano;
- $evaluate$: constante caractere;
- $t_{max_{s5}}$: constante inteira.

O significado de cada um dos atributos declarados anteriormente para a classe "Artigo", bem como as ações necessárias para o entendimento do modelo são os mesmos definidos anteriormente na Seção 5.2. Em contrapartida, as funções de autorização, definidas de acordo com a análise estrutural previamente realizada, são mostradas na Tabela 6. As colunas com as subscrições "*verdadeira se*" e "*incerta se*" representam, respectivamente, as interpretações *verdadeira* e *incerta* associadas às transições através das funções de autorização. A interpretação $falsa$ não é especificada, porém a função de autorização é avaliada como $falsa$ sempre que as condições associadas às interpretações *verdadeira* e *incerta* não sejam válidas ao mesmo tempo. Por fim, se a interpretação *verdadeira* ou *incerta* não for especificada numa função de autorização, a mesma não pode ser avaliada como tal.

Tabela 6 – Funções de autorização associadas às transições pertencentes aos modelos AU e PC em WF-nets possibilísticas apresentadas na Figura 47.

	verdadeira se	***incerta se***
$\eta_x(t_{a1}) =$	$x.endDraft$	
$\eta_x(t_{a2}) =$	$x.ackDraft$	
$\eta_x(t_{a3}) =$	$x.accept$	
$\eta_x(t_{a4}) =$	$x.reject$	
$\eta_x(t_{a5}) =$	$x.accept \wedge \neg tooLate$	
$\eta_x(t_{a6}) =$	$x.tooLate$	
$\eta_x(t_{a7}) =$		$x.endFVersion$
$\eta_x(t_{a8}) =$	$x.tooLate$	
$\eta_x(t_{a9}) =$	$x.ackFinal$	
$\eta_x(t_{p1}) =$	$x.draft$	
$\eta_x(t_{p2}) =$	$x.draft$	
$\eta_x(t_{p3}) =$	$x.ackDraft$	
$\eta_x(t_{p4}) =$	$x.evaluate = 'a'$	
$\eta_x(t_{p5}) =$	$x.evaluate = 'r'$	
$\eta_x(t_{p6}) =$	$x.finalVersion$	
$\eta_x(t_{p7}) =$	$\tau \geq t_{max_{s5}}$	
$\eta_x(t_{p8}) =$	$x.sAFinal$	

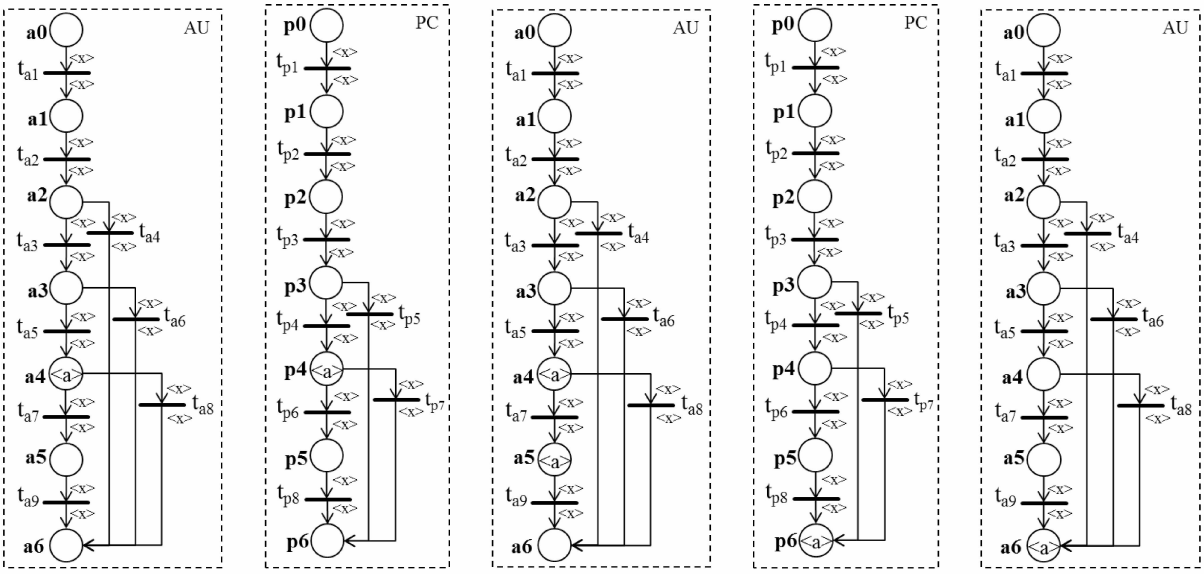
Finalmente, a fim de ilustrar, através do uso dos disparos incertos, uma sequência de disparos responsável pela situação de *deadlock* durante a execução do modelo do processo, a configuração a seguir é considerada:

- o elemento τ representa o tempo corrente;
- o atributo $t_{max_{s5}}$ é inicializado com o valor 90;
- as transições t_{a1}, t_{a2}, t_{a3} e t_{a5} do modelo do processo AU e as transições t_{p1}, t_{p2}, t_{p3} e t_{p4} do modelo do processo PC já foram disparadas (Figuras 48a e 48b);
- as transições t_{a7} e t_{p7} são disparadas ao mesmo tempo com $\tau = 90$.

Levando em conta a configuração acima especificada, os modelos dos processos AU e PC apresentados na Figura 47 são executados de acordo com o "jogador" apresentado na Figura 29, ou seja, disparos incertos e marcações imprecisas são consideradas. Os passos de simulação são detalhados abaixo:

- no tempo corrente $\tau = 90$, considerando que o atributo *tooLate* ainda é $falso$, o autor envia a versão final do artigo para PC, ou seja, como a transição t_{a7} está habilitada por uma marcação precisa e a função de autorização associada à transição t_{a7} pertencente ao modelo do processo AU é avaliada como *incerta* ($\eta_{<a>}(t_{a7}) =$ *incerta*) visto que o atributo $endFVersion = verdadeiro$, t_{a7} é disparada de forma incerta (Figura 48c). A ação A^i_{ta7}, a qual é relacionada ao lugar de comunicação $c6$, é executada, alterando o valor do atributo $finalVersion$ para *verdadeiro*;
- ainda no tempo corrente $\tau = 90$, o tempo limite permitido para enviar a versão final do artigo foi atingido, isto é, a função de autorização associada à transição t_{p7} pertencente ao modelo do processo PC é avaliada como *verdadeira* ($\eta_{<a>}(t_{p7}) =$ *verdadeira*) uma vez que $\tau \geq t_{max_{s5}}$. Dado que t_{p7} está habilitada por uma marcação precisa, ela é disparada com certeza (Figura 48d) e a ação A^c_{tp7} é executada, alterando o valor do atributo *tooLate* para *verdadeiro*. Observe-se que o estado final do modelo do processo PC é alcançado sem que uma situação de *deadlock* ocorra;
- no tempo corrente $\tau = 91$, a transição t_{a8} está habilitada por uma marcação imprecisa e a função de autorização associada a ela se torna *verdadeira* ($\eta_{<a>}(t_{a8}) =$ *verdadeira*) pois o atributo $tooLate = verdadeiro$. Isso significa que através de uma sequência de disparo incerto o modelo do processo AU alcançou uma marcação que sensibiliza uma transição, a qual não é responsável pelo esvaziamento do sifão H_{15} e, consequentemente, pela situação de *deadlock* presente no modelo da IOWF-net mostrada na Figura 27. Assim sendo, o algoritmo de recuperação descrito no Algoritmo 1 é chamado para cancelar o disparo incerto de t_{a7}. Após a execução do

algoritmo de recuperação, a transição t_{a8} é disparada com certeza dado que está habilitada por uma marcação precisa e a função de autorização associada a ela é avaliada como *verdadeira* (Figura 48e). E, caso fosse definida, a ação A^i_{ta8} seria executada.



(a) *AU* (b) *PC* (c) $\tau = 90$ (d) $\tau = 90$ (e) $\tau = 91$

Figura 48 – Resultados da simulação - Prevenção de *deadlocks*.

Ao fim da execução dos modelos dos processos *AU* e *PC* em WF-net possibilística pode-se notar que, mesmo tendo disparado as transições t_{a7} e t_{p7}, a marcação final obtida não alcançou uma situação de *deadlock*. Caso a noção dos disparos incertos não fosse utilizada, a situação de *deadlock* destacada na Figura 49 ocorreria uma vez que o envio da versão final do artigo é realizado no mesmo momento em que o tempo limite permitido para o seu envio é alcançado. Os disparos incertos permitem finalizar o modelo do processo *AU* de forma a evitar que se fique numa espera infindável pelo reconhecimento do recebimento da versão final do artigo por parte do *PC*.

A noção de disparos incertos é considerada para permitir um comportamento flexível quando situações de desvios relacionados à comportamentos extremos são ocasionados, tais como uma situação de *deadlock* alcançada através de uma sequência de disparo imprópria. Desta forma, as sequências de disparos errôneos, ou seja, as sequências que levam a uma situação de *deadlock* neste caso, são evitadas através da IOWF-net possibilística pelos disparos incertos pois qualquer situação de *deadlock* deve ser evitada por qualquer um dos modelos de processo.

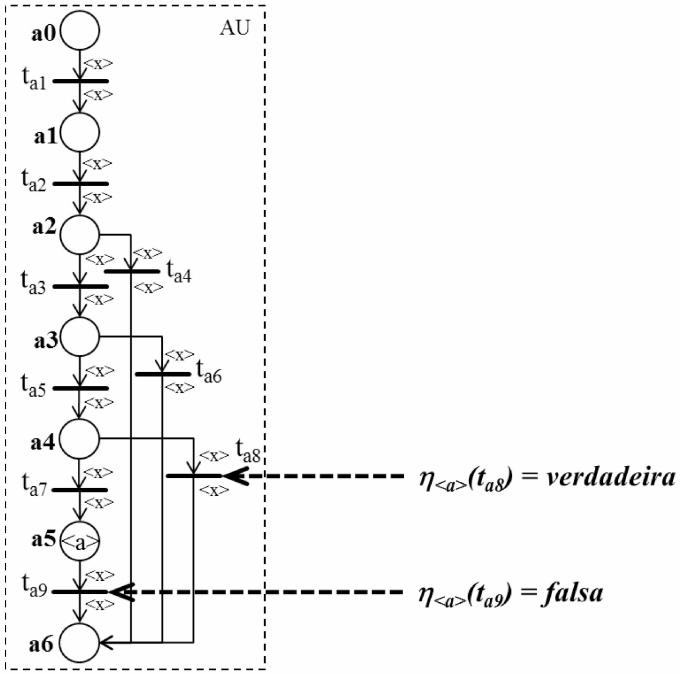


Figura 49 – Situação de *deadlock* no modelo do processo *AU* quando a noção dos disparos incertos é desconsiderada.

5.4 Considerações Finais

Neste capítulo a IOWF-net possibilística foi definida e aplicada em problemas relacionados com desvios em processos de negócio. A IOWF-net possibilística é baseada na combinação da estrutura de roteamento e dos protocolos de comunicação de uma IOWF-net com a noção de disparo incerto e marcação imprecisa de uma rede de Petri possibilística. Tal abordagem produz um tipo de modelo capaz de lidar com possíveis falhas de comunicação e existências de problemas estruturais no modelo dos processos durante a execução dos mesmos.

Na Seção 5.1, a IOWF-net possibilística é aplicada aos processos de negócio interorganizacionais com o objetivo de lidar com problemas de comunicação, tais como mensagens perdidas, atrasadas ou inesperadas entre os processos de forma a evitar situações de inconsistências no modelo dos processos. Através da noção de disparo incerto, situações imprevistas ocasionadas por desvios de comportamento durante a execução do modelo do processo interorganizacional podem ser devidamente tratados de forma o mais automática possível.

Na Seção 5.3, as situações de *deadlock* causadas, em sua grande maioria, por razões estruturais nos modelos dos processos de negócio que se comunicam/colaboram são evitadas através do modelo em IOWF-net possibilística sem que a sua estrutura seja modificada. A noção de disparo incerto permite evitar as sequências de disparos errôneos, ou seja, as sequências que levam a uma situação de *deadlock*.

Capítulo 6

Implementação da WorkFlow net Possibilística

O CPN Tools é uma ferramenta para a edição e análise de modelos *CPN*. Utilizando-o é possível investigar o comportamento de um sistema modelado em CPN para verificar propriedades e avaliar simulações de tipo Monte Carlo para conduzir a análise de desempenho. Ele é, então, um software que oferece funcionalidades de edição, simulação, análise de espaço dos estados e de desempenho de um modelo CPN (JENSEN; KRISTENSEN, 2009).

Com o objetivo de mostrar a modelagem e a implementação de diferentes modelos de processo em WF-net possibilística, a ferramenta computacional CPN Tools é utilizada. Esta ferramenta permite desenvolver um modelo em CPN com códigos descritos na linguagem CPN ML de forma que os disparos incertos e, consequentemente, o algoritmo de defuzzificação possam ser configurados. Além disto, por meio dos lugares de comunicação entre modelos definidos em subpáginas diferentes na área de trabalho do software, a ferramenta CPN Tools viabiliza modelar processos interorganizacionais em IOWF-net possibilística.

A edição de um modelo baseado em WF-net possibilística usando a ferramenta CPN Tools necessita de um lugar de controle responsável por gerenciar a execução do modelo. Isto ocorre por causa da natureza incerta da WF-net possibilística, a qual não é inerente em um modelo baseado em CPN. Tal lugar deve coordenar os tipos de disparos (certo ou incerto) e a recuperação do estado do sistema (iniciar e concluir o procedimento de recuperação, e finalizar ou cancelar um disparo incerto).

Além do lugar de controle, uma transição responsável por gerenciar o tempo corrente deve ser adicionada ao modelo com prioridade de disparo alterada para P_LOW[1]. Mesmo que na ferramenta CPN Tools existam funções de tempo preestabelecidas, esta transição permite a realização de um disparo, mesmo que o tempo corrente ainda não tenha alcan-

[1] A fim de tornar a visualização do modelo mais simples, as prioridades de disparo foram simplificadas da seguinte forma: P_HIGH para PH e P_LOW para PL.

çado o tempo que o evento associado ao objeto em questão se torna verdadeiro. Desta forma, os disparos incertos podem ser realizados, o que não é possível considerando as opções de gerenciamento de tempo disponíveis no CPN Tools.

Para considerar o comportamento de um disparo incerto, isto é, a não remoção dos objetos (fichas) dos lugares de entrada da transição pseudo-disparada, arcos adicionais são inseridos no modelo baseado em CPN. Tais arcos têm uma condição associada a eles para permitir o "retorno" dos objetos quando um disparo incerto é realizado. Assim sendo, todas as transições têm um arco orientado para cada lugar de entrada associado a elas.

Por fim, para simplificar a edição de uma WF-net possibilística no CPN Tools, o cancelamento do disparo incerto de uma transição t é realizado pelas transições de entrada dos lugares de saída de t. Entretanto, se o disparo incerto realizado em t não pode ser cancelado através destas transições de entrada, transições adicionais devem ser inseridas no modelo para realizar, exclusivamente, o cancelamento do disparo incerto de t.

Observa-se que, se definidas, as ações inerentes do modelo, ou seja, as ações A^c_{ta}, A^i_{ta}, A^f_{ta} e A^l_{ta}, são associadas através de segmentos de código associados às transições relacionadas. As funções e os tipos de dados comuns aos modelos de processo em WF-net possibilística ou IOWF-net possibilística quando a ferramenta CPN Tools é utilizada são especificados no Apêndice B. Nas Seções 6.1 e 6.2, os modelos dos processos usados para ilustrar as propostas descritas nos Capítulos 4 e 5 são editados na ferramenta CPN Tools. Na Seção 6.1, três modelos, representados por uma WF-net possibilística, são apresentados e, na Seção 6.2, mais dois modelos, representados por uma IOWF-net possibilística, são apresentados. Por fim, na Seção 6.3, os modelos são executados e os resultados obtidos são avaliados e comparados com os resultados quando não se utiliza o conceito de disparo incerto.

6.1 Edição de WF-net possibilística na Ferramenta CPN Tools

Nesta Seção, os dois modelos de processos em WF-net possibilística usados para ilustrar a proposta descrita no Capítulo 4 são editados na ferramenta CPN Tools. Além destes dois modelos, o modelo do processo de tratamento de reclamações em WF-net possibilística apresentado na Seção 2.2 também é editado na ferramenta CPN Tools.

A implementação destes modelos na ferramenta CPN Tools faz uso das funções apresentadas no Apêndice B e de uma função adicional que se refere às funções de autorização associadas às transições pertencentes aos modelos. As funções de autorização relacionadas aos três modelos implementados no CPN Tools são descritas, respectivamente, nas Seções C.1, C.2 e C.3.

Além disto, cinco variáveis são necessárias para realizar a edição dos modelos no CPN Tools. Variáveis CPN têm a mesma característica de variáveis em qualquer linguagem de

programação, porém os tipos de dados são as Cores (*colset*). A declaração destas variáveis é dada a seguir:

- as variáveis x e y são usadas nas expressões de arco associadas aos arcos de entrada e de saída dos lugares do tipo $CASO$ [2];

```
var x, y : CASO;
```

- as variáveis c e r são usadas nas expressões de arco associadas, respectivamente, aos arcos de saída e de entrada do lugar de controle (nomeado C nos modelos) do tipo $CTRL$ [3]. A variável r corresponde à variável c modificada após o disparo de uma transição;

```
var c, r: CTRL;
```

- a variável dr é usada no segmento de código associado às transições e na função E associada às expressões de arco de saída das transições. Esta variável se refere à duração necessária para que o evento relacionado à conclusão da atividade se torne verdadeiro;

```
var dr: INT;
```

Nas subseções 6.1.1, 6.1.2 e 6.1.3, o processo de tratamento de reclamações apresentado na Seção 2.2, o processo de encomenda exposto na Seção 4.2 e a versão simplificada de um processo de aplicação de cartão de crédito apresentado na Seção 4.3, todos modelados em WF-net possibilística, são respectivamente editados na ferramenta CPN Tools.

6.1.1 Edição de WF-net Possibilística que aceita Desvios

A edição no CPN Tools do modelo em WF-net possibilística do processo de tratamento de reclamações definido por (AALST; HEE, 2004) é apresentada na Figura 50. Com o intuito de facilitar o entendimento das funções descritas no Apêndice B, identificadores numéricos são associados aos nomes atribuídos aos lugares e às transições pertencentes ao modelo. Nos lugares os identificadores são mostrados antes do nome atribuído ao lugar e nas transições os identificadores são mostrados após o ponto.

Neste modelo, quando um objeto do tipo $CASO$ se encontra nos lugares de espera ($W1, W2, W3, W4, W5, W6$ e $W7$), a atividade seguinte pode ser iniciada imediatamente. Desta forma, quando o disparo de uma transição t produz um objeto em um destes lugares

[2] O tipo $CASO$ representa a classe do objeto C_{aso} utilizada na definição da WF-net Possibilística. Nesta classe, o conjunto de atributos é formado pelo identificador do Caso e pelo tempo de chegada de um evento associado ao Caso. Para maiores detalhes, vide Apêndice B.

[3] O tipo $CTRL$ é responsável por gerenciar os dados necessários para permitir a redefinição do "jogador" do CPN Tools pelo "jogador" de WF-net possibilística. Para maiores detalhes, vide Apêndice B.

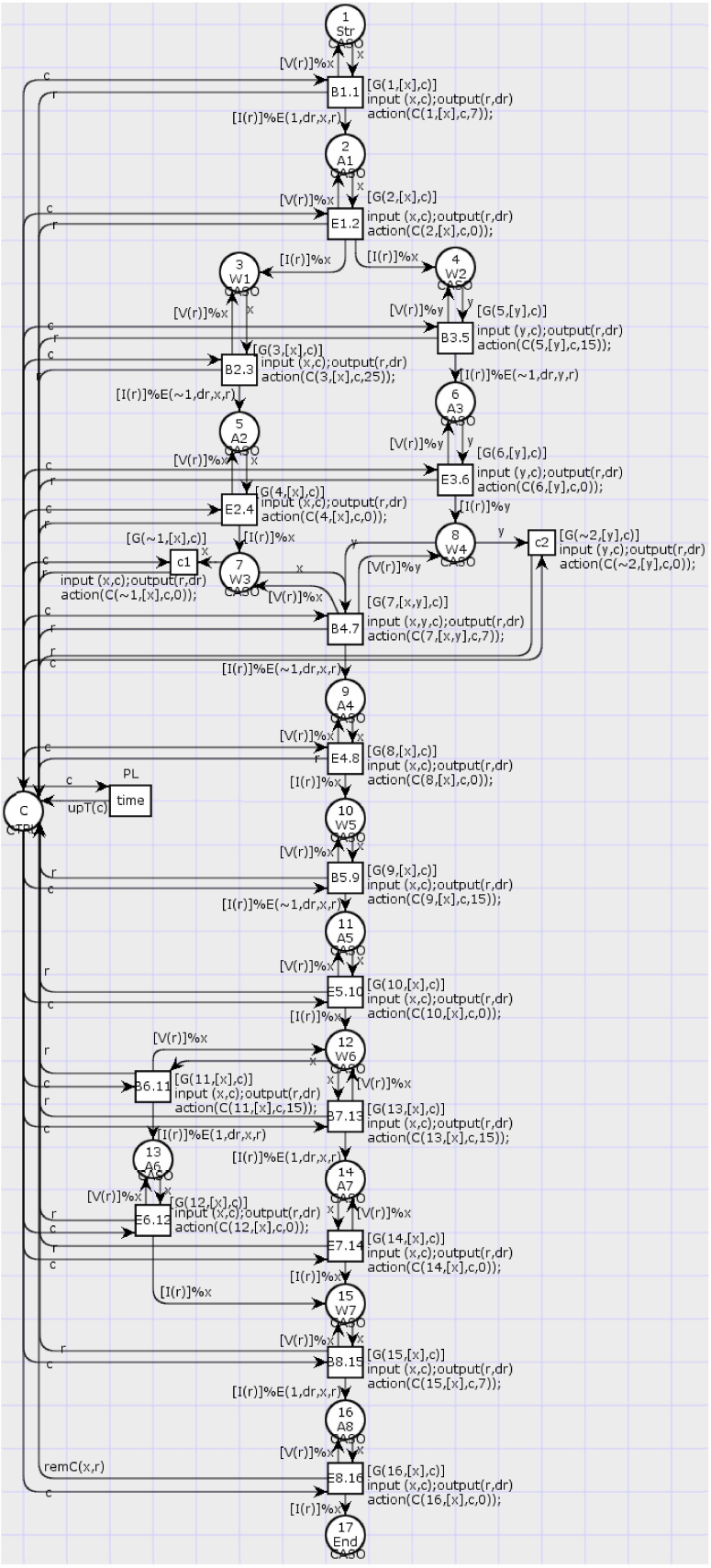


Figura 50 – WF-net possibilística de um processo de tratamento de reclamações editada no CPN Tools.

de espera, o tempo de chegada do evento responsável por iniciar a atividade subsequente é associado a este objeto com valor idêntico ao tempo máximo associado aos objetos pertencentes aos lugares de entrada de t, ou seja, a função E [4] não é usada nas expressões dos arcos de saída de t. Assim sendo, a transição de saída destes lugares de espera pode ser disparada imediatamente por um disparo certo ou incerto dependendo do disparo anteriormente realizado e de sua função de autorização.

Ao disparar uma das transições representadas pelos identificadores 1, 3, 5, 7, 9, 11, 13 e 15, a duração necessária para que o evento relacionado à conclusão da execução da atividade associada à transição disparada se torne verdadeiro é acrescentada aos objetos produzidos nos lugares de saída da transição através da execução da função E. Tal duração é obtida de forma exponencial através da execução da função tmE [5] presente na função C [6] considerando as durações médias associadas às transições. Segundo Doane e Seward (2011), uma distribuição exponencial de probabilidade é muito útil para descrever o tempo que se leva para completar uma tarefa; por exemplo, descrever o tempo entre a chegada de um motoboy à casa de um cliente ou o tempo exigido para alguma tarefa dentro de uma fábrica, podendo assim ser aplicado em qualquer área aonde exista a necessidade de identificar tempos percorridos aleatórios.

O tempo médio de execução de um Caso, considerando os roteamentos paralelo e condicional presentes no modelo do processo e as durações médias associadas às transições descritas a seguir, é de 76 unidades de tempo. O valor de tais durações médias são:

- transição $B1.1$ - duração média = 7 unidades de tempo;
- transição $B2.3$ - duração média = 25 unidades de tempo;
- transição $B3.5$ - duração média = 15 unidades de tempo;
- transição $B4.7$ - duração média = 7 unidades de tempo;
- transição $B5.9$ - duração média = 15 unidades de tempo;
- transição $B6.11$ - duração média = 15 unidades de tempo;
- transição $B7.13$ - duração média = 15 unidades de tempo;
- transição $B8.15$ - duração média = 7 unidades de tempo.

A fim de indicar que um determinado evento não ocorre durante a execução do Caso correspondente, o tempo de chegada associado ao objeto do tipo $CASO$ relativo a este

[4] A função E é responsável por definir o tempo de chegada do evento associado ao Caso identificado por idC. Para maiores detalhes, vide Seção B.1.

[5] Para maiores detalhes, vide Seção B.1.

[6] A função C é responsável por atualizar o objeto que se localiza no lugar de controle e por definir a duração necessária para a chegada de um determinado evento. Para maiores detalhes, vide Seção B.6.

evento é indicado com o valor negativo -1. Este valor é obtido de modo aleatório a partir da execução da função E associada aos arcos de entrada. No modelo, os eventos que podem ser perdidos são os relacionados à conclusão da execução das atividades $A2$, $A3$, $A4$ e $A5$. Isto pode ser observado nos arcos de entrada dos lugares relacionados a estas atividades através da chamada da função E com a presença do valor negativo -1 no primeiro parâmetro. Este parâmetro tem com o objetivo indicar à função E que a chegada do evento em questão pode ser perdida.

As transições $c1$ e $c2$ presentes no modelo do processo são responsáveis, exclusivamente, por cancelar, respectivamente, o disparo incerto das transições $E2$ e $E3$. Isto ocorre porque a transição $B4$ (única transição de entrada do lugar de saída tanto da transição $E2$ quanto da transição $E3$) não é capaz de cancelar o disparo incerto realizado na transição $E2$ ou na transição $E3$. Esta incapacidade se dá devido a necessidade de ambos os lugares de entrada da transição $B4$ ($W3$ e $W4$) estarem marcados para que ela possa ser disparada, o que nem sempre ocorre quando é necessário cancelar o disparo incerto realizado em apenas uma das transições.

Por fim, as funções de autorização associadas às transições são agrupadas na função $funAut$, a qual é descrita na Seção C.1. Esta função é responsável por definir, no momento da execução do modelo do processo, se a função de autorização associada à transição em questão é avaliada como $verdadeira$, $incerta$ ou $falsa$.

6.1.2 Edição de WF-net possibilística com Estrutura Sequencial que aceita Desvios para o Paralelismo inerente ao Processo

A edição no CPN Tools do modelo em WF-net possibilística referente ao processo de encomenda definido por (WESKE, 2007) é apresentada na Figura 51. Com o intuito de facilitar o entendimento das funções descritas no Apêndice B, identificadores numéricos são associados aos nomes atribuídos aos lugares e às transições pertencentes ao modelo. Nos lugares os identificadores são mostrados antes do nome atribuído ao lugar e nas transições os identificadores são mostrados após o ponto.

Neste modelo, quando um objeto do tipo $CASO$ se encontra nos lugares de espera ($W1$, $W2$, $W3$, $W4$, $W5$ e $W6$), a atividade seguinte pode ser iniciada imediatamente. Desta forma, quando o disparo de uma transição t produz um objeto em um destes lugares de espera, o tempo de chegada do evento responsável por iniciar a atividade subsequente é associado a este objeto com valor idêntico ao tempo associado ao objeto pertencente ao lugar de entrada de t, ou seja, a função E não é usada na expressão do arco de saída de t. Assim sendo, a transição de saída destes lugares de espera pode ser disparada imediatamente por um disparo certo ou incerto dependendo do disparo anteriormente realizado e de sua função de autorização.

Ao disparar uma das transições representadas pelos identificadores 1, 3, 5, 7, 9, 11

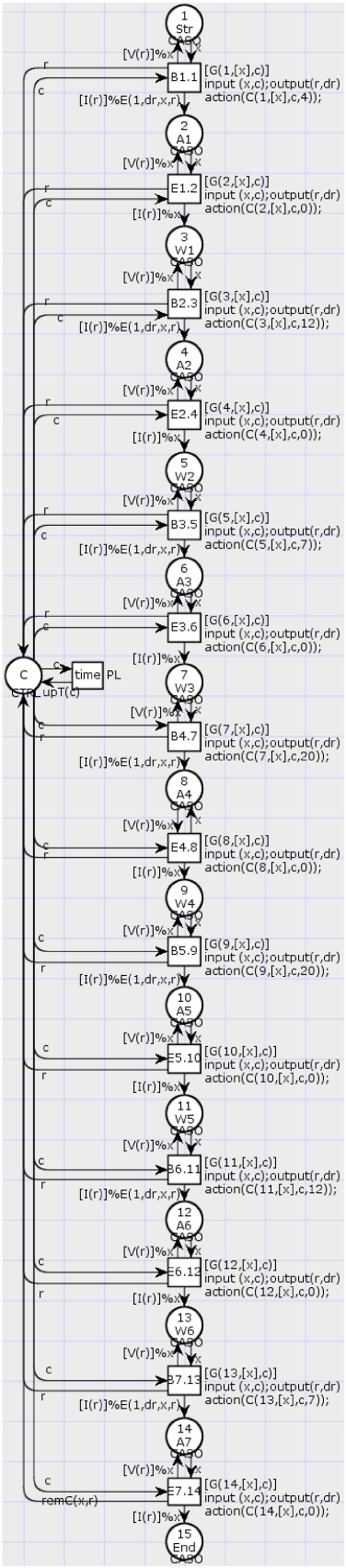


Figura 51 – WF-net possibilística (Figura 30) de um processo de encomenda editado no CPN Tools.

e 13, a duração necessária para que o evento relacionado à conclusão da execução da atividade associada à transição disparada se torne verdadeiro é acrescentada aos objetos produzidos nos lugares de saída da transição através da execução da função E. Tal duração é obtida de forma exponencial através da execução da função tmE presente na função C considerando as durações médias associadas às transições. O tempo médio de execução de um Caso, considerando as durações médias associadas às transições descritas a seguir, é de 82 unidades de tempo. As durações médias associadas às transições são:

- transição $B1.1$ - duração média = 4 unidades de tempo;
- transição $B2.3$ - duração média = 12 unidades de tempo;
- transição $B3.5$ - duração média = 7 unidades de tempo;
- transição $B4.7$ - duração média = 20 unidades de tempo;
- transição $B5.9$ - duração média = 20 unidades de tempo;
- transição $B6.11$ - duração média = 12 unidades de tempo;
- transição $B7.13$ - duração média = 7 unidades de tempo.

O valor do tempo associado aos objetos do tipo $CASO$, que representa a ocorrência de um evento, nunca será negativo, ou seja, neste modelo a chegada da confirmação dos eventos pode estar atrasada, mas nunca perdida. Desta forma, em nenhuma chamada da função E o primeiro parâmetro contem o valor negativo -1, dado que não existe a possibilidade de eventos perdidos.

Por fim, as funções de autorização associadas às transições pertencentes ao modelo apresentado na Figura 30 são agrupadas na função $funAut$, a qual é descrita na Seção C.2. A função $funAut$ é responsável por definir, no momento da execução do modelo do processo, se a função de autorização associada à transição em questão é avaliada como $verdadeira$, $incerta$ ou $falsa$. Além de $funAut$, as funções $aLPF$, $cLPF$ e $GFiring$ [7] descritas no Apêndice B são modificadas para dar suporte a troca do algoritmo de defuzzificação apresentado no Algoritmo 1 pelo apresentado no Algoritmo 3.

6.1.3 Edição de WF-net Possibilística que aceita Desvios para o Cancelamento de Processos

A edição no CPN Tools do modelo em WF-net possibilística de uma versão simplificada de um processo de solicitação de cartão de crédito definido por (WYNN et al., 2009) é apresentada na Figura 52.

[7] As funções $aLPF$ e $cLPF$ são responsáveis por definir a lista de transições a terem o disparo incerto cancelado e a função $GFiring$ por permitir ou não o disparo certo ou incerto de uma determinada transição.

Ao disparar uma transição t, a duração necessária para que o evento relacionado à conclusão da execução da atividade associada a t se torne verdadeiro é acrescentada aos objetos produzidos nos lugares de saída de t através da execução da função E. Tal duração

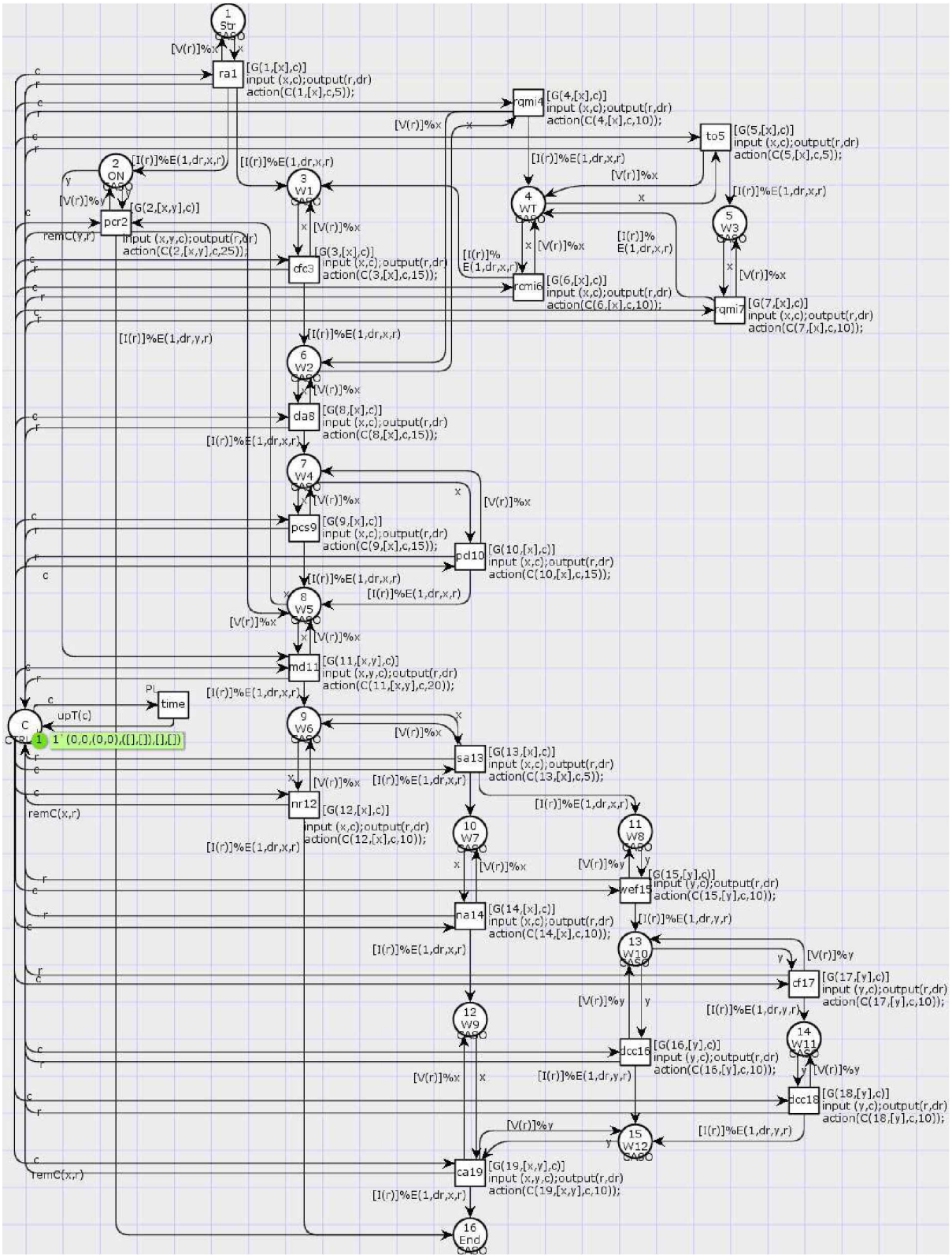


Figura 52 – WF-net possibilística de uma versão simplificada de um processo de solicitação de cartão de crédito (Figura 34) editado no CPN Tools.

é obtida de forma exponencial através da execução da função tmE presente na função C considerando as durações médias associadas às transições. O tempo médio de execução de um Caso, considerando as durações médias associadas às transições descritas a seguir, é de 175 unidades de tempo. As durações médias associadas às transições são:

- transição $ra1$ - duração média = 5 unidades de tempo;
- transição $pcr2$ - duração média = 25 unidades de tempo;
- transição $cfc3$ - duração média = 15 unidades de tempo;
- transição $rqmi4$ - duração média = 10 unidades de tempo;
- transição $to5$ - duração média = 5 unidades de tempo;
- transição $rcmi6$ - duração média = 10 unidades de tempo;
- transição $rqmi7$ - duração média = 10 unidades de tempo;
- transição $cla8$ - duração média = 15 unidades de tempo;
- transição $pcs9$ - duração média = 15 unidades de tempo;
- transição $pcl10$ - duração média = 15 unidades de tempo;
- transição $md11$ - duração média = 20 unidades de tempo;
- transição $nr12$ - duração média = 10 unidades de tempo;
- transição $sa13$ - duração média = 5 unidades de tempo;
- transição $na14$ - duração média = 10 unidades de tempo;
- transição $wef15$ - duração média = 10 unidades de tempo;
- transição $dcc16$ - duração média = 10 unidades de tempo;
- transição $cf17$ - duração média = 10 unidades de tempo;
- transição $dcc18$ - duração média = 10 unidades de tempo;
- transição $ca19$ - duração média = 10 unidades de tempo.

Observa-se que no modelo os identificadores dos lugares e das transições utilizados nas funções anteriormente descritas são mostrados nos nomes dos lugares e das transições para facilitar o entendimento do mesmo. Nos lugares os identificadores são mostrados antes do nome atribuído ao lugar e nas transições os identificadores são mostrados após o nome.

Por fim, as funções de autorização descritas na Tabela 2 referente ao modelo apresentado na Figura 34 são agrupadas na função $funAut$, a qual é descrita na Seção C.3. A função $funAut$ é responsável por definir, no momento da execução do modelo do processo, se a função de autorização associada à transição em questão é avaliada como $verdadeira$, $incerta$ ou $falsa$. Lembre-se que neste modelo apenas as transições com identificadores 3, 4, 5, 6, 7, 8, 9 e 10 podem ser disparadas de forma incerta. Estas transições se referem à região de cancelamento, isto é, quando um pedido de cancelamento é solicitado, se o Caso referente a este pedido se localizar em algum dos lugares de entrada destas transições, uma sequência de disparos incertos será realizada enquanto a transição com identificador 2 não estiver habilitada. No momento em que ela estiver habilitada, o procedimento de defuzzificação será iniciado, permitindo assim cancelar a execução do Caso em questão. Nas outras transições nenhum disparo incerto pode ser realizado.

6.2 Edição de IOWF-net possibilística na Ferramenta CPN Tools

Nesta Seção, os dois modelos de processos representados por uma IOWF-net possibilística usados para ilustrar a proposta descrita no Capítulo 5 são editados na ferramenta CPN Tools. Para isto, faz-se necessário o uso das funções apresentadas no Apêndice B juntamente com uma função adicional que se refere às funções de autorização associadas às transições pertencentes aos modelos em questão. As funções de autorização relacionadas a estes dois modelos implementados no CPN Tools são descritas, respectivamente, nas Seções C.4 e C.5.

De acordo com a definição de uma IOWF-net possibilística, os lugares de comunicação são considerados como eventos enviados ou recebidos por outro modelo. Assim sendo, quando uma transição t recebe ou envia a confirmação da ocorrência de um evento via um lugar de comunicação p, arcos de entrada e de saída entre t e p são inseridos no modelo editado no CPN Tools a fim de considerar o comportamento de um disparo incerto.

Considerando que os objetos presentes nos lugares de comunicação são compostos por um identificador (idC) e pelo momento em que o evento se torna verdadeiro (aEv), se ao disparar uma transição t uma comunicação é enviada ao processo parceiro, então o valor de aEv é modificado através da função IC [8]. Em contrapartida, se t apenas recebe a comunicação, então nenhuma alteração de valor é realizado em aEv.

Devido à característica particular dos lugares de comunicação, a função G [9] deve diferenciar quando os lugares de comunicação associados a uma determinada transição t

[8] A função IC é responsável por definir o tempo de chegada da comunicação associada ao objeto do tipo $CASO$ localizado em um dos lugares de comunicação. Para maiores detalhes, vide Seção B.1.

[9] A função G é responsável por permitir ou não o disparo de uma transição. Para maiores detalhes, vide Seção B.6.

correspondem à chegada ou ao envio de um comunicado. Para isto, ao chamar a função G na função de guarda associada a cada transição pertencente ao modelo, os lugares de comunicação que caracterizam o envio de um comunicado são separados dos outros lugares de entrada de t, inclusive dos lugares de comunicação que caracterizam o recebimento de um comunicado.

Além das funções usadas nos modelos, oito variáveis são necessárias para realizar a edição dos mesmos no CPN Tools. A declaração destas variáveis é dada a seguir:

- as variáveis x, y e k são usadas nas expressões de arco associadas aos arcos de entrada e de saída dos lugares do tipo $CASO$;

```
var x, y, k : CASO;
```

- as variáveis c e q são usadas nas expressões de arco de saída do lugar de controle (nomeado C nos modelos) do tipo $CTRL$ e as variáveis r e s nas expressões de arco de entrada de C. A variável r corresponde à variável c modificada após o disparo de uma transição e a variável s à variável q modificada;

```
var c, q, r, s: CTRL;
```

- a variável dr é usada no segmento de código associado às transições e na função E associada às expressões de arco de saída das transições. Esta variável se refere à duração necessária para que o evento relacionado à conclusão da execução da atividade se torne verdadeiro;

```
var dr: INT;
```

Nas subseções 6.2.1 e 6.2.2, o processo que precede a apresentação de um artigo em uma conferência modelado em IOWF-net possibilística é editado na ferramenta CPN Tools. O modelo editado na subseção 6.2.1 respeita a propriedade *Soundness* e o outro, na subseção 6.2.2, a propriedade *Relaxed Soundness*.

6.2.1 Edição de IOWF-net Possibilística que aceita Desvios em casos de Falhas de Comunicação entre Processos

A edição no CPN Tools dos processos AU e PC, que compõem a IOWF-net possibilística referente ao processo que precede a apresentação de um artigo em uma conferência definido por (AALST, 1998b), são apresentados, respectivamente, nas Figuras 53 e 54. A transição responsável por gerenciar o tempo corrente entre os modelos dos processos é definido em outra subpágina na área de trabalho do CPN Tools, a qual é mostrada na Figura 55.

Os lugares de comunicação entre os modelos dos processos são representados através dos lugares de fusão presentes tanto no modelo do processo AU quanto no modelo do processo PC. São eles: $draft$, $ackDraft$, $reject$, $accept$, $tooLate$, $finalVersion$ e $ackFinal$. Estes lugares são considerados como variáveis booleanas associadas à função de autorização ou à ação de uma dada transição. Observa-se que os lugares de controle $C.AU$ e

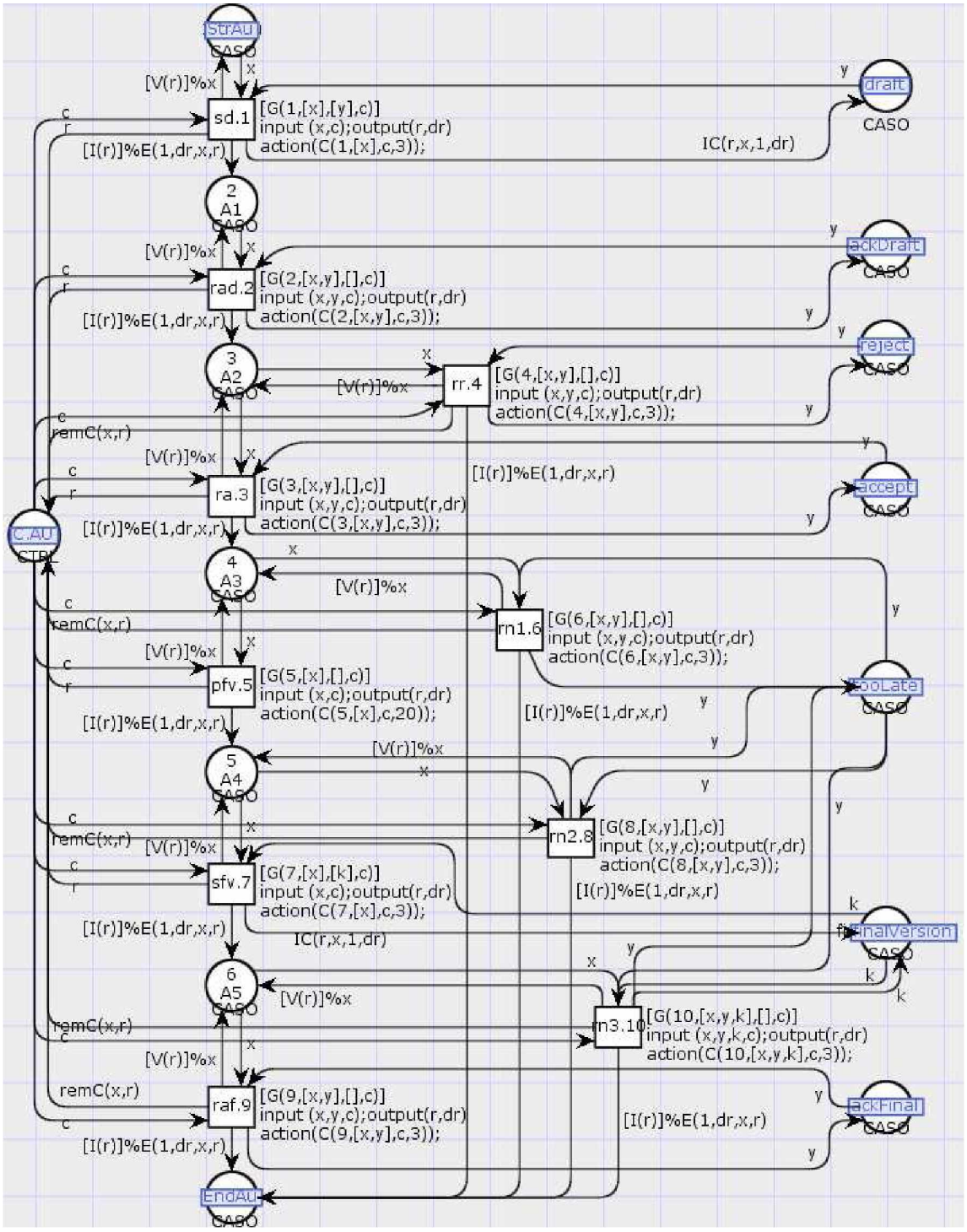


Figura 53 – IOWF-net possibilística *sound* do processo AU editada no CPN Tools referente ao processo que precede a apresentação de um artigo em uma conferência.

$C.PC$ são considerados, também, como lugares de fusão. Isto ocorre devido à presença da transição *time* necessária para gerenciar o tempo corrente entre ambos os modelos.

A fim de sinalizar a falha de comunicação entre os processos parceiros AU e PC, o tempo associado ao objeto do tipo $CASO$, produzido em um dos lugares de comunicação

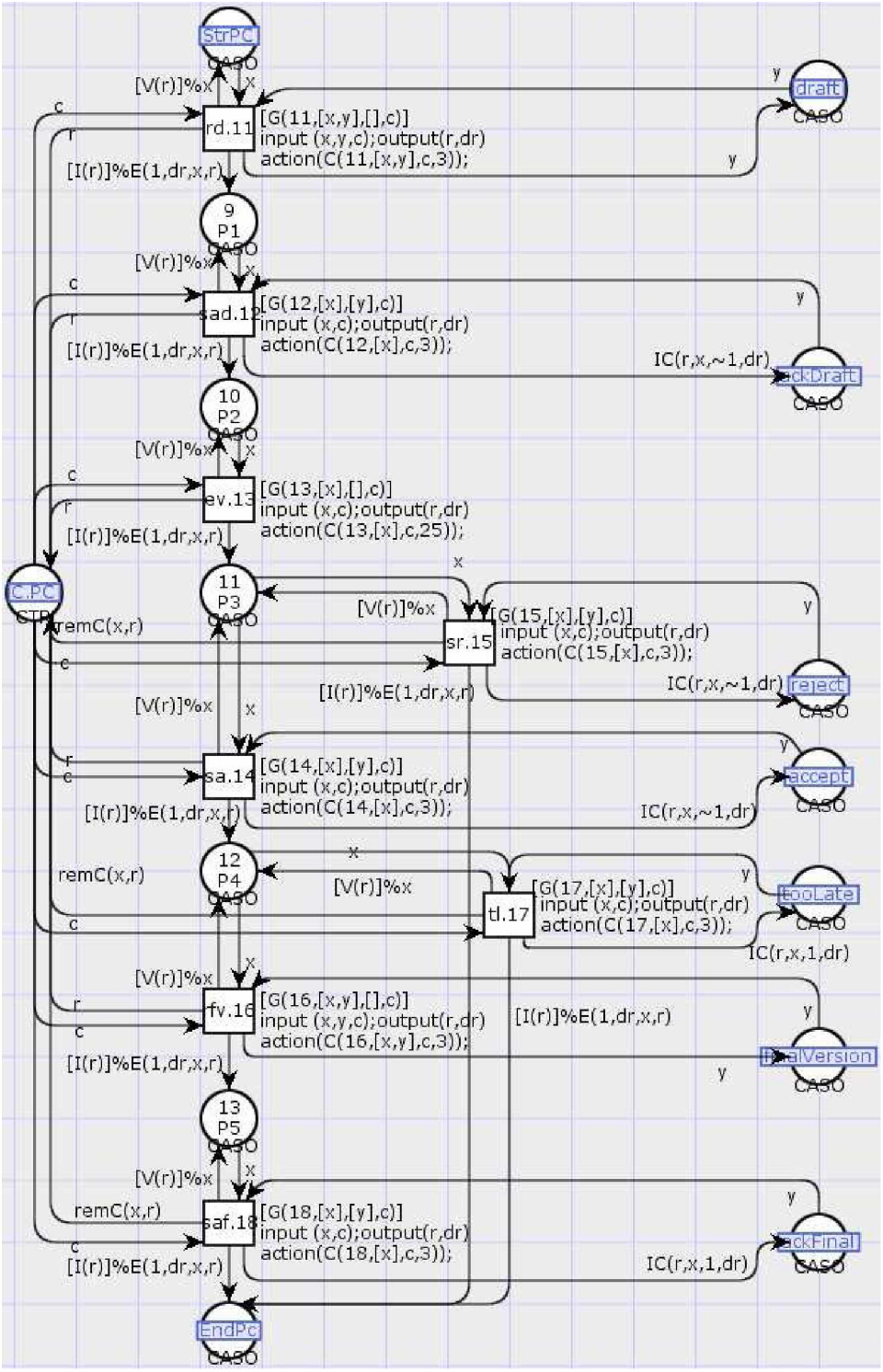


Figura 54 – IOWF-net possibilística *sound* do processo PC editada no CPN Tools referente ao processo que precede a apresentação de um artigo em uma conferência.

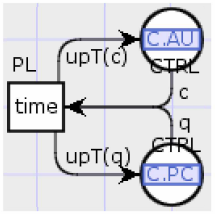


Figura 55 – Transição responsável por gerenciar o tempo corrente entre os modelos dos processos AU (Figura 53) e PC (Figura 54).

após o disparo de uma transição responsável por enviar uma informação ao processo parceiro, é indicado com o valor negativo -1. Este tempo indica o momento em que o evento correspondente se torna verdadeiro, ou seja, se negativo, o evento sempre é falso. As transições $sad.12$, $sa.14$ e $sr.15$ pertencentes ao modelo do processo PC podem ter o evento enviado perdido. Isto pode ser observado nos arcos de entrada dos lugares de comunicação $ackDraft$, $accept$ e $reject$ através da chamada da função IC com a presença do valor negativo -1 no terceiro parâmetro.

Ao disparar as transições, a duração necessária para que o evento relacionado à conclusão da execução da atividade associada à transição disparada se torne verdadeiro é acrescentada aos objetos produzidos nos lugares de saída da transição através da execução da função E. Tal duração é obtida de forma exponencial através da execução da função tmE presente na função C considerando as durações médias associadas às transições. O tempo médio de execução de um Caso, considerando as durações médias associadas às transições descritas a seguir, é de 100 unidades de tempo. O valor de tais durações médias são:

- ❏ transição $sd.1$ - duração média = 3 unidades de tempo;
- ❏ transição $rad.2$ - duração média = 3 unidades de tempo;
- ❏ transição $ra.3$ - duração média = 3 unidades de tempo;
- ❏ transição $rr.4$ - duração média = 3 unidades de tempo;
- ❏ transição $pfv.5$ - duração média = 20 unidades de tempo;
- ❏ transição $rn1.6$ - duração média = 3 unidades de tempo;
- ❏ transição $sfv.7$ - duração média = 3 unidades de tempo;
- ❏ transição $rn2.8$ - duração média = 3 unidades de tempo;
- ❏ transição $raf.9$ - duração média = 3 unidades de tempo;
- ❏ transição $rn3.10$ - duração média = 3 unidades de tempo;
- ❏ transição $rd.11$ - duração média = 3 unidades de tempo;

- transição $sad.12$ - duração média = 3 unidades de tempo;
- transição $ev.13$ - duração média = 25 unidades de tempo;
- transição $sa.14$ - duração média = 3 unidades de tempo;
- transição $sr.15$ - duração média = 3 unidades de tempo;
- transição $rfv.16$ - duração média = 3 unidades de tempo;
- transição $tl.17$ - duração média = 3 unidades de tempo;
- transição $saf.18$ - duração média = 3 unidades de tempo.

Nota-se que no modelo, os identificadores dos lugares e das transições utilizados nas funções descritas no Apêndice B são mostrados nos nomes dos lugares e das transições para facilitar o entendimento do mesmo. Nos lugares os identificadores são mostrados antes do nome atribuído ao lugar e nas transições após ponto.

Por fim, as funções de autorização descritas na Tabela 3 referente ao modelo apresentado na Figura 38 são agrupadas na função $funAut$, a qual é descrita na Seção C.4. A função $funAut$ é responsável por definir, no momento da execução do modelo do processo, se a função de autorização associada à transição em questão é avaliada como *verdadeira*, *incerta* ou $falsa$. Neste modelo, apenas as funções de autorização associadas às transições com identificadores 2, 3 e 5 podem ser avaliadas, além de *verdadeira* e $falsa$, como *incerta*. Isto significa que somente estas transições podem ser disparadas de forma incerta e em todas as outras transições nenhum disparo incerto pode ser realizado.

6.2.2 Edição de IOWF-net Possibilística que aceita Desvios em casos de Deadlocks Estruturais

A edição no CPN Tools dos modelos dos processos AU e PC, que compõem a IOWF-net possibilística referente ao processo que precede a apresentação de um artigo em uma conferência definido por (AALST, 1998b), são apresentados, respectivamente, nas Figuras 56 e 57. A transição responsável por gerenciar o tempo corrente entre os modelos é definida em outra subpágina na área de trabalho do CPN Tools a qual é mostrada na Figura 58.

Os lugares de comunicação entre os modelos dos processos são representados através dos lugares de fusão presentes tanto no modelo do processo AU quanto no do processo PC. São eles: $draft$, $ackDraft$, $reject$, $accept$, $tooLate$, $finalVersion$ e $ackFinal$. Observa-se que os lugares de controle $C.AU$ e $C.PC$ são considerados, também, como lugares de fusão. Isto ocorre devido à presença da transição *time* necessária para gerenciar o tempo corrente de ambos os modelos.

Ao disparar as transições, a duração necessária para que o evento relacionado à conclusão da execução da atividade associada à transição disparada se torne verdadeiro é

acrescentada aos objetos produzidos nos lugares de saída da transição através da execução da função E. Tal duração é obtida de forma exponencial através da execução da função tmE presente na função C considerando as durações médias associadas às transições. O tempo médio de execução de um Caso, considerando as durações médias associadas às

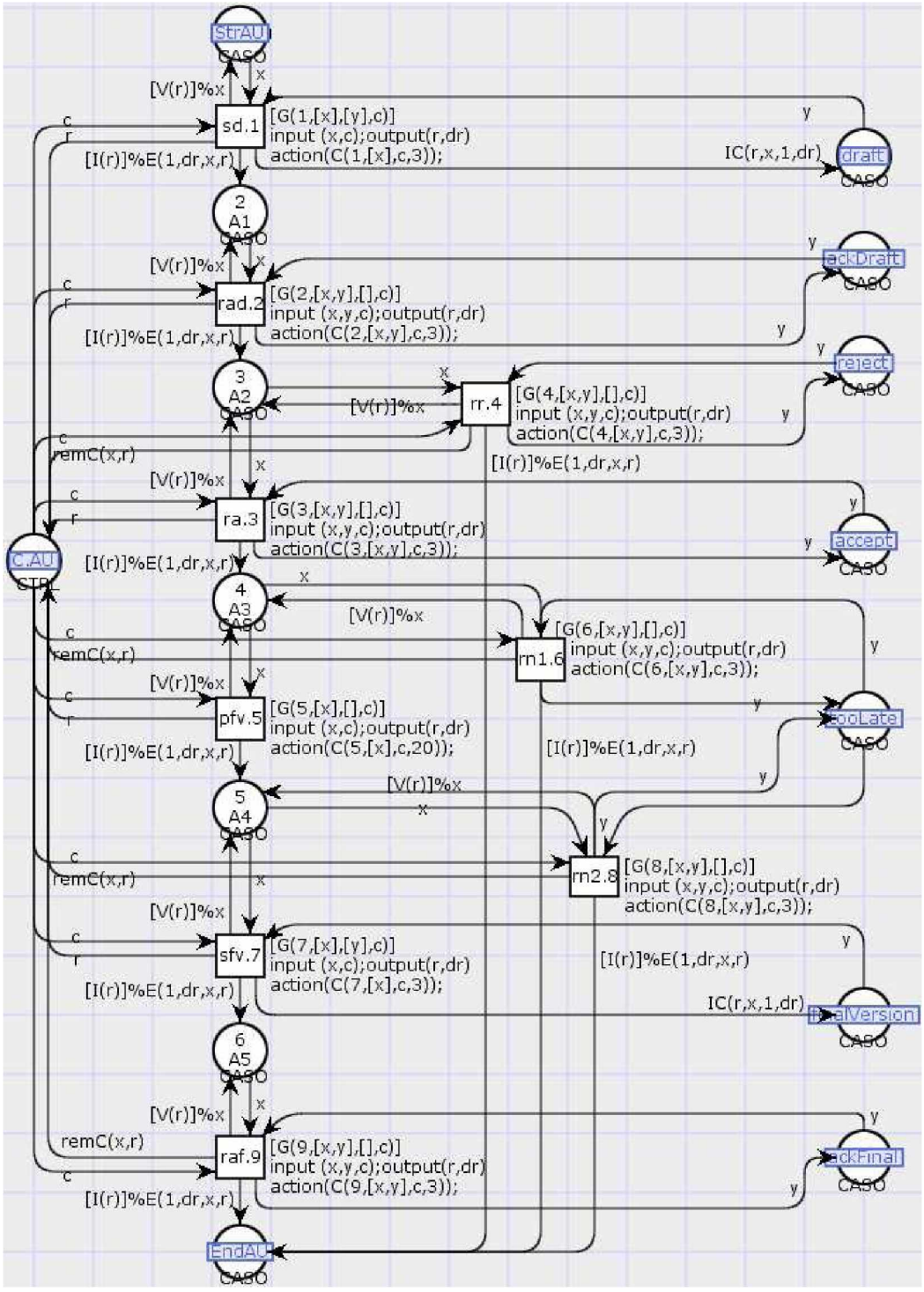


Figura 56 – IOWF-net possibilística *Relaxed Soundness* do processo AU editada no CPN Tools referente ao processo que precede a apresentação de um artigo em uma conferência.

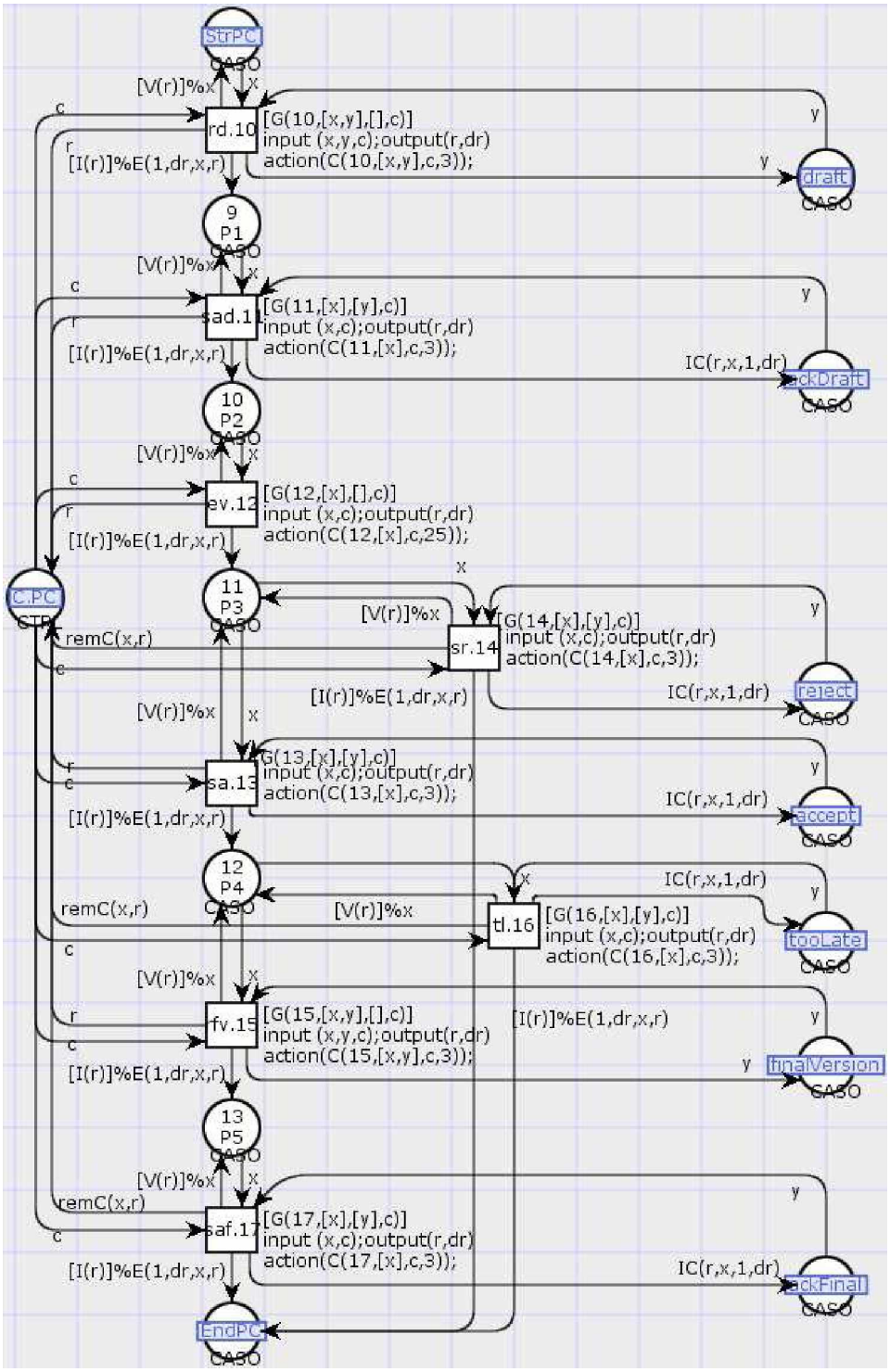


Figura 57 – IOWF-net possibilística *Relaxed Soundness* do processo *PC* editada no CPN Tools referente ao processo que precede a apresentação de um artigo em uma conferência.

transições descritas a seguir, é de 100 unidades de tempo. O valor de tais durações médias são:

- transição $sd.1$ - duração média = 3 unidades de tempo;

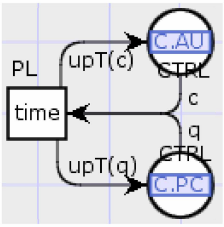


Figura 58 – Transição responsável por gerenciar o tempo corrente entre os modelos dos processos AU (Figura 56) e PC (Figura 57).

- ❏ transição $rad.2$ - duração média = 3 unidades de tempo;
- ❏ transição $ra.3$ - duração média = 3 unidades de tempo;
- ❏ transição $rr.4$ - duração média = 3 unidades de tempo;
- ❏ transição $pfv.5$ - duração média = 20 unidades de tempo;
- ❏ transição $rn1.6$ - duração média = 3 unidades de tempo;
- ❏ transição $sfv.7$ - duração média = 3 unidades de tempo;
- ❏ transição $rn2.8$ - duração média = 3 unidades de tempo;
- ❏ transição $raf.9$ - duração média = 3 unidades de tempo;
- ❏ transição $rd.10$ - duração média = 3 unidades de tempo;
- ❏ transição $sad.11$ - duração média = 3 unidades de tempo;
- ❏ transição $ev.12$ - duração média = 25 unidades de tempo;
- ❏ transição $sa.13$ - duração média = 3 unidades de tempo;
- ❏ transição $sr.14$ - duração média = 3 unidades de tempo;
- ❏ transição $rfv.15$ - duração média = 3 unidades de tempo;
- ❏ transição $tl.16$ - duração média = 3 unidades de tempo;
- ❏ transição $saf.17$ - duração média = 3 unidades de tempo.

Nota-se que no modelo, os identificadores dos lugares e das transições utilizados nas funções anteriormente descritas são mostrados nos nomes dos lugares e das transições para facilitar o entendimento do mesmo. Nos lugares os identificadores são mostrados antes do nome atribuído ao lugar e nas transições os identificadores são mostrados após o ponto.

Por fim, as funções de autorização descritas na Tabela 6 referente ao modelo apresentado na 47 são agrupadas na função $funAut$, a qual é descrita na Seção C.5. A função

funAut é responsável por definir, no momento da execução do modelo do processo, se a função de autorização associada à transição em questão é avaliada como *verdadeira*, *incerta* ou *falsa*. Lembra-se que neste modelo apenas a transição com identificador 7 que pode ser disparada de forma incerta. Este disparo incerto permite evitar a ocorrência de um estado de *deadlock* dado que ele ou é finalizado quando o reconhecimento do recebimento da versão final por parte do modelo do processo *PC* é enviado ao modelo do processo *AU*, ou cancelado quando o tempo limite para o envio da versão final é alcançado. Desta forma, o possível erro de sincronização após o envio dos eventos *too_late* e *final_version* é contornado permitindo concluir a execução do Caso em questão. Nas outras transições nenhum disparo incerto pode ser realizado.

6.3 Simulação de Processos baseado na execução/evolução dos Modelos e Avaliação dos Resultados

Com a intenção de expressar melhor a realidade dentro de uma empresa, os Casos são produzidos com uma dada frequência f. Levando em consideração 24 horas de execução com novos Casos chegando com uma frequência $f = 2$ minutos, 720 Casos são gerados em intervalos distribuídos de forma exponencial com tempo médio de 2 minutos.

A fim de produzir os 720 Casos de tempos em tempos, uma transição nomeada *arv* e um lugar nomeado *next* do tipo *CASO* juntamente com duas funções, uma associada à guarda e outra ao segmento de código desta transição, e, pelo menos, duas variáveis são definidas para produzir um novo Caso a ser executado. A declaração das variáveis usadas para simular a evolução de um determinado processo é dada juntamente com a descrição da execução dos modelos do mesmo. A declaração da função associada à guarda da transição *arv*, nomeada *gI*, e da função associada ao segmento de código, nomeada *init*, são dadas a seguir:

```
fun gI(idC, time, ctrl:CTRL) =
    let val tm = #1(ctrl)
    in if (time = tm) andalso (idC < 721)
       then true
       else false
    end

fun init(idC, str, (tm, tyF, Df, TDf, Lm, ILPn):CTRL) =
    (tm, tyF, Df, TDf, [(idC,[(str,0)])]^^Lm,
     [(idC,cIL(tm))]^^ILPn)
```

A função *gI* permite o disparo da transição *arv* somente se o tempo corrente alcançou o tempo estipulado por *time* ($time = tm$) e a quantidade produzida de Casos é inferior à

721 ($idC < 721$). Se uma destas verificações não é válida, a transição arv não é disparada.

A função $init$ é responsável por retornar ao lugar C um objeto atualizado com os dados do novo Caso. Esta atualização ocorre na lista de marcação representada pela variável Lm e na lista de inteiros $ILPn$ que contém, para cada Caso em execução, as constantes usadas nas funções de autorização associadas às transições. O produto $(id, [(str, 0)])$ é inserido em Lm de forma a informar que o novo Caso, determinado pelo identificador idC, se encontra no lugar definido pelo identificador str. A função cIL é responsável por definir a lista de constantes para o novo Caso. A declaração da mesma será dada em conjunto com a execução do modelo em questão.

Além das funções gI e $init$, a função tmE é associada à expressão do arco de entrada do lugar $next$ a fim de definir a chegada do próximo Caso a ser executado. Como o tempo médio estipulado é de dois em dois minutos, o parâmetro da função tmE tem como valor o inteiro 2.

Para executar os modelos, os lugares de inicialização e de controle têm, respectivamente, como marcação inicial os objetos $(1, 0)$ e $(0, 0, (0, 0), ([\,], [\,]), [\,], [\,])$. As duas variáveis presentes no produto $(1, 0)$ se referem, nesta ordem, ao identificador do Caso e ao tempo de chegada do evento relacionado ao objeto em questão. As variáveis presentes no segundo produto se referem:

1. ao tempo corrente;
2. ao tipo de disparo;
3. ao identificador do Caso e à transição responsável por inicializar a execução do procedimento de defuzzificação neste Caso;
4. às listas das transições a terem o disparo incerto finalizado ou cancelado;
5. à lista de marcação da rede;
6. à lista de inteiros usados nas funções de autorização associadas às transições.

A fim de acompanhar a evolução de alguns parâmetros durante a execução de um modelo no CPN Tools, este permite a criação de "monitores" para coleta de dados através do *Data Collector Monitoring Functions*. Este coletor apresenta funções que são escritas em códigos descritos na linguagem CPN ML para atender necessidades específicas. Um "monitor" inclui uma função de predicado (*predicate function*), que determina quando o dado deve ser coletado, e uma função de observação (*observation function*) que determina qual dado deve ser coletado. Assim, as informações obtidas através de um determinado "monitor" durante a execução de um modelo são informados em conjunto com a descrição da evolução do mesmo.

Por fim, nas subseções subsequentes, os modelos anteriormente editados no CPN Tools são executados e os resultados obtidos são avaliados e comparados com os resultados

quando não se considera as interpretações incertas. As execuções dos modelos editados nas subseções 6.1.1, 6.1.2, 6.1.3, 6.2.1 e 6.2.2 são apresentadas, respectivamente, nas subseções 6.3.1, 6.3.2, 6.3.3, 6.3.4 e 6.3.5.

6.3.1 Resultado da Simulação de processo modelado em WF-net Possibilística que aceita Desvios

O lugar *next* e a transição *arv*, responsáveis por gerar os novos Casos para o processo de tratamento de reclamações (modelo apresentado na Figura 50), são mostrados na Figura 59. A declaração das variáveis *id* e *z*, usadas na transição *arv* e nos arcos de entrada e de saída do lugar *next*, e a declaração da função de *cIL*, responsável por definir os tempos máximos de espera usados nas funções de autorização associadas às transições do Caso em questão, são dadas abaixo:

```
var id, z: INT;

fun cIL(tm) = [0      , 0      , 10+tm, 40+tm, 10+tm, 30+tm,
               40+tm, 50+tm, 50+tm, 70+tm, 70+tm, 0      ,
               70+tm, 0      , 90+tm, 0      ]
```

A função *cIL* determina, caso o valor seja maior que zero, o momento em que um disparo incerto pode ser realizado. Para determinar este valor, o tempo corrente é atribuído ao parâmetro *tm*. Lembre-se que os valores iguais a zero indicam que a transição em questão não pode ser pseudo-disparada.

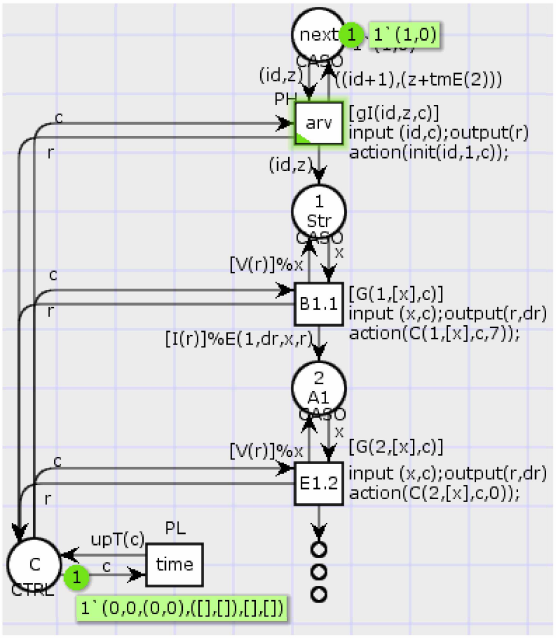


Figura 59 – Lugar *next* e transição *arv* responsáveis por produzir um novo Caso no modelo apresentado na Figura 50.

Por fim, o modelo descrito na Figura 50 acrescido do lugar *next* e da transição *arv*, tal como mostrado na Figura 59, é executado considerando as interpretações incertas. A Figura 60 mostra, no tempo corrente $\tau = 49$, a replicação registrada no CPN Tools do estado da execução do modelo apresentado na Figura 50. Considerando este estado em particular, 8 observações podem ser realizadas. São elas:

1. um novo Caso de $id = 31$ será produzido quando o tempo corrente for igual ao valor 52;

2. 30 Casos estão em execução[10];

3. os Casos em execução estão localizados nos seguintes lugares: 4 objetos em $A1$, 9 em $A2$, 3 em $W3$, 5 em $A3$, 8 em $W4$, 6 em $A4$, 7 em $A5$ e 2 em $A6$;

4. ainda não foi concluída a execução de nenhum Caso, ou seja, o lugar *End* contém 0 Casos;

5. o Caso com identificador 16 se localiza tanto no lugar $A3$ quanto no lugar $W4$. Isto indica uma marcação imprecisa para este Caso, a qual é obtida através do disparo incerto realizado na transição $E3$. Tal disparo foi efetuado considerando que no tempo limite definido para este Caso nesta transição, através da função cIL, o evento relacionado a $E3$ ainda é tido como falso $(16, -1)$, ou seja, o contato com o cliente não foi possível;

6. tal como o evento relacionado à transição $E3$ para o Caso com identificador 16, o evento relacionado a esta transição para o Caso com identificador 19 também é tido como falso $(19, -1)$. Logo apenas um disparo incerto poderá ser realizado em $E3$ para este Caso quando o tempo corrente alcançar o tempo limite de espera;

7. o Caso com identificador 9 se localiza tanto no lugar $A3$ quanto no lugar $W4$. Diferentemente do Caso com identificador 16 que teve um disparo incerto devido à perda do evento $(16, -1)$, esta marcação imprecisa é obtida devido à demora, considerando o tempo limite estipulado pela função cIL, em finalizar o contato com o cliente. Desta forma, um disparo incerto foi realizado com o objetivo de iniciar a atividade $A4$ quando o contato com o departamento for concluído;

8. idem ao item anterior, mas em relação à demora para finalizar o contato com o departamento, o Caso com identificador 6 se localiza nos lugares $A2$ e $W3$ de forma imprecisa devido ao disparo incerto da transição $E3$. Tendo em vista que tal Caso se encontra no lugar $W4$ (note-se que de forma precisa) e que o tempo limite permitido para esperar o contato com o departamento foi alcançado, a transição $B4$ está apta para ser disparada considerando este Caso, tal como destacado na Figura 50.

[10] Objetos em lugares diferentes com mesmo identificador se referem a um mesmo Caso em execução.

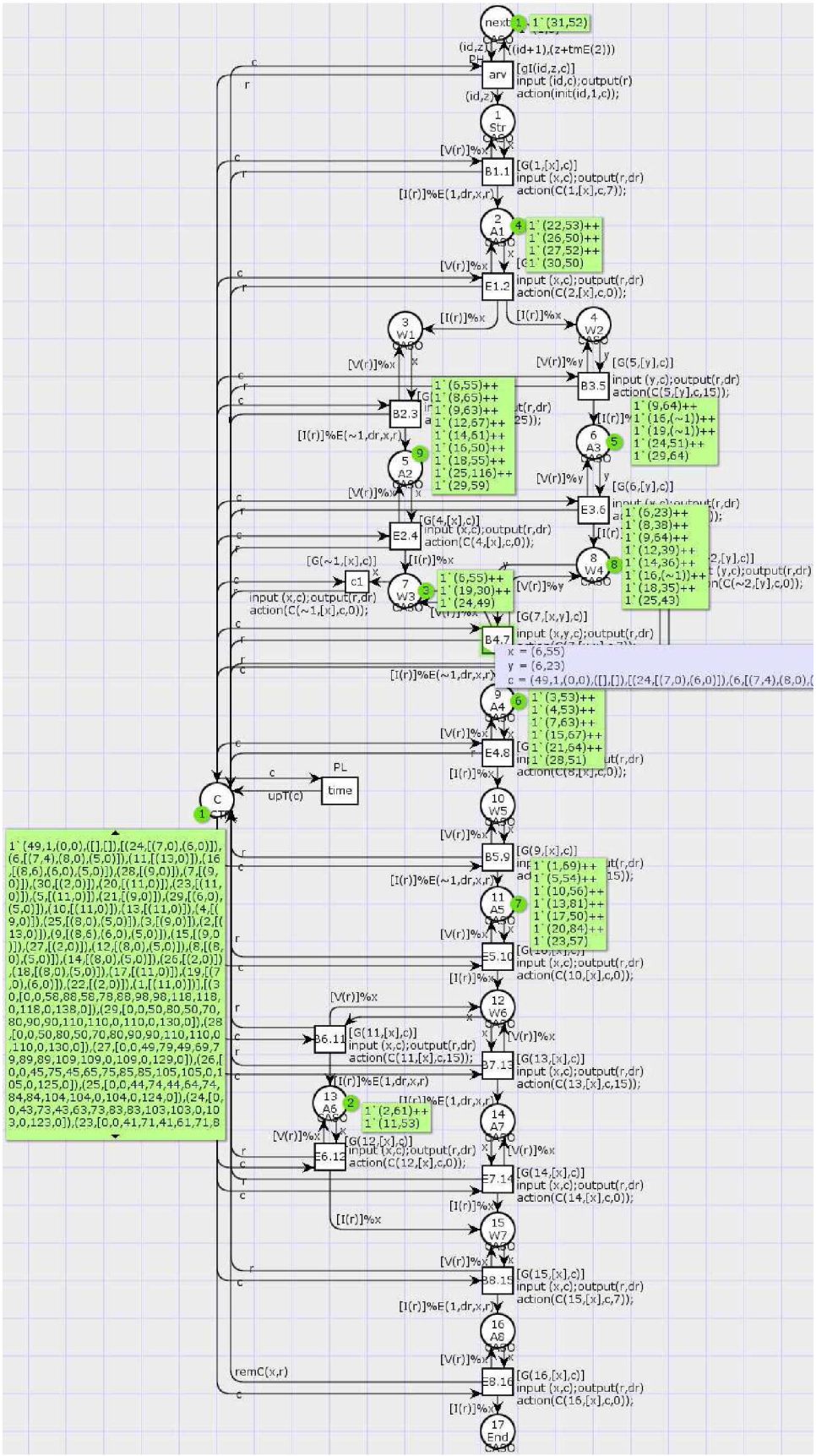


Figura 60 – Visualização da execução do modelo no CPN Tools, mostrado na Figura 50, no tempo corrente $\tau = 49$.

Levando em consideração que a transição $B4$ permite apenas disparos certos e que a mesma está habilitada por uma marcação imprecisa, o algoritmo de defuzzificação é então inicialmente chamado para posteriormente dispará-la de forma certa. A Figura 61 mostra, através da marcação do lugar de controle exibida na variável c, no tempo corrente 49, que o Caso de identificador 6 está em processo de defuzzificação por causa da transição de identificador 7. Além disto, o disparo incerto realizado na transição $E2$, representada pelo identificador 4 e destacada na figura, será finalizado e não existe qualquer outro disparo incerto a ser cancelado.

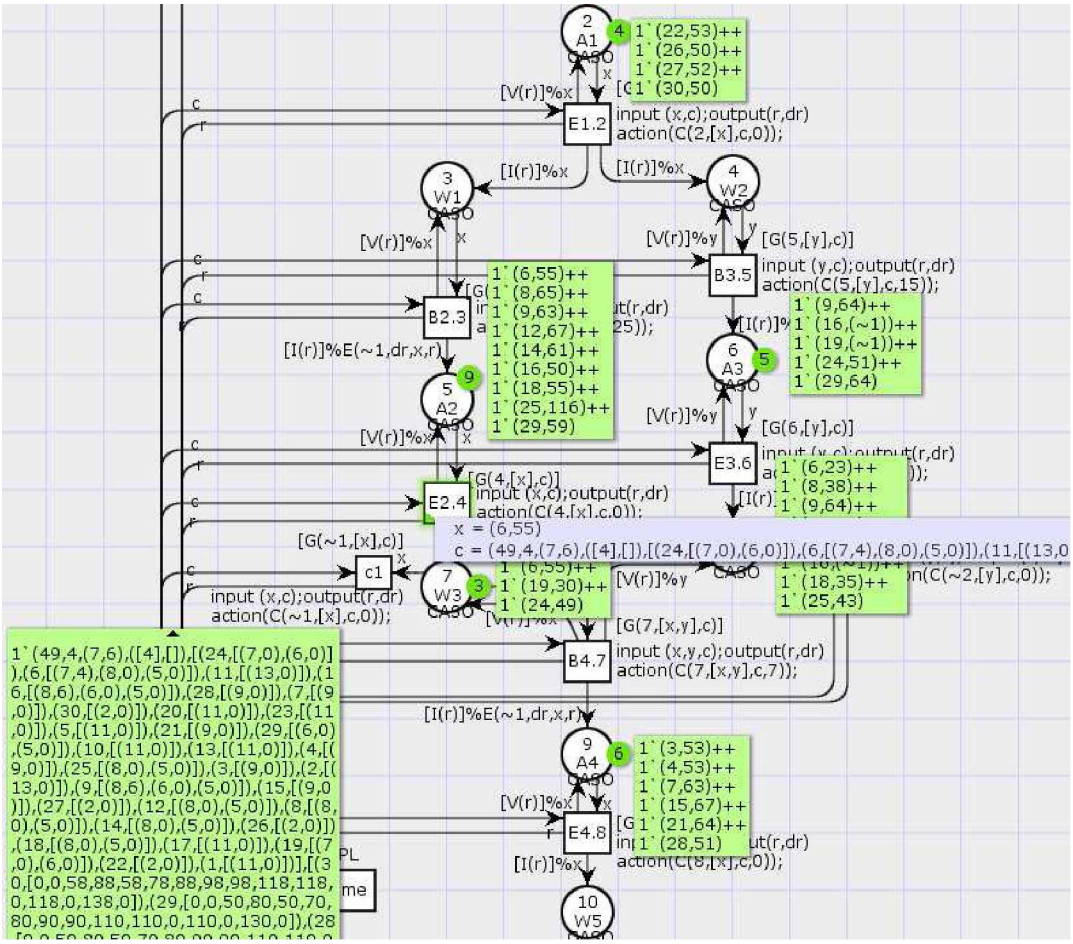


Figura 61 – Transição $E2$ apta a ter o disparo incerto finalizado no tempo corrente $\tau = 49$ da execução do modelo mostrado na Figura 50.

Ao observar a Figura 62, pode-se notar que o procedimento de defuzzificação foi finalizado e a transição $B4$, além de estar apta a ser disparada, está habilitada de forma precisa. Deste modo, a mesma será disparada de forma certa.

A Figura 63 mostra a conclusão da execução dos 720 Casos produzidos no lugar Str. Note-se que o último Caso a concluir a execução foi o Caso com identificador igual à 718 no tempo corrente $\tau = 1622$.

A fim de obter informações tais como o identificador do(s) Caso(s) responsável(is) pelo disparo de uma transição t, o momento e o tipo deste disparo além do momento em

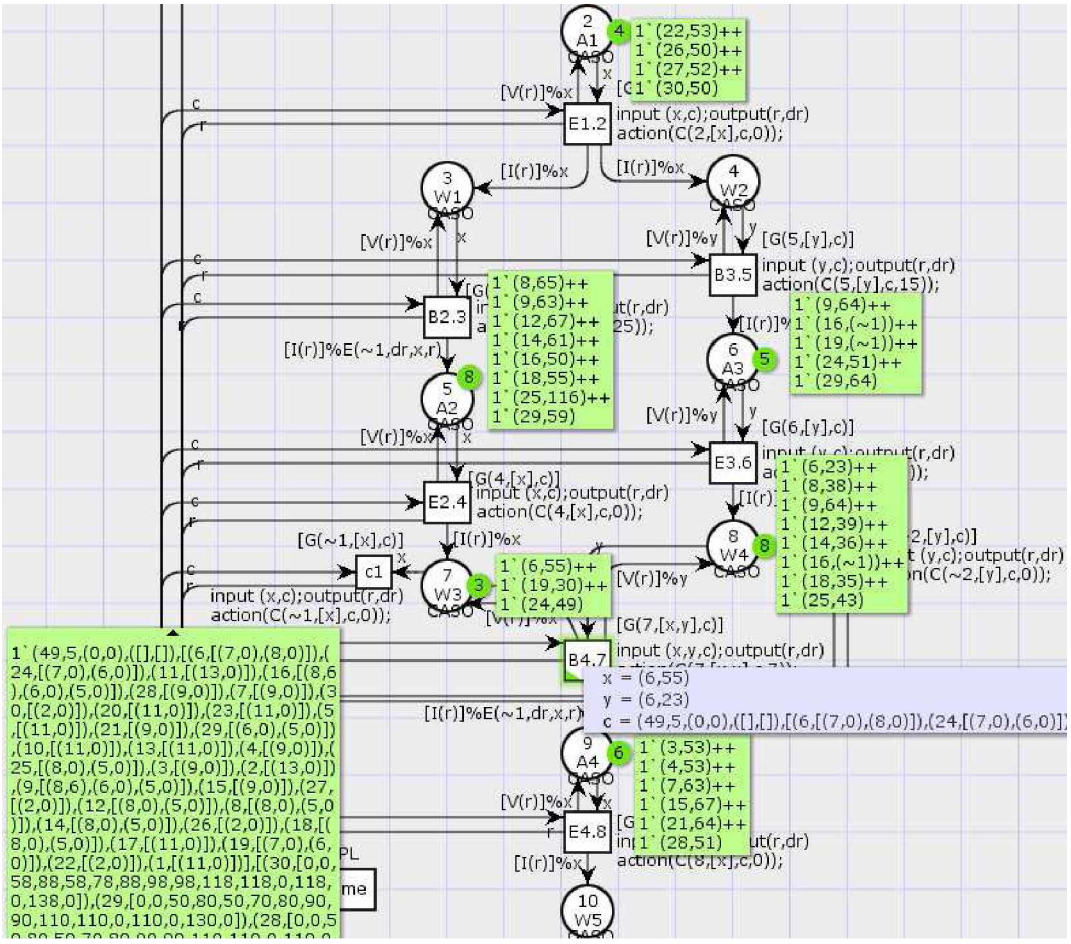


Figura 62 – Transição $B4$ apta a ser disparada de forma certa no tempo corrente $\tau = 49$ da execução do modelo mostrado na Figura 50.

que o(s) evento(s) relacionado(s) a t se torna(m) verdade, um monitor é associado a cada transição pertencente ao modelo. As informações obtidas permitem verificar a quantidade de Casos e quais transições tiveram, durante a execução do modelo, eventos perdidos e/ou atrasados.

A Figura 64 mostra a quantidade de Casos finalizados que tiveram no mínimo 3 a no máximo todos os eventos recebidos no tempo esperado durante a execução do modelo mostrado na Figura 50. Observa-se que um pouco mais de 60% dos Casos produzidos (435 Casos) foram concluídos com pelo menos 1 evento perdido[11] ou atrasado durante a simulação. Destes 60% dos Casos, 81,8% (316 Casos) tiveram entre 1 a 5 eventos perdidos ou atrasados.

Levando em consideração a Figura 64 e analisando a Figura 65, pode-se concluir que 16 Casos tiveram pelo menos um evento perdido dado que, pela Figura 65, nenhum evento chegou atrasado em 301 Casos e, pela Figura 64, foram 285 Casos que tiveram todos os eventos esperados recebidos dentro do tempo estipulado. Além disto pode-se observar

[11] Um evento é considerado como perdido quando o tempo de ocorrência do mesmo é indicado no modelo com o valor negativo -1. Este valor é obtido através da execução da função E (descrita na Seção B.1) associada aos arcos de saída da transição.

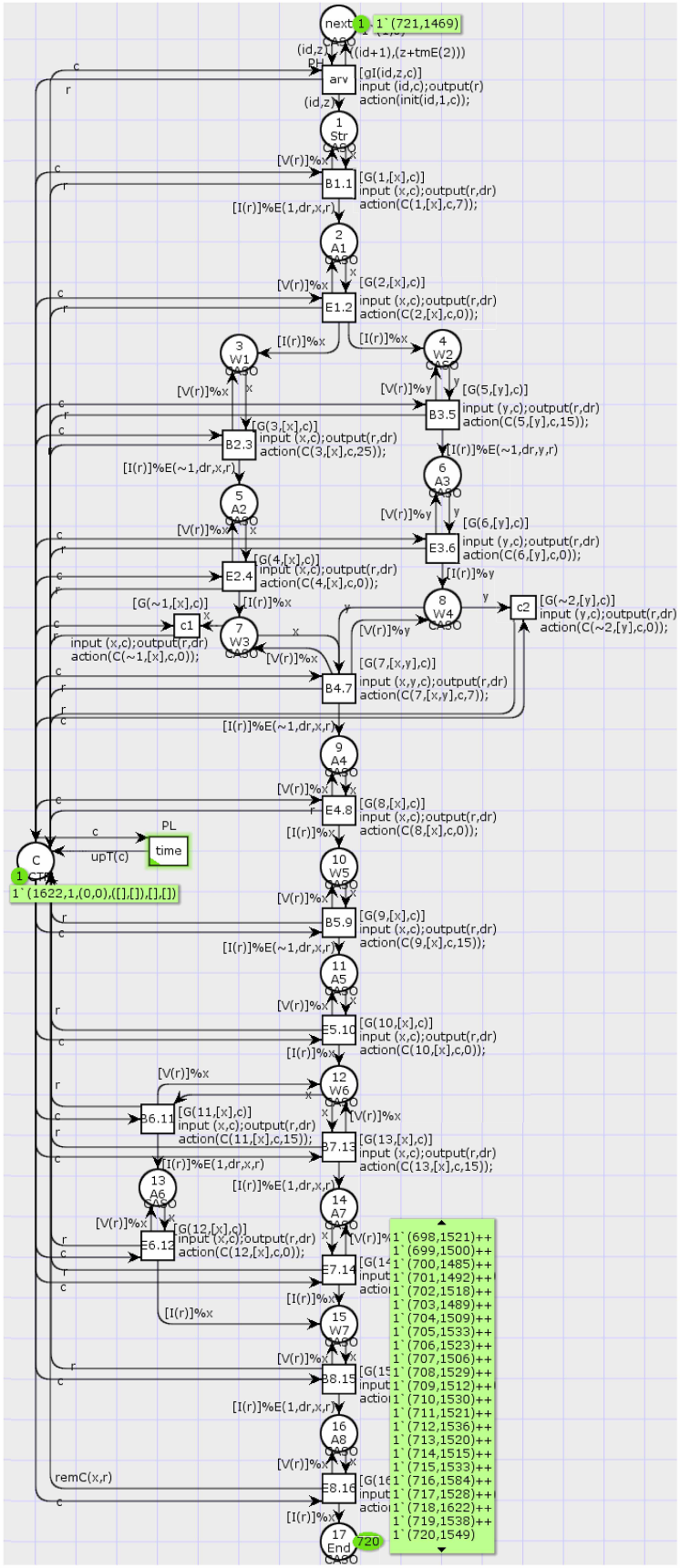


Figura 63 – Visualização da execução do modelo no CPN Tools, mostrado na Figura 50, após a execução dos 720 Casos produzidos.

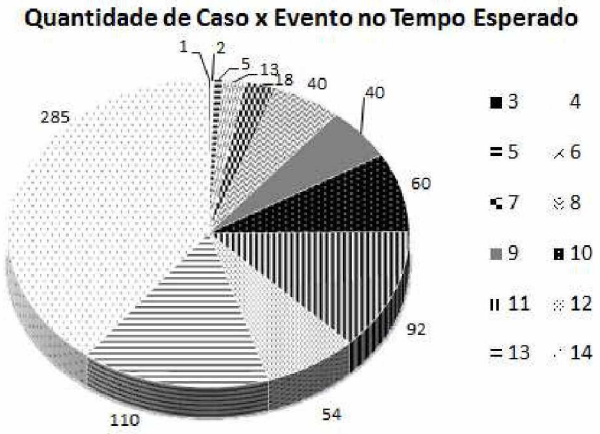


Figura 64 – Quantidade de Casos finalizados que tiveram no mínimo 3 a no máximo todos os eventos recebidos no tempo esperado durante a execução do modelo mostrado na Figura 50.

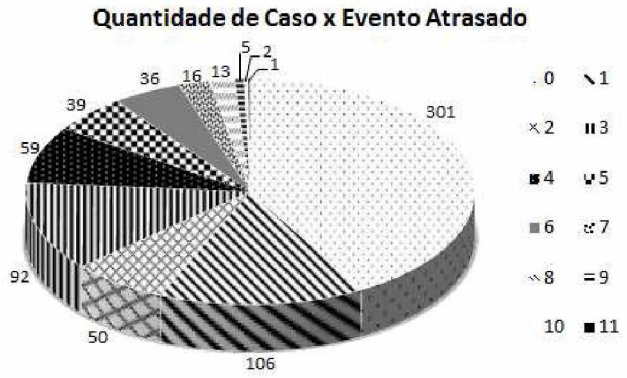


Figura 65 – Quantidade de Casos finalizados que tiveram nenhum a no máximo 11 eventos recebidos atrasados durante a execução do modelo mostrado na Figura 50.

que, dos 58, 2% Casos restantes (419 Casos), menos de 18% (73 Casos) tiveram mais de 6 eventos atrasados.

Ao analisar a Figura 66, pode-se observar que os eventos perdidos foram principalmente os associados à conclusão das atividades $A2$, $A3$, $A4$ e $A5$. Os outros eventos perdidos foram os associados ao início das atividades $A5$, $A6$ e $A7$. Isto ocorre porque os eventos relacionados ao início de uma atividade são diretamente influenciados pelos eventos relacionados à conclusão das atividades imediatamente anteriores, no caso $A4$ e $A5$.

Ao fim da execução, como pode ser visto na Figura 67, um total de 37 Casos tiveram 1 ou 2 eventos perdidos. Destes 37, 15 tiveram um evento perdido e 22 dois eventos perdidos. Ao confrontar as Figuras 66 e 67, pode-se notar que independentemente da perda ou atraso no recebimento dos eventos, todos os Casos produzidos foram completamente concluídos e mais de 90% dentro do prazo estipulado (653 Casos) comprovando o efeito do comportamento flexível do "jogador" de WF-net possibilística em caso de desvios no processo.

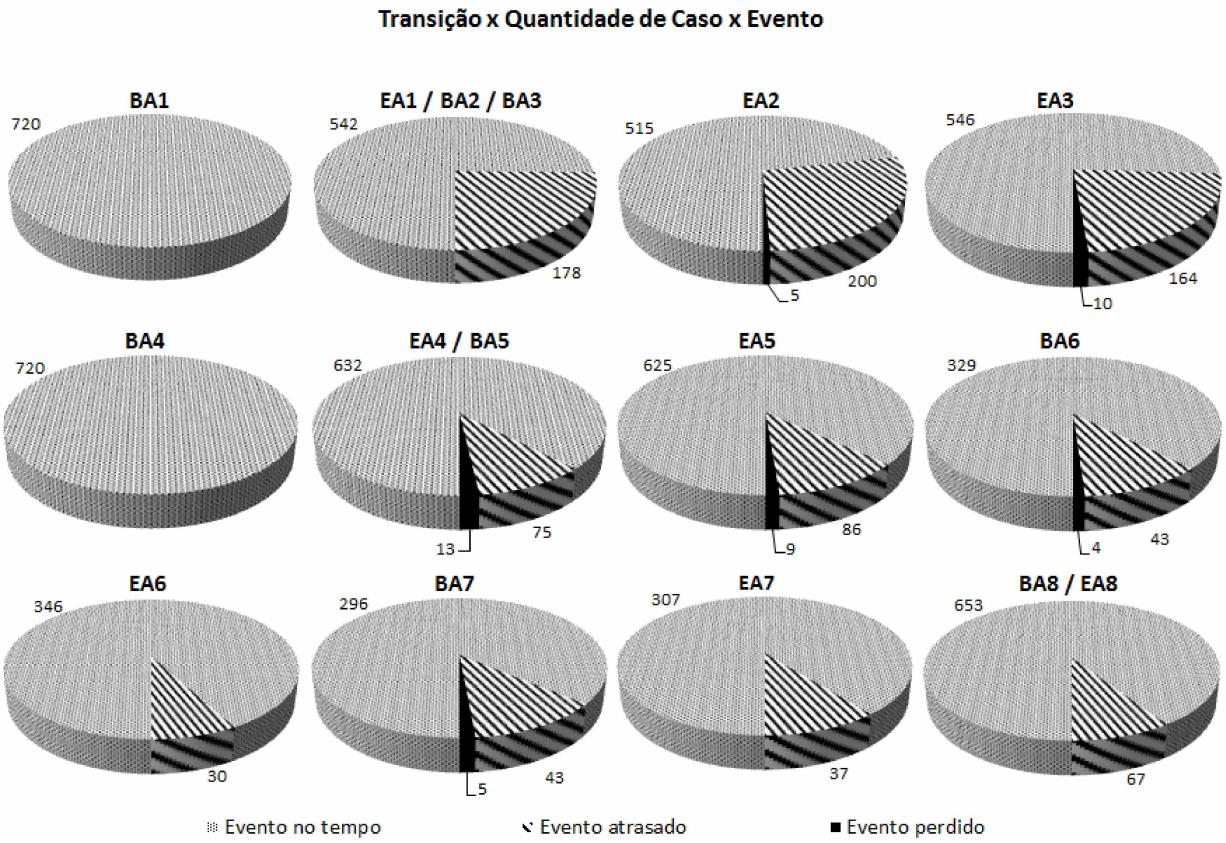


Figura 66 – Quantidade de Casos por transição que tiveram eventos perdidos, atrasados ou recebidos dentro do prazo estipulado durante a simulação do modelo mostrado na Figura 50.

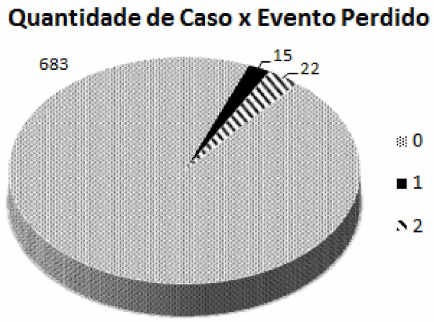


Figura 67 – Quantidade de Casos finalizados que tiveram nenhum, 1 ou 2 eventos perdidos durante a execução do modelo mostrado na Figura 50.

Para os resultados obtidos nesta execução, uma nova execução é realizada onde as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados. O modelo utilizado nesta nova execução é o mesmo apresentado na Figura 50, porém com as interpretações associadas às transições podendo ser avaliadas apenas como verdadeira ou falsa, ou seja, nenhuma interpretação pode ser avaliada como incerta. Além disto, os mesmos padrões de chegada de novos Casos foram consideradas.

Uma vez desconsideradas as interpretações incertas e, consequentemente, os disparos

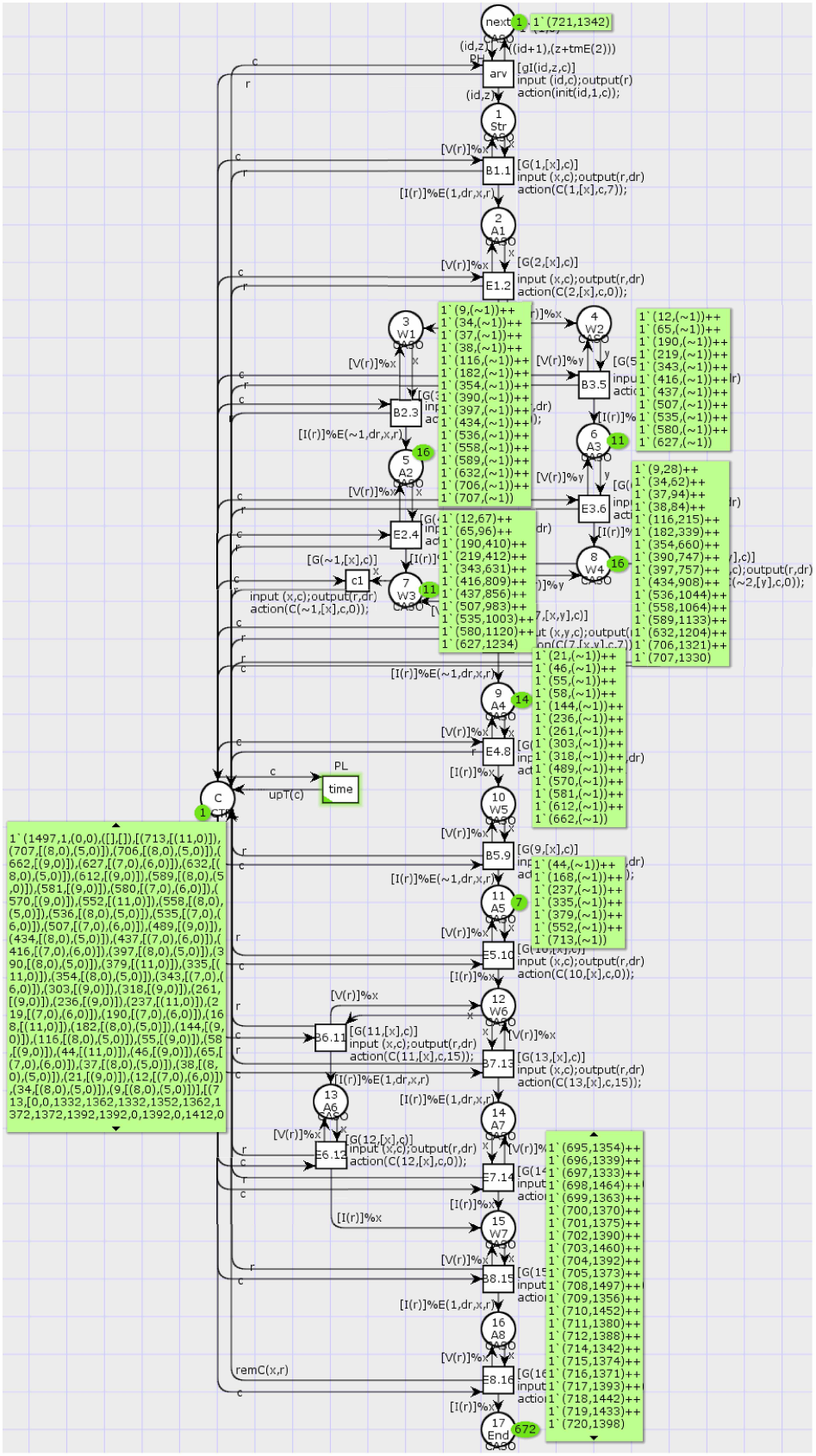


Figura 68 – Visualização da execução do modelo no CPN Tools, mostrado na Figura 50, após a execução dos 720 Casos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

incertos, alguns Casos podem ser concluídos com atraso ou nem ser concluídos. A Figura 68 mostra o estado do modelo após a execução dos 720 Casos produzidos no lugar *Str* quando as interpretações associadas às transições podem ser avaliadas apenas como verdadeira ou falsa. Note-se que 48 Casos (6, 7%) não concluíram sua execução devido ao não recebimento de um evento esperado. Destes 48 Casos, 27 pararam sua execução devido à falha ao contactar o departamento ou o cliente ($B4$), 14 pararam antes de concluir a coleta das informações obtidas através do departamento e do cliente ($E4$) e 7 pararam antes de definir se o cliente receberá um pagamento ou uma carta referente ao Caso em questão ($E5$).

Ao observar o grafo referente à transição $E8$ na Figura 69, percebe-se que 29% dos casos concluídos (195 Casos) foram posteriores ao prazo estipulado. Isto ocorre porque, no caso de uma simulação sem o uso de pseudo-disparos, se a chegada de um evento relacionado a conclusão de uma atividade se atrasa, é provável que todas as outras subsequentes também se atrasarão.

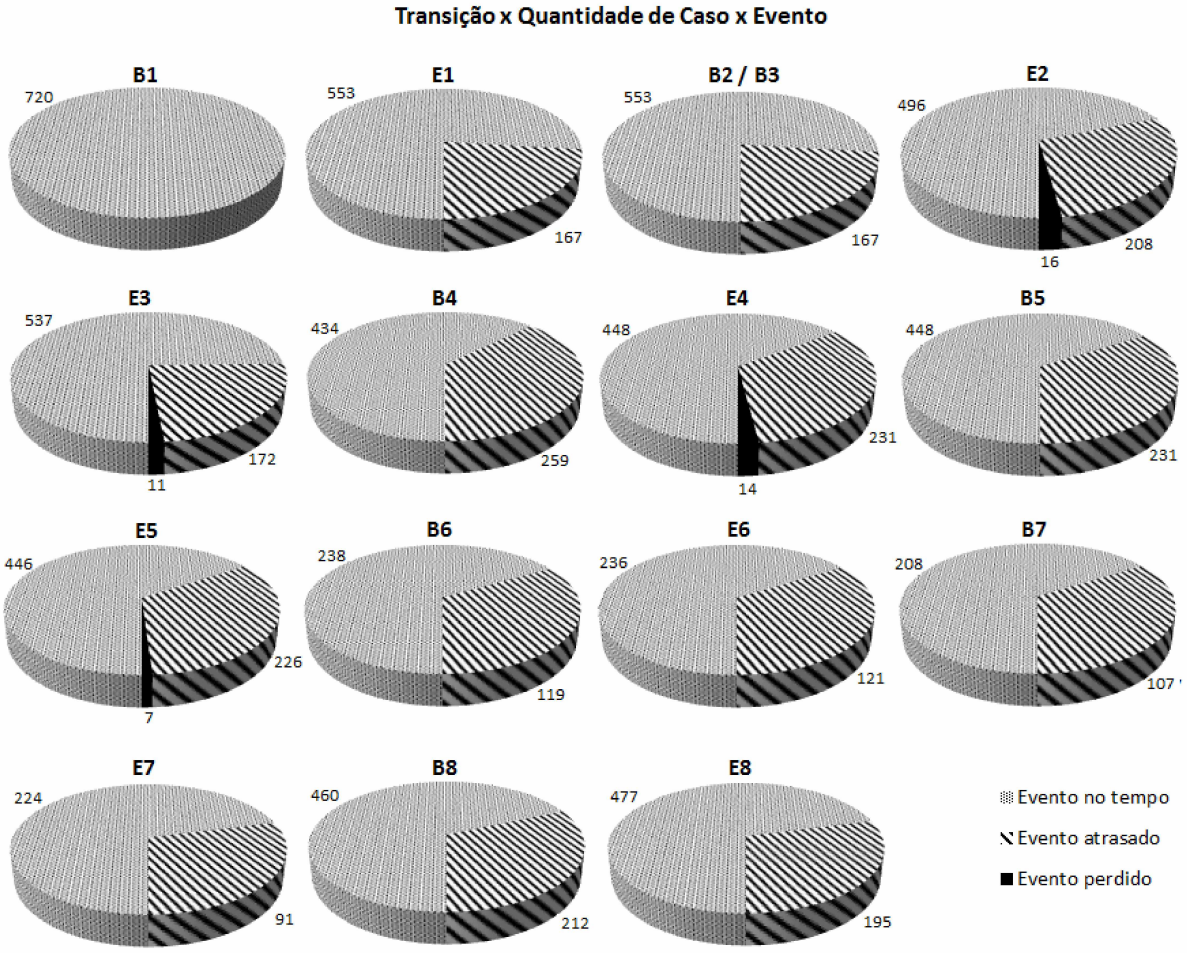


Figura 69 – Quantidade de Casos por transição que tiveram eventos perdidos, atrasados ou recebidos dentro do prazo estipulado durante a execução do modelo mostrado na Figura 50 quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

Levando ainda em consideração os Casos finalizados, na Figura 70, pode-se notar que 39,4% dos Casos finalizados tiveram todos eventos recebidos dentro do prazo estipulado. Em contrapartida, mais de 80% dos 60,6% Casos restantes tiveram no mínimo 2 a no máximo 10 eventos atrasados.

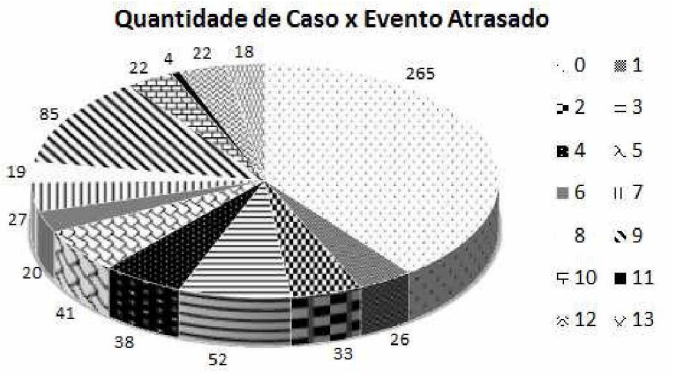


Figura 70 – Quantidade de Casos finalizados que tiveram no mínimo 1 a no máximo todos os eventos recebidos no tempo esperado durante a simulação do modelo mostrado na Figura 50 quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

Por fim, a Tabela 7 faz uma comparação entre os resultados obtidos através das duas execuções apresentadas. Pode-se observar que, a noção de disparo incerto permitiu, além de evitar estados inconsistentes, melhorar o desempenho dos processos. Esta melhora é obtida devido a característica dos disparos incertos de iniciar a execução de uma atividade antes do término da(s) anterior(es) ou deduzir o estado atual do processo a partir de um evento posterior.

Tabela 7 – Comparação entre os resultados obtidos através das duas execuções realizadas na WF-net Possibilística mostrada na Figura 50 considerando primeiramente a noção de disparo incerto e posteriormente desconsiderando-a

		Com a noção de disparo incerto	Sem a noção de disparo incerto
Quantidade	Casos finalizados	720	672
	Casos finalizados dentro do prazo	653	477
	Casos finalizados com todos eventos dentro do prazo	301	265

Nota-se na Tabela 7 que quando todos os eventos estão dentro do prazo, a quantidade de Casos finalizados se difere quando a noção de disparo incerto é considerada ou desconsiderada. Mesmo que os parâmetros de entrada sejam idênticos, de forma proporcional, 12 Casos a mais foram finalizados quando a noção de disparo incerto é considerada. Isto pode ser devido aos 48 Casos não finalizados por causa dos eventos perdidos quando a noção de disparo incerto é desconsiderada. Lembra-se que se um evento é perdido a execução do modelo do processo é continuada através dos disparos incertos considerando

os tempos limites definidos. Assim sendo, o disparo incerto permite manter os prazos estabelecidos mesmo quando um evento é perdido.

6.3.2 Resultado da Simulação de um Processo modelado em WF-net possibilística com Estrutura Sequencial que aceita Desvios para Exploração do Paralelismo inerente ao Processo

A Figura 71 mostra a inserção do lugar *next* e da transição *arv* no modelo de um processo de encomendas apresentado na Figura 51. A declaração das variáveis *id* e *z* é idêntica à descrita na subseção anterior. A declaração da função *cIL* descrita a seguir determina, nas primeiras 14 constantes inteiras, o momento em que um disparo incerto pode ser realizado, se maior que zero, e, nos outros 7 valores zerados restantes, o tempo de execução gasto das atividades pertencentes ao modelo.

```
fun cIL(tm) = [0     , 0     , 0     , 0     , 0     ,
               tm+25, tm+25, tm+50, tm+50, tm+75,
               tm+75, tm+90, 0     , 0     ,
               0     , 0     , 0     , 0     , 0     ,0     ,0]
```

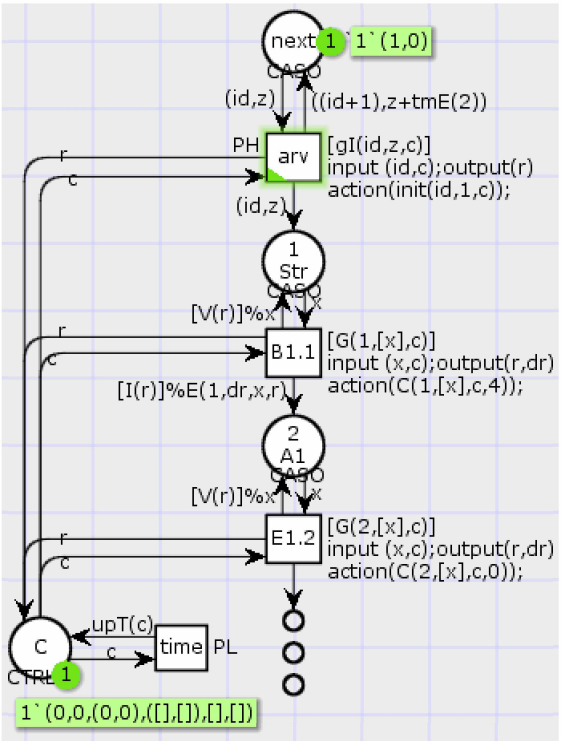


Figura 71 – Lugar *next* e transição *arv* responsáveis por produzir um novo Caso do tipo *CASO* no modelo apresentado na Figura 51.

O modelo descrito na Figura 51 acrescido do lugar *next* e da transição *arv*, tal como mostrado na Figura 71, é executado e a Figura 72 mostra, no tempo corrente $\tau = 99$,

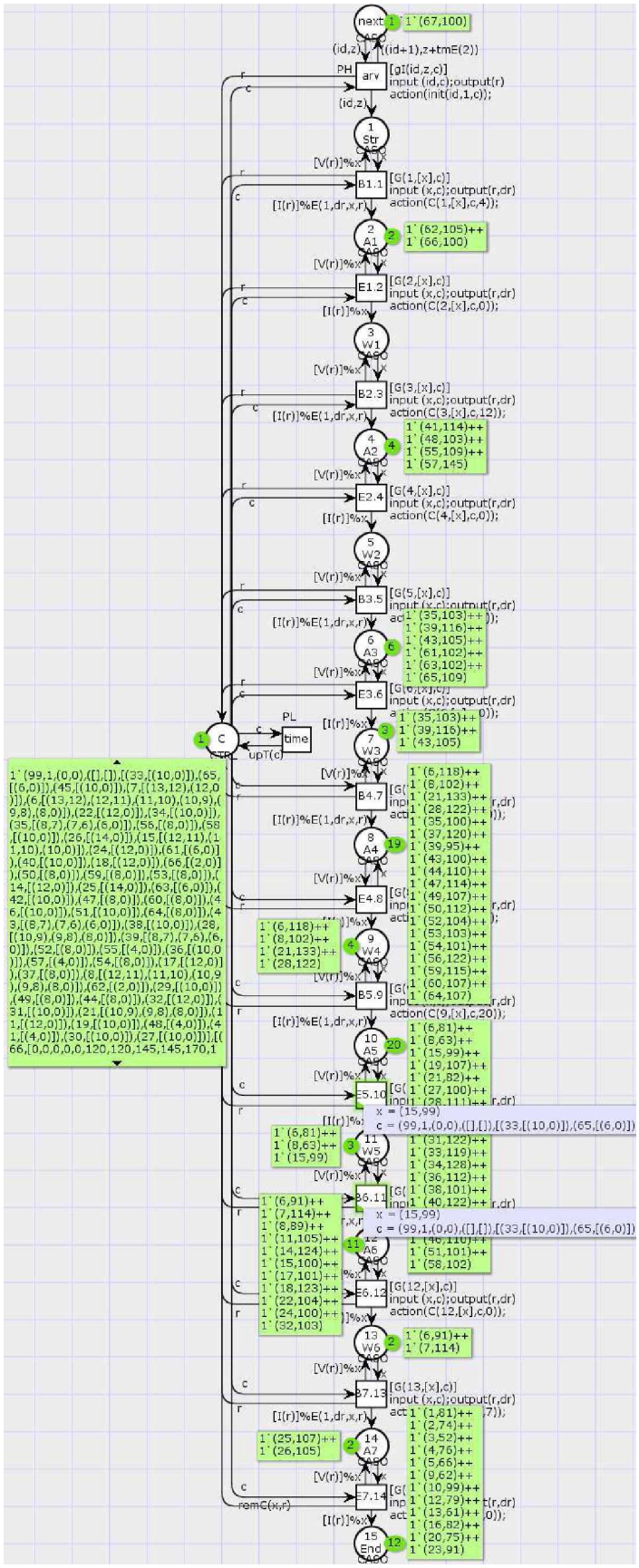


Figura 72 – Visualização da execução do modelo no CPN Tools, mostrado na Figura 51, no tempo corrente $\tau = 99$.

uma replicação registrada no CPN Tools do estado do modelo apresentado na Figura 51. Considerando este estado em particular, 7 observações podem ser realizadas. São elas:

1. um novo Caso de $id = 67$ será produzido quando o tempo corrente for igual ao valor 100;

2. 54 Casos estão em execução[12];

3. os Casos em execução estão localizados nos seguintes lugares: 2 objetos em $A1$, 4 em $A2$, 6 em $A3$, 3 em $W3$, 19 em $A4$, 4 em $W4$, 20 em $A5$, 3 em $W5$, 11 em $A6$, 2 em $W6$ e 2 em $A7$;

4. a execução de 12 Casos foram concluídas, ou seja, o lugar *End* contém 12 Casos;

5. os Casos com identificador iguais à 6, 8, 15, 21, 28, 35, 39 e 43 estão executando pelo menos duas atividades ao mesmo tempo. Isto pode ser observado, por exemplo, nos lugares $A3$, $W3$ e $A4$ para os Casos com identificadores iguais à 35, 39 e 43. Isto ocorre devido ao disparo incerto realizado nas transições $E3$ e $B4$ quando o tempo limite de espera para concluir a tarefa $A3$ foi alcançado por cada um deles;

6. o Caso com identificador 7 teve um disparo incerto realizado na transição $W6$, entretanto em $B7$ nenhum disparo foi realizado. Isto ocorre porque diferentemente das atividades $A4$, $A5$ e $A6$ que permitem o seu início antes da conclusão das suas anteriores, a atividade $A7$ só pode ser iniciada quando todas as anteriores forem concluídas. Desta forma, apenas quando a execução da atividade $A6$ finalizar, ou seja, quando o tempo corrente for igual à 114, que a atividade $A7$ pode ser iniciada;

7. o Caso com identificador 15 teve a execução da atividade $A5$ concluída. Assim sendo, a transição $E5$ está apta para ser disparada, tal como destacado na Figura 72. Entretanto, a transição $B6$ também está apta a ser disparada devido ao disparo incerto anteriormente realizado. Independentemente de qual seja a escolhida, um algoritmo de defuzzificação será chamado para que os disparos incertos realizados tanto em $E5$ quanto em $B6$ possam ser cancelados para, posteriormente, concluir a atividade $A5$ e prosseguir com a execução deste Caso.

A Figura 73 mostra, através da marcação do lugar de controle exibida na variável c, no tempo corrente 99, que o objeto de identificador 15 está em processo de defuzzificação por causa da transição de identificador 10, no caso $E5$. Além disto, nenhum disparo incerto é finalizado e os disparos incertos realizados nas transições $B6$ e $E5$ serão cancelados através do disparo das transições $E6$ e $B6$ representadas, respectivamente, pelos identificadores 12 e 11.

[12] Objetos em lugares diferentes com mesmo identificador se referem a um mesmo caso em execução.

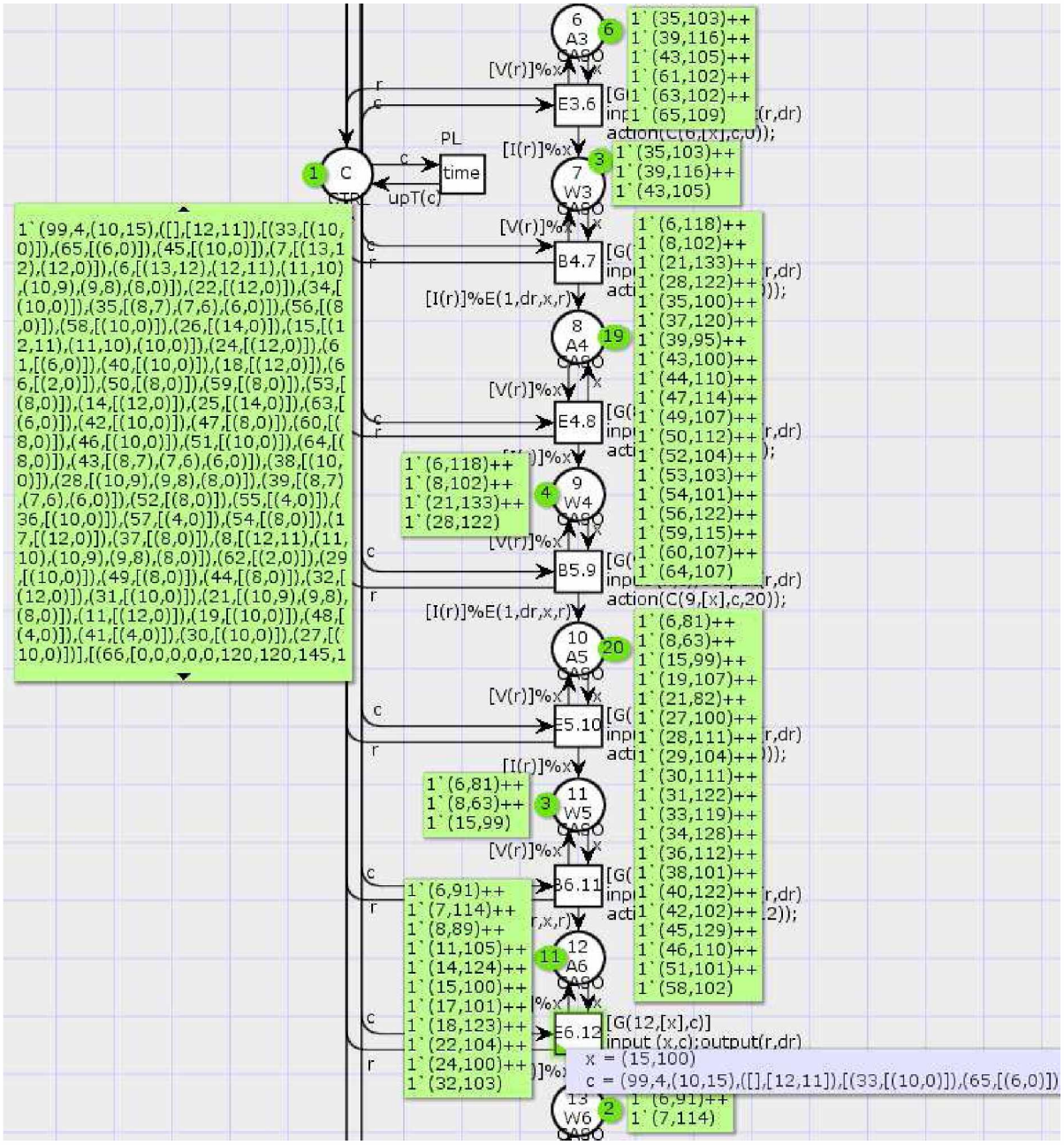


Figura 73 – Transição $B6$ apta a ter o disparo incerto cancelado através do disparo da transição $B6$ no tempo corrente $\tau = 99$ da execução do modelo mostrado na Figura 51.

Ao observar a Figura 74, pode-se notar que o procedimento de defuzzificação foi finalizado e a transição $E5$, além de estar apta a ser disparada, está habilitada de forma precisa. Deste modo, a mesma será disparada de forma certa.

A Figura 75 mostra a conclusão da execução dos 720 Casos produzidos no lugar Str. Note-se que o último objeto a concluir a execução foi o objeto com identificador igual à 719 no tempo corrente $\tau = 1538$.

A fim de obter informações tais como o identificador do(s) objeto(s) responsável(is) pelo disparo de uma transição t, o momento e o tipo deste disparo, um monitor é associado

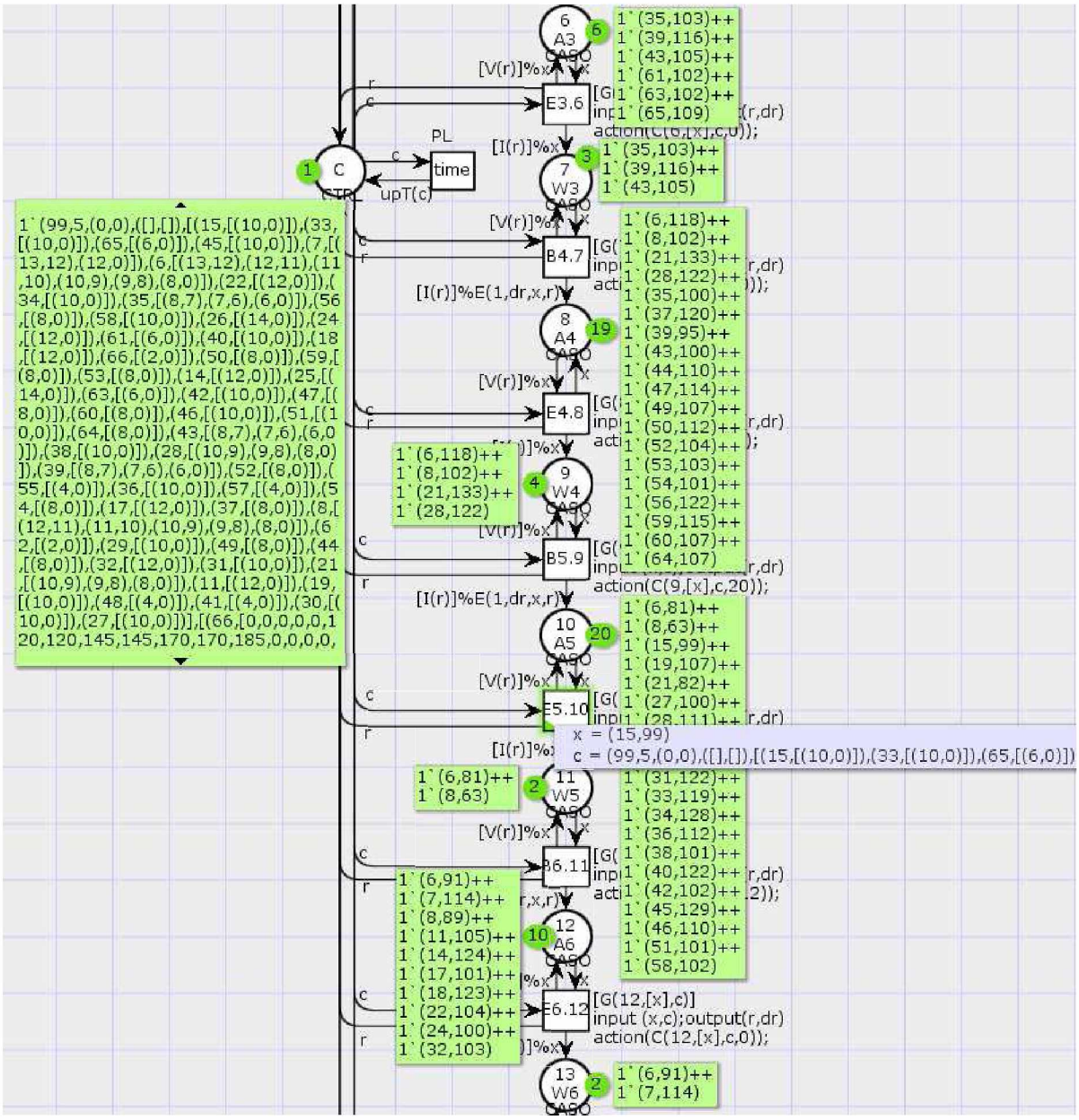


Figura 74 – Transição $E5$ apta a ser disparada de forma certa no tempo corrente $\tau = 99$ da execução do modelo mostrado na Figura 51.

a cada transição pertencente ao modelo. As informações obtidas permitem verificar a quantidade de Casos e quais transições tiveram, durante a execução do modelo, eventos perdidos e/ou atrasados.

Na Figura 76, pode-se observar que somente 35% dos Casos tiveram todos os eventos recebidos no tempo esperado. Dos 65% restantes, quase 70% tiveram no mínimo 2 a no máximo 6 eventos recebidos atrasados. Entretanto, como pode ser observado na Figura 77, mesmo com esta quantidade de eventos atrasados, 86, 4% dos Casos foram concluídos dentro do tempo médio de 82 minutos[13] (24, 5% de dispersão) ou do tempo limite de 100 minutos (5, 7% de dispersão). Dos 13, 6% restantes, 75, 8% tiveram um atraso de no

[13] O valor 82 é obtido somando os tempos médios definidos na subseção 6.1.2

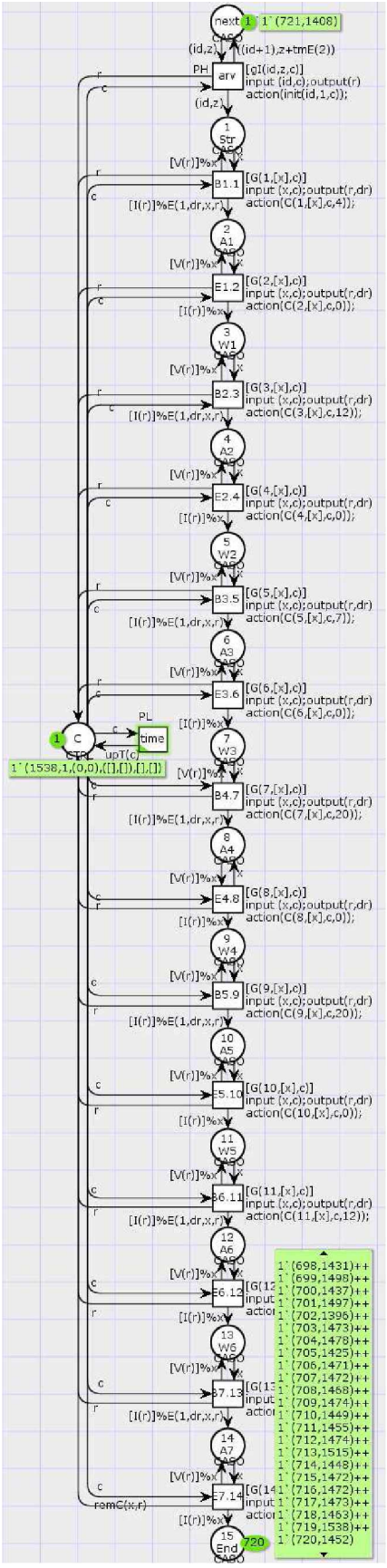


Figura 75 – Visualização do modelo no CPN Tools, mostrado na Figura 51, após a execução dos 720 Casos produzidos.

Figura 76 – Quantidade de Casos finalizados que tiveram nenhum a no máximo 12 eventos recebidos atrasados durante a execução do modelo mostrado na Figura 51.

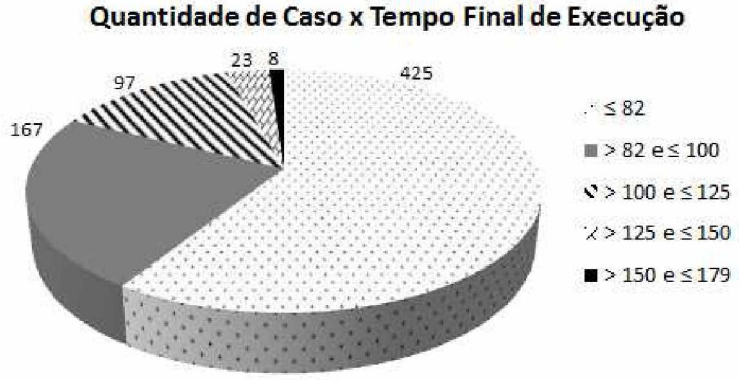


Figura 77 – Quantidade de Casos finalizados atrasados ou dentro do prazo estipulado durante a execução do modelo mostrado na Figura 51.

máximo 25 minutos (6, 4% de dispersão), ou seja, um aumento de 25% no tempo limite de execução. Além disto pode-se observar que o maior tempo de execução foi 179 minutos.

A Figura 78 mostra um gráfico para cada transição com a quantidade de Casos com eventos recebidos atrasados ou dentro do prazo limite estipulado na variável $ILPN$. Pode-se notar que a quantidade de eventos recebidos atrasados vai aumentando até o início da atividade $B4$. Após isto, a quantidade de eventos atrasados vão se reduzindo até chegar ao valor 128. Isto ocorre devido à paralelização parcial obtida através dos disparos incertos quando uma atividade se atrasa e o tempo limite para iniciar a seguinte é alcançado.

Esta paralelização parcial obtida por meio dos disparos incertos pode ser vista através das distribuições de possibilidade relacionadas aos locais $A1$, $A2$, $A3$, $A4$, $A5$, $A6$ e $A7$ para o objeto $< x >$, que representa o Caso de identificador 6, mostradas na Figura 79 (as linhas grossas representam uma possibilidade igual a 1(um) e a ausência de linhas uma possibilidade igual a 0(zero)). Note que as atividades $A5$ e $A6$, após a execução do algoritmo de recuperação, são concluídas logo em sequência pois as mesmas já foram pseudo-iniciadas anteriormente pelos seus respectivos responsáveis. Além disso, observando as distribuições de possibilidades, é fácil verificar que as atividades $A4$, $A5$ e $A6$

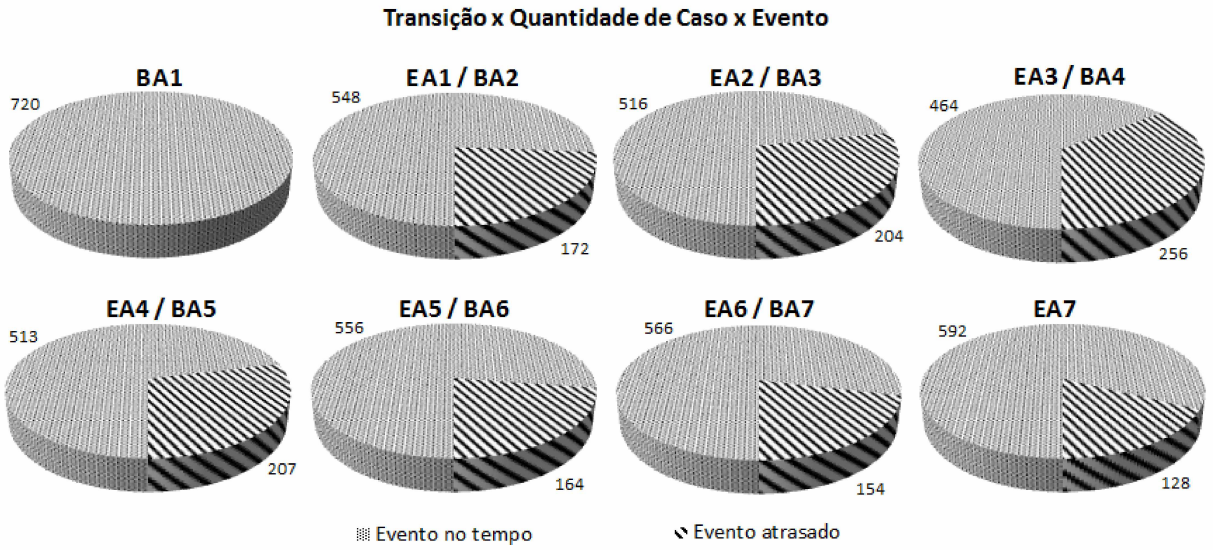


Figura 78 – Quantidade de Casos por transição que tiveram eventos atrasados ou recebidos dentro do prazo limite durante a execução do modelo mostrado na Figura 51.

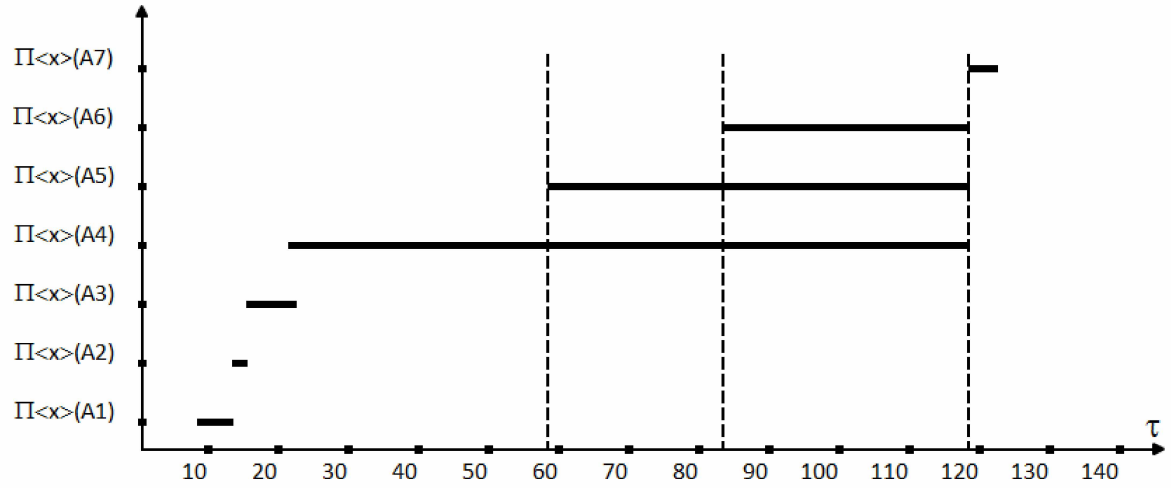


Figura 79 – Distribuições de possibilidade relacionadas aos locais $A1$, $A2$, $A3$, $A4$, $A5$, $A6$ e $A7$ para o objeto $< x >$ de identificador 6 referente à execução do modelo mostrado na Figura 51.

foram pseudo-paralelizadas durante o período de tempo [84, 118], apesar de que no modelo do processo elas estão colocadas de modo puramente sequencial.

Para avaliar os resultados obtidos nesta execução, uma nova execução é realizada onde as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados. O modelo utilizado nesta nova execução é o mesmo apresentado na Figura 51, porém com as interpretações associadas às transições podendo ser avaliadas apenas como verdadeira ou falsa, ou seja, nenhuma interpretação pode ser avaliada como incerta.

Uma vez desconsideradas as interpretações incertas e, consequentemente, os disparos incertos, o tempo de execução dos Casos podem ser concluídos com um atraso pequeno ou com um atraso relativamente excessivo. A Figura 80 mostra o estado do modelo após

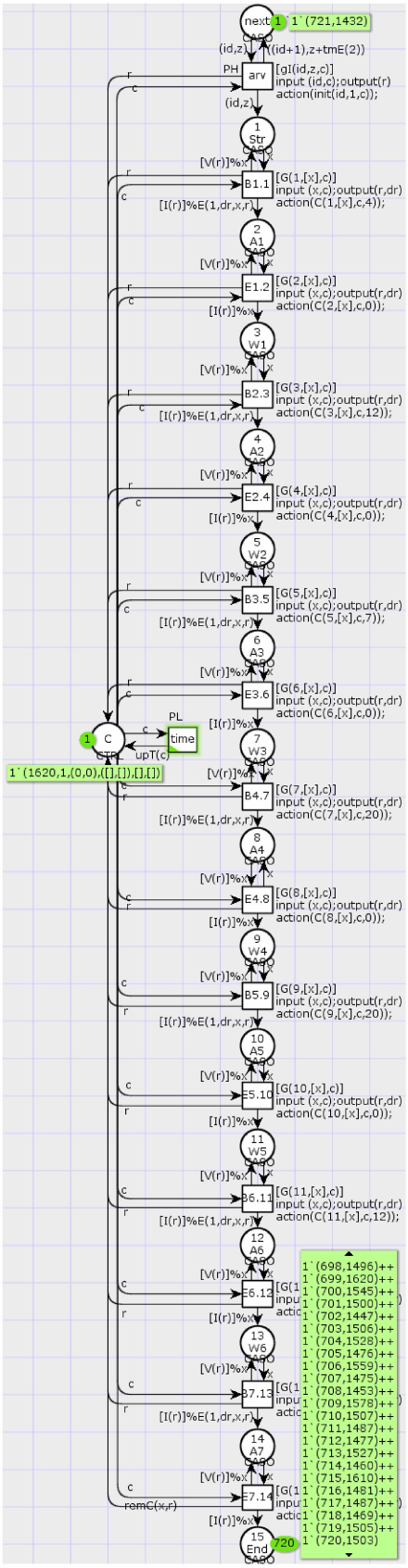


Figura 80 – Visualização do modelo no CPN Tools, mostrado na Figura 51, após a execução dos 720 Casos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

a execução dos 720 Casos produzidos no lugar *Str* quando as interpretações associadas às transições podem ser avaliadas apenas como verdadeira ou falsa. Nota-se que o tempo de execução total foi de 1620 minutos sendo o Caso de identificador 699 o último a concluir a sua execução.

Ao comparar a Figura 81 com a Figura 76, pode-se perceber que a quantidade de Casos que tiveram todos os eventos recebidos no tempo esperado é ligeiramente menor. Entretanto, ao comparar as Figuras 77 e 82, verifica-se que 81 Casos a menos foram concluídos dentro do tempo médio de 82 minutos (26, 2% de dispersão) ou do tempo limite de 100 minutos (5, 7% de dispersão) quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados. Além disto pode-se observar que 39 Casos tiveram um acréscimo de no mínimo 50% no tempo de execução (17, 2% de dispersão) sendo que 16 destes Casos tiveram um acréscimo entre 79% a 194% (16% de dispersão). Estas comparações são sintetizadas na Tabela 8.

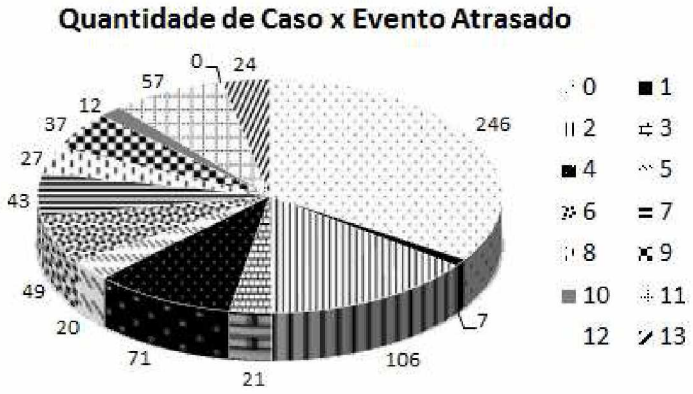


Figura 81 – Quantidade de Casos finalizados que tiveram nenhum a no máximo 13 eventos recebidos atrasados durante a execução do modelo mostrado na Figura 51 quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

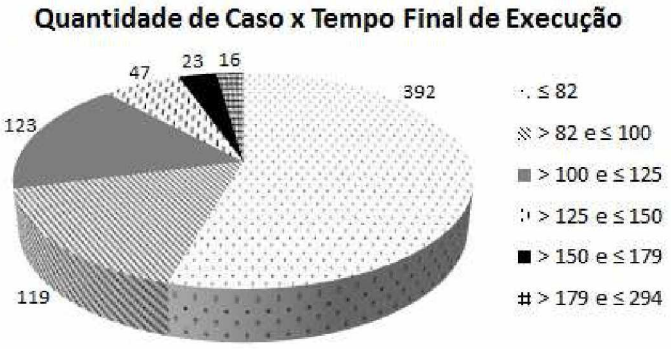


Figura 82 – Quantidade de Casos finalizados atrasados ou dentro do prazo estipulado durante a execução do modelo mostrado na Figura 51 quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

Tabela 8 – Comparação entre os resultados obtidos através das duas execuções realizadas na WF-net Possibilística mostrada na Figura 51 considerando primeiramente a noção de disparo incerto e posteriormente desconsiderando-a

		Com a noção de disparo incerto	Sem a noção de disparo incerto
Quantidade	Casos finalizados com todos eventos dentro do prazo	252	246
	Casos finalizados dentro do tempo limite de 100 minutos	592	511
	Casos finalizados com um acréscimo de no mínimo 50% no tempo limite	8	39
	Casos finalizados com um acréscimo entre 79% e 194% no tempo limite	0	16

Levando em consideração o momento em que os eventos foram recebidos, a Figura 83 detalha, para cada transição, a quantidade de Casos que tiveram o evento relacionado à transição em questão recebido dentro dentro do prazo estipulado ou atrasado. Pode-se constatar que por mais que os eventos recebidos atrasados foram diminuindo após a conclusão da atividade $A4$, esta redução foi mínima e pode ser atribuída à um tempo de duração associada à atividade em questão relativamente baixo.

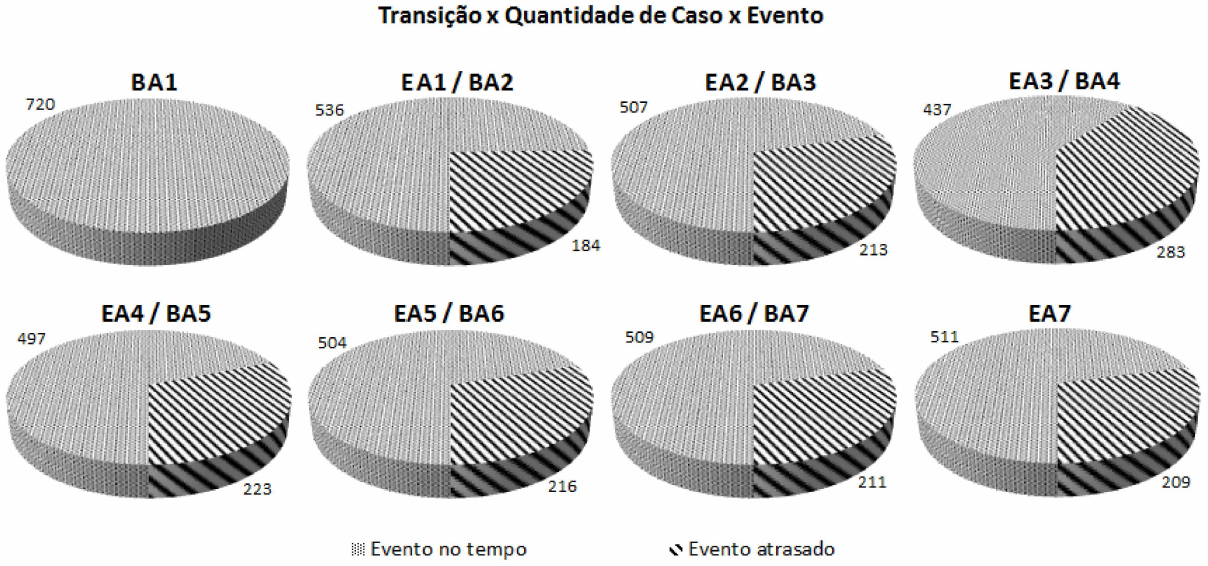


Figura 83 – Quantidade de Casos por transição que tiveram eventos atrasados ou recebidos dentro do prazo limite durante a execução do modelo mostrado na Figura 51.

Com o propósito de mostrar a diferença do tempo de processamento gasto quando o modelo do processo é executado de modo puramente sequencial, as distribuições de possibilidade relacionadas aos locais $A1$, $A2$, $A3$, $A4$, $A5$, $A6$ e $A7$ para o objeto $< x >$ que representa o Caso de identificador 6 são mostradas na Figura 84 (as linhas grossas representam uma possibilidade igual a 1(um) e a ausência de linhas uma possibilidade igual a

0(zero)). Ao comparar a Figura 79 com a 84, percebe-se que o tempo de processamento gasto ao se utilizar os disparos incertos é 16% menor. Este valor pode ser relativamente pequeno, entretanto em outros casos o tempo de processamento foi inferior à 50%.

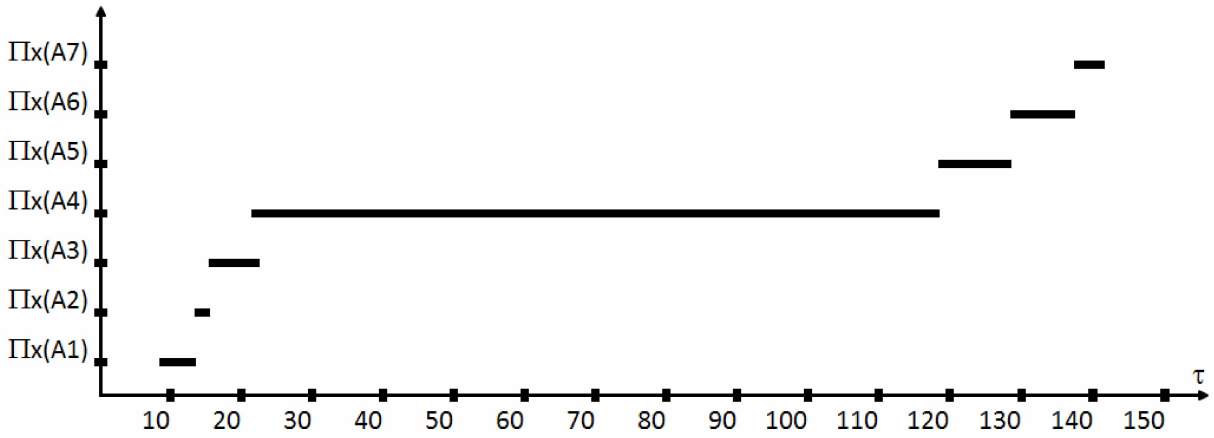


Figura 84 – Distribuições de possibilidade relacionadas aos locais $A1$, $A2$, $A3$, $A4$, $A5$, $A6$ e $A7$ para o objeto $< x >$ de identificador 6 referente à execução do modelo mostrado na Figura 51.

6.3.3 Resultado da Simulação de um processo modelado em WF-net Possibilística que aceita Desvios de forma a permitir o Cancelamento de Processos

A Figura 85 mostra a inserção do lugar *next* e da transição *arv* no modelo apresentado na Figura 52 de um processo de solicitação de cartão de crédito. A declaração das variáveis *id* e *z* é idêntica à descrita na subseção 6.3.1. A declaração da função *cIL* é descrita a seguir:

```
fun cIL(tm) = let val cnl = round(uniform(0.0, 1.0))
                  val tmp = tmE(80)
              in [cnl, tm + tmp]
              end
```

A função *cIL* determina se um pedido de cancelamento será solicitado e, de forma exponencial, o momento desta solicitação para o Caso em questão. O primeiro elemento da lista retornada indica se o pedido de cancelamento foi solicitado (1) ou não (0). O segundo elemento é a soma do tempo corrente ao tempo obtido através da função *tmE*. Observa-se que foi considerado um tempo médio de 80 minutos para determinar a chegada do pedido de cancelamento. Este tempo foi estipulado considerando a soma dos tempos médios de execução das atividades pertencentes à região de cancelamento definido neste modelo.

O modelo descrito na Figura 52 acrescido do lugar *next* e da transição *arv*, tal como mostrado na Figura 85, é executado e a Figura 86 mostra, no tempo corrente $\tau = 162$,

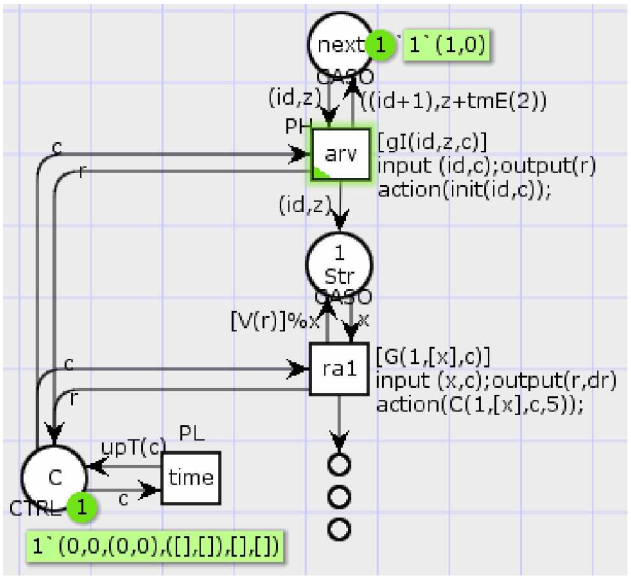


Figura 85 – Lugar *next* e transição *arv* responsáveis por produzir um novo Caso no modelo apresentado na Figura 52.

uma replicação registrada no CPN Tools do estado do modelo apresentado na Figura 52. Considerando este estado em particular, 7 observações podem ser realizadas. São elas:

1. um novo Caso de $id = 92$ será produzido quando o tempo corrente for igual ao valor 164;

2. 49 Casos estão em execução[14];

3. os Casos em execução estão localizados nos seguintes lugares: 35 objetos em ON, 8 em $W1$, 9 em WT, 7 em $W2$, 5 em $W4$, 6 em $W5$, 10 em $W6$, 2 em $W9$, 1 em $W10$ e 1 em $W12$;

4. dos 49 Casos em execução, 35 Casos ainda podem ter um pedido de cancelamento solicitado dado que se encontram dentro da região de cancelamento formada pelos lugares $W1$, $W2$, $W3$, $W4$, $W5$ e WT;

5. qualquer pedido de cancelamento solicitado para um dos 14 Casos presentes nos lugares $W6$, $W9$, $W10$ e $W12$ será considerado inválido dado que os Casos se encontram fora da região de cancelamento determinada para este modelo;

6. a execução de 44 Casos foi concluída, ou seja, o lugar *End* contém 44 Casos;

7. o Caso com identificador 90 teve um pedido de cancelamento solicitado, ou seja, a execução deste Caso deve ser cancelada. Para isto a transição *pcr* deve ser disparada, entretanto, como pode ser observado, o Caso de $id = 90$ se localiza em $W1$ impossibilitando assim o disparo de *pcr*. Assim sendo, como a transição cfc está

[14] Objetos em lugares diferentes com mesmo identificador se referem a um mesmo Caso em execução.

apta a ser disparada, tal como destacado na Figura 86, neste momento de forma incerta, uma sequência de disparos a partir de cfc à pcr é obtida para, posteriormente,

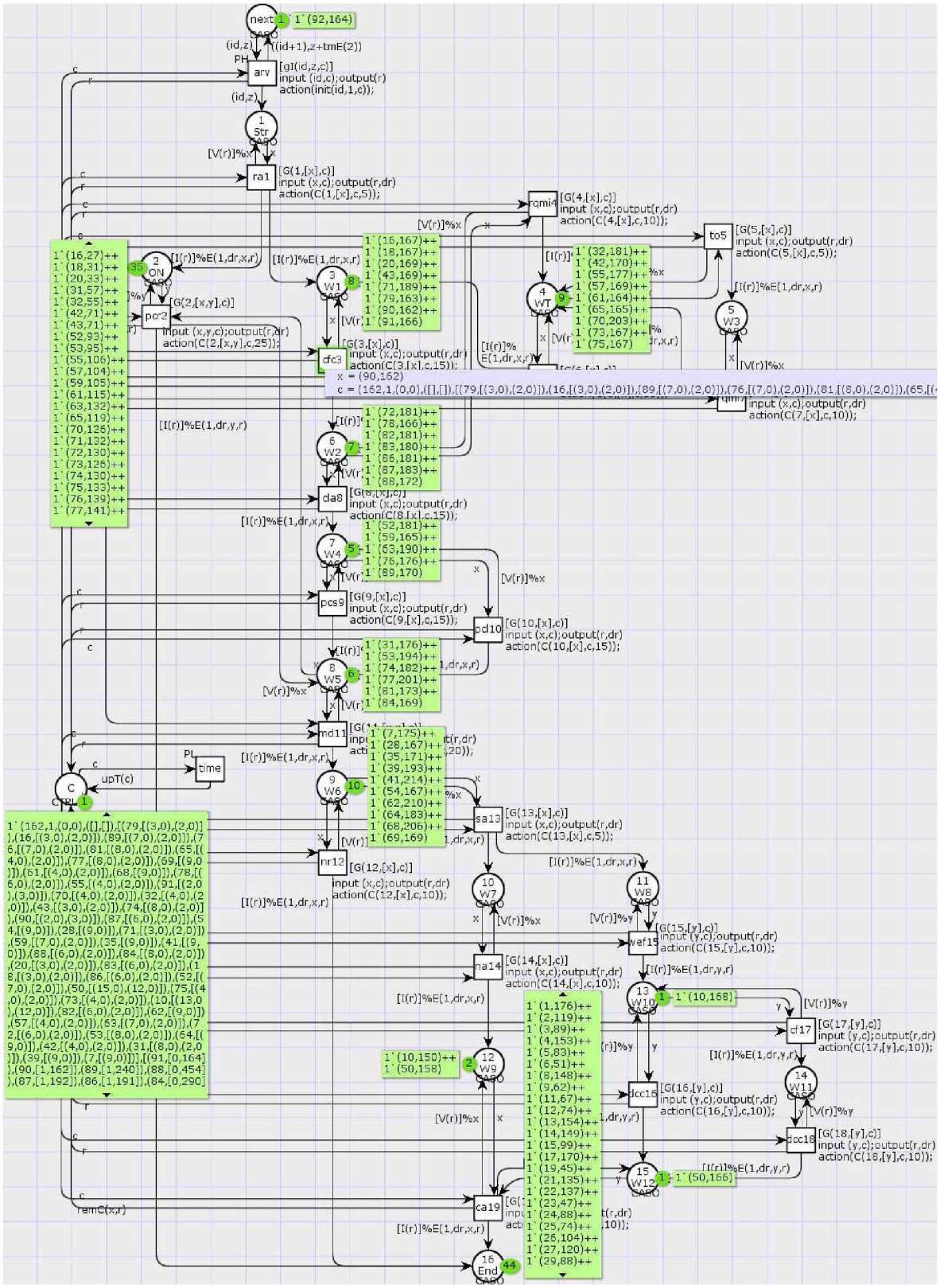


Figura 86 – Visualização da execução do modelo no CPN Tools, mostrado na Figura 52, no tempo corrente $\tau = 162$.

cancelar a execução deste caso disparando a transição *pcr*.

A Figura 87 mostra a transição *pcr* apta a ser disparada após a sequência de disparo incerta realizada a partir de cfc. Levando em consideração que a transição *pcr* está habilitada por uma marcação imprecisa, dado que o Caso de $id = 90$ se encontra nos lugares $W1$, $W2$, $W3$, $W4$, $W5$ e WT, o algoritmo de defuzzificação é, então, chamado para posteriormente dispará-la de forma certa e cancelar a execução do Caso.

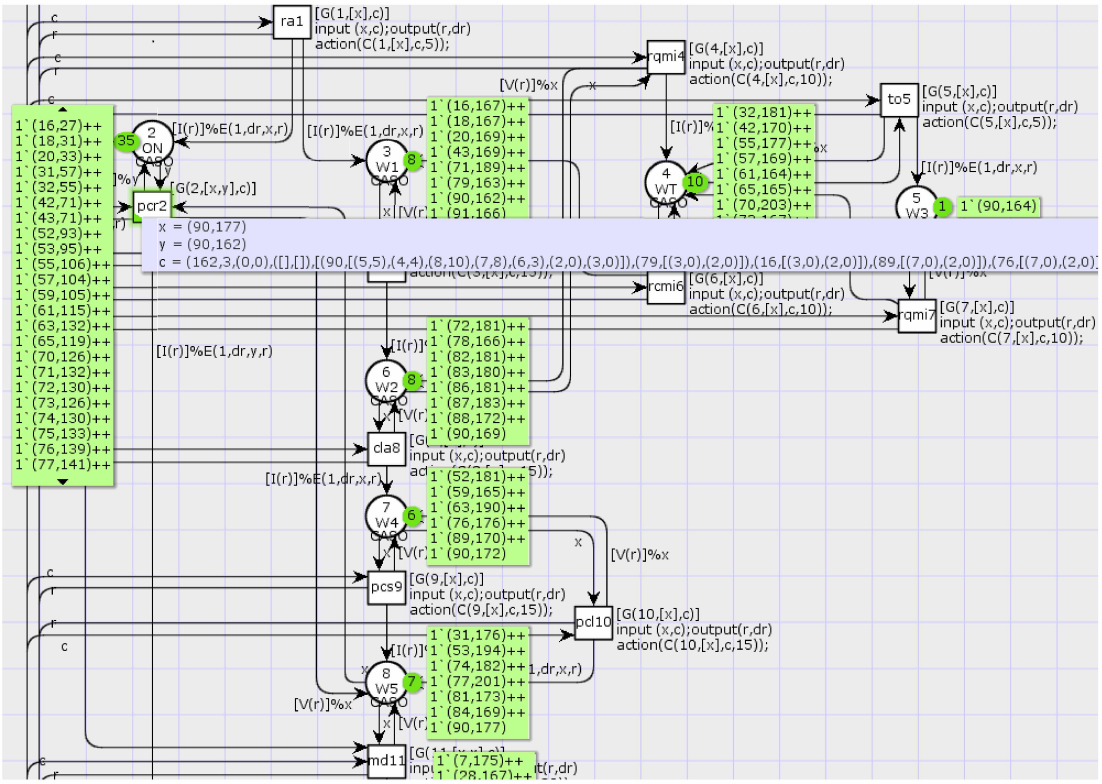


Figura 87 – Transição *pcr* habilitada por uma marcação imprecisa no tempo corrente $\tau = 162$ da execução do modelo mostrado na Figura 52.

A Figura 88 mostra, através da marcação do lugar de controle exibida na variável c, no tempo corrente 162, que o objeto de identificador 90 está em processo de defuzzificação por causa da transição de identificador 2. Além disto, os disparos incertos realizados nas transições cfc, *cla* e *pcl* representadas, respectivamente, pelos identificadores 3, 8 e 10, serão finalizados e, os realizados nas transições *to* e *rqmi* serão cancelados através do disparo das transições *rqmi* e *rcmi* representadas, respectivamente, pelos identificadores 7 e 6.

Ao observar a Figura 89, pode-se notar que o procedimento de defuzzificação foi finalizado e a transição *pcr*, além de estar apta a ser disparada, está habilitada de forma precisa. Deste modo, a mesma será disparada de forma certa cancelando, assim, a execução do Caso de $id = 90$.

A Figura 90 mostra a conclusão da execução dos 720 objetos produzidos no lugar *Str*. Note-se que todos os Casos produzidos tiveram sua execução concluída independentemente

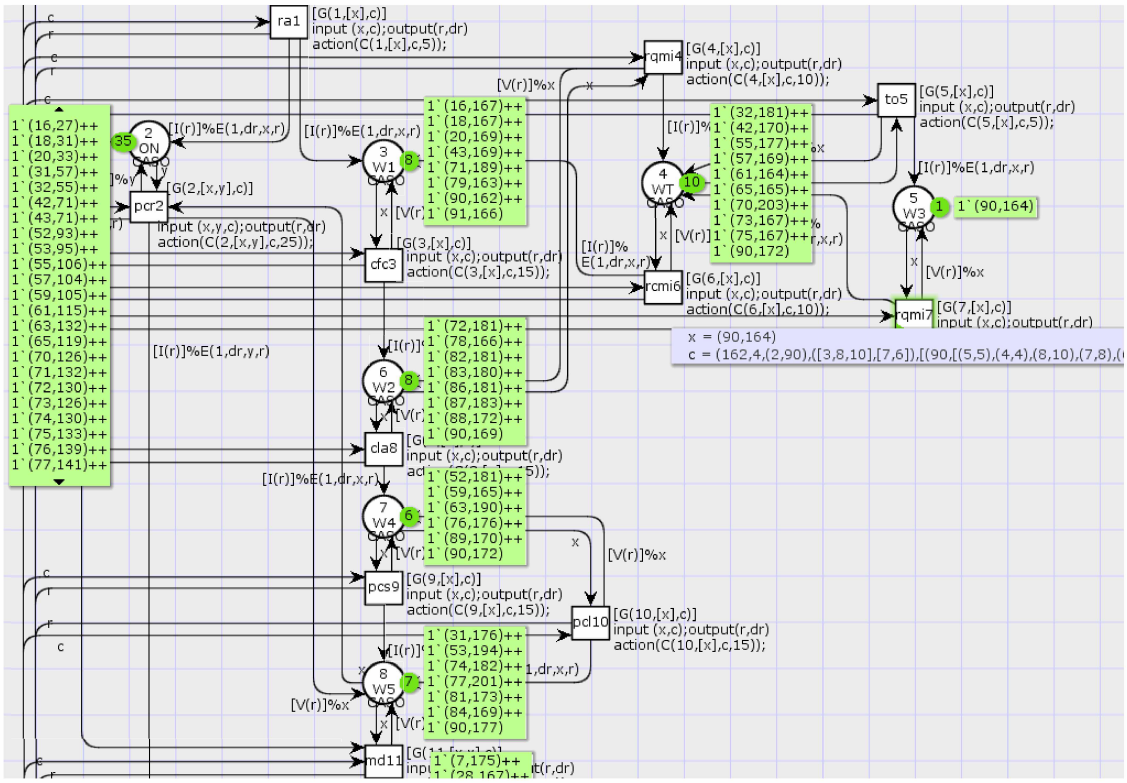


Figura 88 – Transição *to* apta a ter o disparo incerto cancelado através do disparo da transição *rqmi*7 no tempo corrente $\tau = 162$ da execução do modelo mostrado na Figura 52.

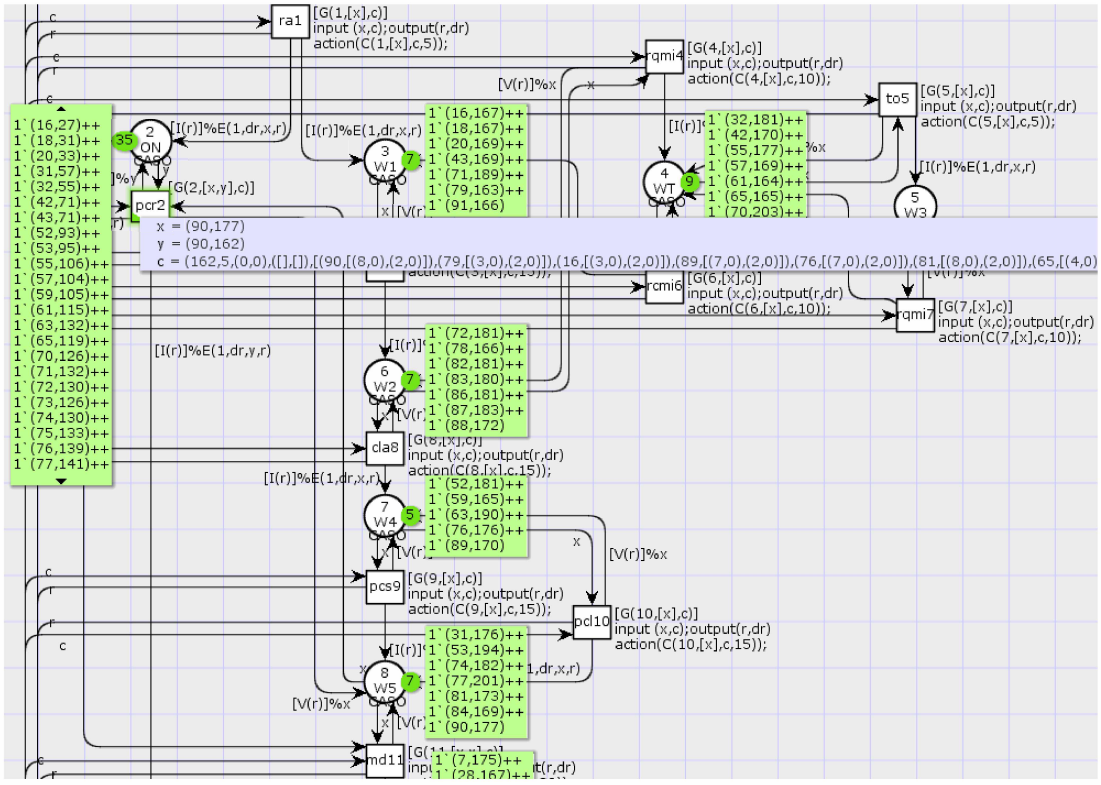


Figura 89 – Transição *pcr* apta a ser disparada de forma certa no tempo corrente $\tau = 162$ da execução do modelo mostrado na Figura 52.

se um pedido de cancelamento foi ou não solicitado.

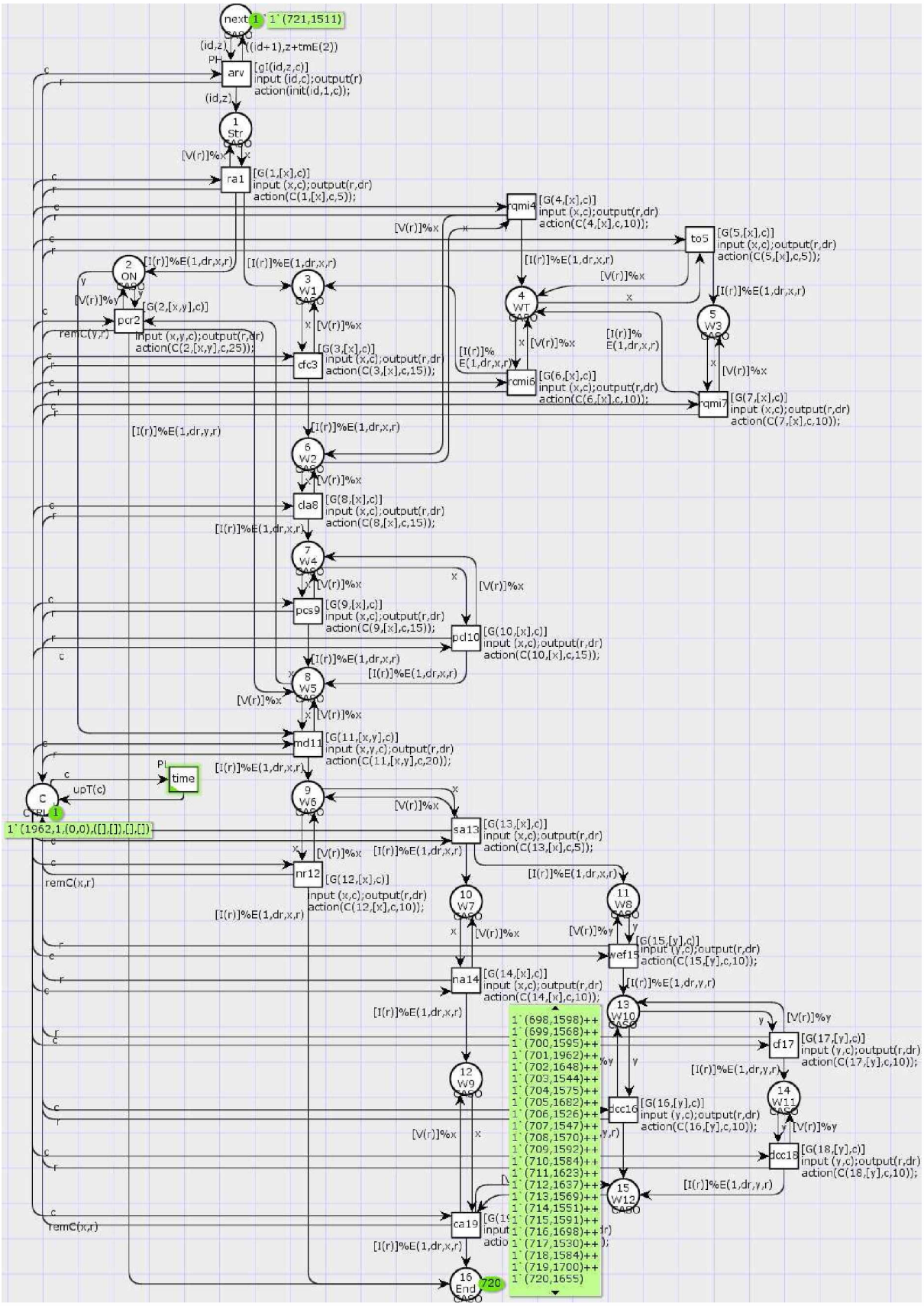


Figura 90 – Visualização do modelo no CPN Tools, mostrado na Figura 52, após a execução dos 720 objetos produzidos.

A fim de obter informações tais como o pedido de cancelamento para cada Caso produzido bem como o identificador do(s) objeto(s) responsável(is) pelo disparo de uma transição t, o momento e o tipo deste disparo, um monitor é associado a cada transição pertencente ao modelo. As informações obtidas permitem verificar a quantidade de Casos bem como a localização do Caso no modelo quando, durante a execução, um pedido de cancelamento é solicitado, seja ele dentro ou fora da região de cancelamento.

A Figura 91 mostra a quantidade de Casos finalizados que tiveram ou não um pedido de cancelamento solicitado. Dos 720 Casos produzidos, 370 não tiveram nenhum pedido de cancelamento solicitado. Dos 350 restantes, 154 Casos tiveram um pedido de cancelamento solicitado quando a execução dos mesmos se encontrava fora da região de cancelamento, ou seja, nenhum destes pedidos foram aceitos. Por fim, 196 Casos tiveram a sua execução cancelada após o recebimento do pedido de cancelamento.

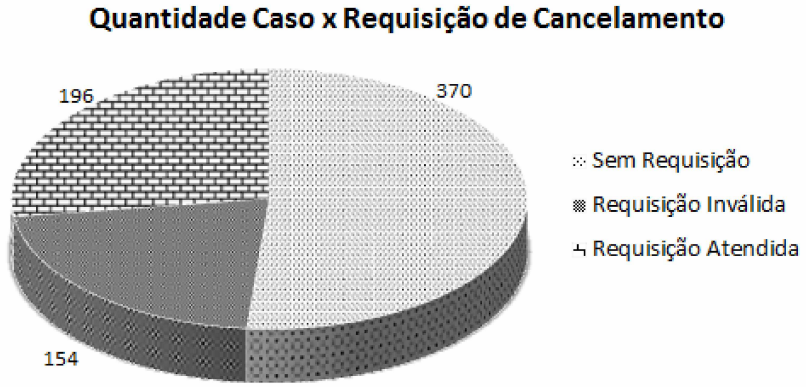


Figura 91 – Quantidade de Casos finalizados que tiveram ou não um pedido de cancelamento solicitado durante a execução do modelo mostrado na Figura 52.

Tendo em vista os 196 Casos cancelados, a Figura 92 mostra a quantidade de Casos que tiveram um pedido de cancelamento solicitado em um determinado lugar dentro da

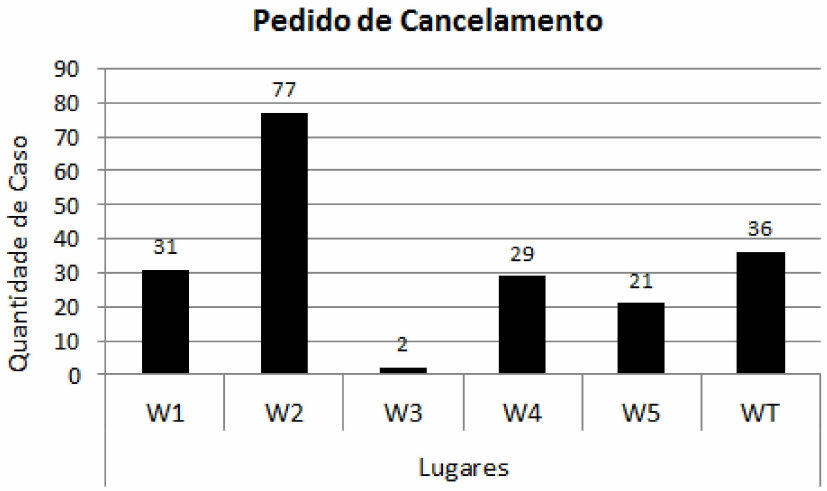


Figura 92 – Quantidade de Casos que tiveram um pedido de cancelamento solicitado em um determinado lugar durante a execução do modelo mostrado na Figura 52.

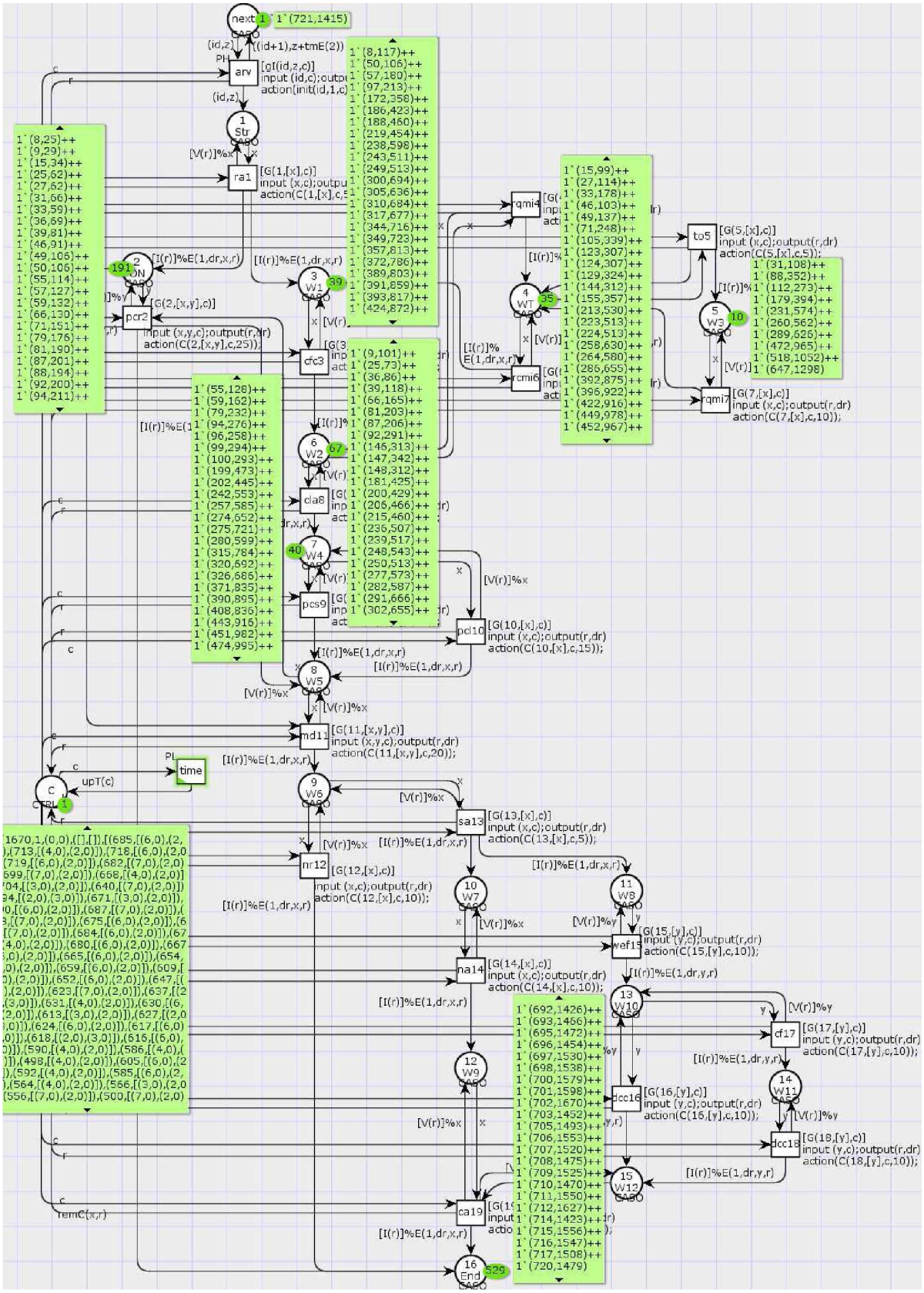


Figura 93 – Visualização do modelo no CPN Tools, mostrado na Figura 52, após a execução dos 720 objetos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

região de cancelamento. Observa-se que os disparos incertos não foram necessários para os 21 Casos que tiveram o pedido de cancelamento solicitado quando o mesmo se localizava em $W5$. Isto ocorre porque os Casos localizados em $W5$ habilitam tanto a transição *pcr* quanto a transição *md*. Do contrário, ou seja, para os 175 Casos restantes, os disparos incertos foram necessários para se obter uma sequência de disparo válida que fosse capaz de habilitar a transição *pcr* e, por conseguinte, cancelar a execução dos mesmos.

Para avaliar os resultados obtidos nesta execução, uma nova execução é realizada onde interpretações incertas e, consequentemente, os disparos incertos são desconsiderados. O modelo utilizado nesta nova execução é o mesmo apresentado na Figura 52, porém com as interpretações associadas às transições podendo ser avaliadas apenas como verdadeira ou falsa, ou seja, nenhuma interpretação pode ser avaliada como incerta.

Uma vez desconsideradas as interpretações incertas e, consequentemente, os disparos incertos, alguns Casos podem não ser concluídos devido à localização do Caso no momento da solicitação. A Figura 93 mostra o estado do modelo após a execução dos 720 Casos produzidos no lugar *Str* quando as interpretações associadas às transições podem ser avaliadas apenas como verdadeira ou falsa. Nota-se que 191 Casos $(26,5\%)$ não concluíram sua execução devido a sua localização. Uma vez que o pedido de cancelamento é solicitado, a execução do Caso em questão é suspensa ficando num estado de espera permanente.

A Figura 94 mostra que 355 Casos foram finalizados sem que um pedido de cancelamento fosse solicitado. Dos 365 restantes, 143 tiveram um pedido de cancelamento solicitado quando a execução dos mesmos se encontrava fora da região de cancelamento, ou seja, nenhum destes pedidos foram aceitos. Por fim, observe-se que, mesmo se localizando dentro da região de cancelamento no momento do pedido, 191 Casos tiveram um pedido de cancelamento solicitado, porém não atendido devido a impossibilidade da transição *pcr* se tornar habilitada. Em contrapartida, 31 Casos tiveram a sua execução cancelada após o recebimento do pedido de cancelamento dado que os mesmos se encontravam no

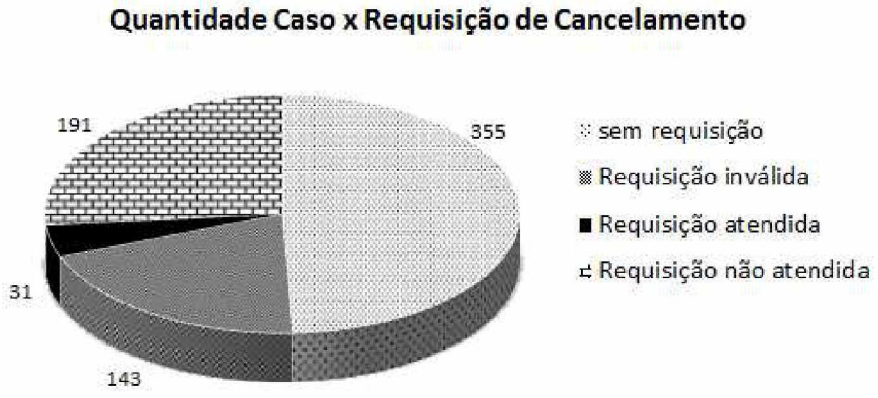


Figura 94 – Quantidade de Casos finalizados que tiveram ou não um pedido de cancelamento solicitado durante a execução do modelo mostrado na Figura 52 quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

lugar $W5$ permitindo habilitar a transição pcr.

Quando se desconsidera as interpretações incertas e, consequentemente, os disparos incertos, um Caso é cancelado se, e somente se, o mesmo se encontra no lugar $W5$, o qual é um lugar de entrada da transição pcr. A fim de verificar este fato, as Figuras 95 e 96 discriminam a localização dos Casos quando os mesmos receberam um pedido de cancelamento. Através da Figura 96, pode-se observar que todos os Casos cancelados se localizavam exclusivamente no lugar $W5$. Por outro lado, todos os Casos que tiveram um pedido de cancelamento dentro da região de cancelamento e não foram atendidos se localizavam nos lugares $W1$, $W2$, $W3$, $W4$ e WT.

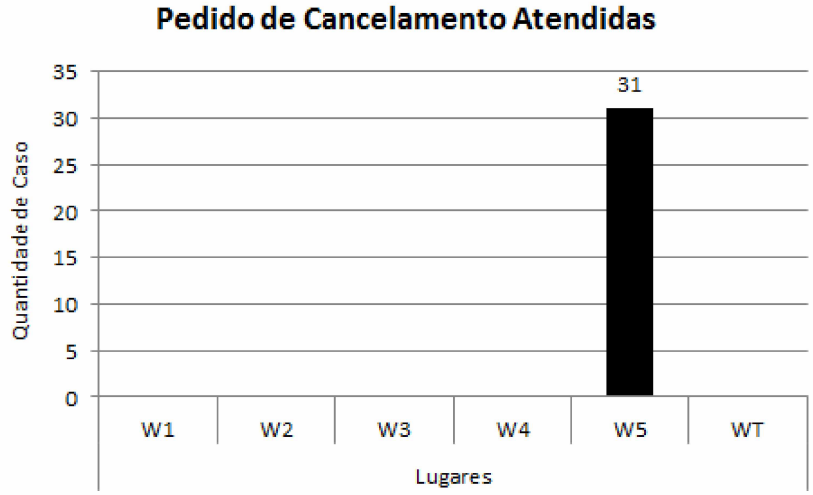


Figura 95 – Quantidade de Casos que tiveram um pedido de cancelamento solicitado e atendido em um determinado lugar durante a execução do modelo mostrado na Figura 52.

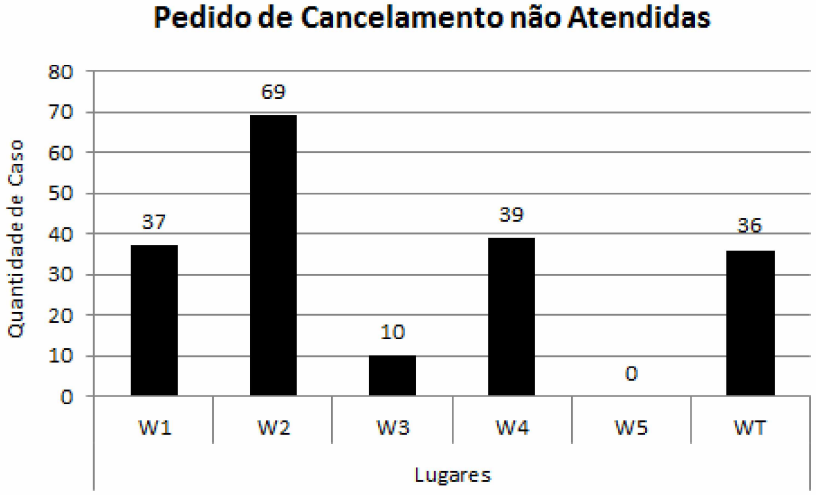


Figura 96 – Quantidade de Casos que tiveram um pedido de cancelamento solicitado mas não atendido em um determinado lugar durante a execução do modelo mostrado na Figura 52.

6.3.4 Resultado da Simulação de um Processo modelado em IOWF-net Possibilística que aceita Desvios em casos de Falhas de Comunicação entre Processos

O lugar *next* e a transição *arv*, responsáveis por gerar os novos Casos para o processo que precede a apresentação de um artigo em uma conferência (modelos nas Figuras 53, 54 e 55), são mostrados na Figura 97. Observa-se que um objeto composto pelo identificador do caso produzido (id) e pelo valor negativo -1 é produzido em cada um dos lugares de comunicação. Isto permite indicar que o evento relacionado ao lugar de comunicação se encontra falso para o objeto com identificador igual à id. Desta forma, a fim de otimizar a execução do modelo, ao fim da execução de um Caso determinado por id, todos os objetos com este identificador serão removidos dos lugares de comunicação através das sete transições (df, ad, rj, ac, tl, fv e af) adicionadas ao modelo implementado.

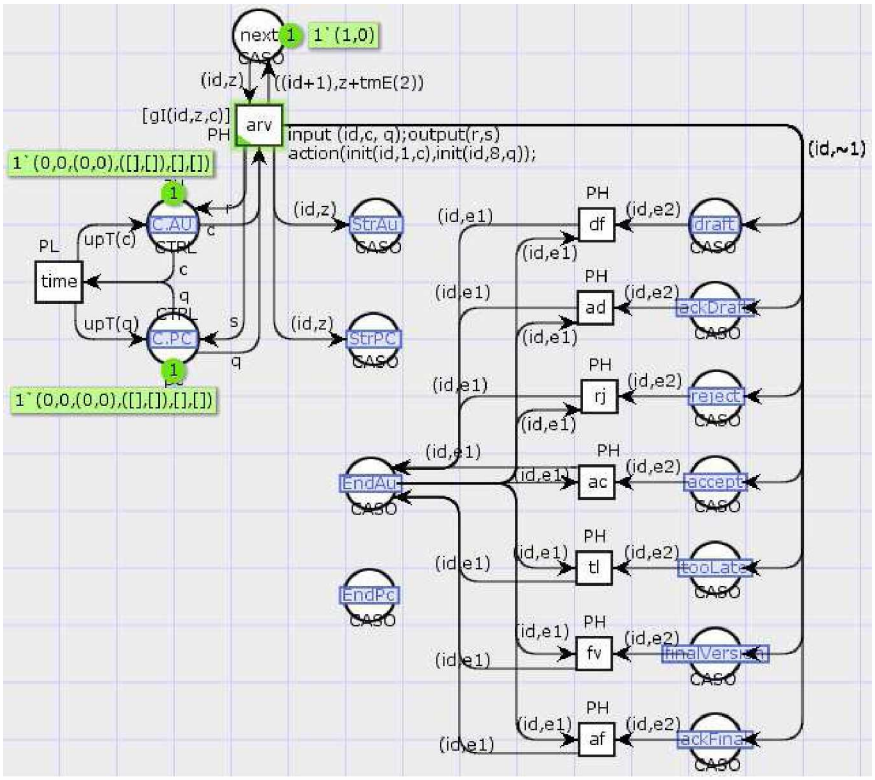


Figura 97 – Lugar *next* e transição *arv* em conjunto com as outras sete transições adicionais no modelo apresentado na Figura 55.

Nesta execução, além das variáveis id e z já declaradas na subseção 6.3.1, mais duas variáveis usadas nos arcos de entrada e/ou de saída das sete transições adicionadas ao modelo são declaradas:

```
var e1, e2: INT;
```

A função cIL, usada na função *init* já declarada, se comporta de forma idêntica à anteriormente definida na subseção 6.3.1, porém com valores diferentes. A declaração

dela é dada a seguir:

$$\begin{aligned} \mathrm{fun\ cIL(tm)} = [&0,\ 25{+}\mathrm{tm},\ 60{+}\mathrm{tm},\ 300{+}\mathrm{tm},\ 61{+}\mathrm{tm},\ 0,\ 0,\ 0\ \ \ ,\ 0,\\ &0,\ 0\ \ \ ,\ 0\ \ \ ,\ 0\ \ \ ,\ 0\ \ \ ,\ 0,\ 0,\ 95{+}\mathrm{tm},\ 0] \end{aligned}$$

Por fim, os modelos nas Figuras 53, 54 e 55, acrescido do lugar *next* e das oito transições (*arv*, *df*, *ad*, *rj*, *ac*, *tl*, *fv* e *af*), tal como mostrado na Figura 97, é executado e as Figuras 98 e 99 mostram, respectivamente, no tempo corrente $\tau = 60$, uma replicação registrada no CPN Tools do estado dos modelos dos processos *AU* e *PC*. Considerando este estado em particular e lembrando que os lugares de comunicação representam a chegada ou envio de um evento entre os modelos dos processos, 6 observações podem ser realizadas. São elas:

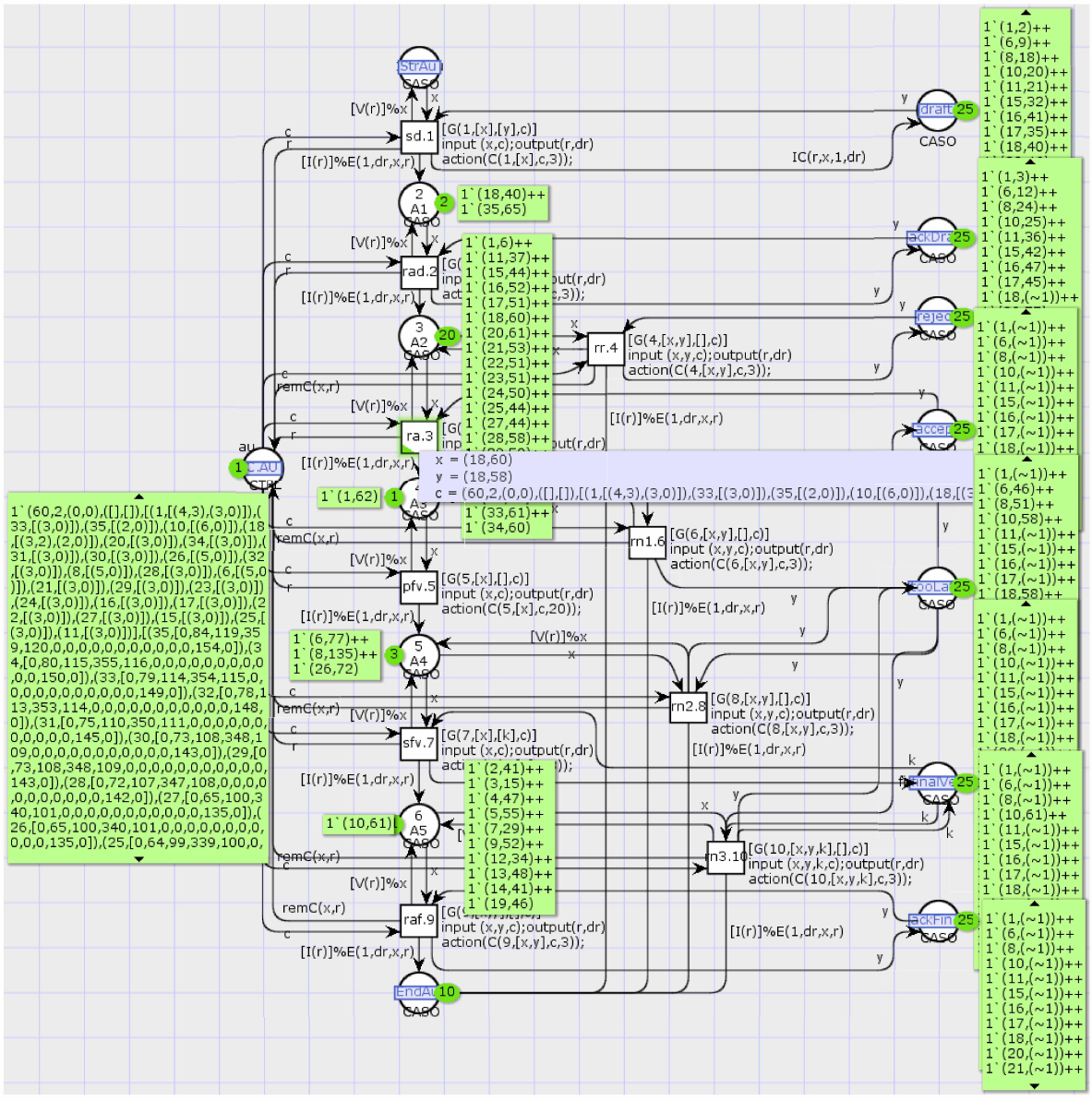


Figura 98 – Visualização da execução do modelo do processo *AU* no CPN Tools, mostrado na Figura 53, no tempo corrente $\tau = 60$

1. 25 Casos estão em execução no modelo do processo AU e 24 em PC [15];

2. os Casos em execução no modelo do processo AU estão localizados nos seguintes lugares: 2 objetos em $A1$, 20 em $A2$, 1 em $A3$, 3 em $A4$ e 1 em $A5$;

3. os Casos em execução no modelo do processo PC estão localizados nos seguintes lugares: 1 em $StrPC$, 15 em $P3$ e 8 em $P4$;

4. a execução de 10 Casos foram concluídas no modelo do processo AU e 11 no PC, ou seja, o lugar $EndAU$ contém 10 e o $EndPC$ 11 Casos;

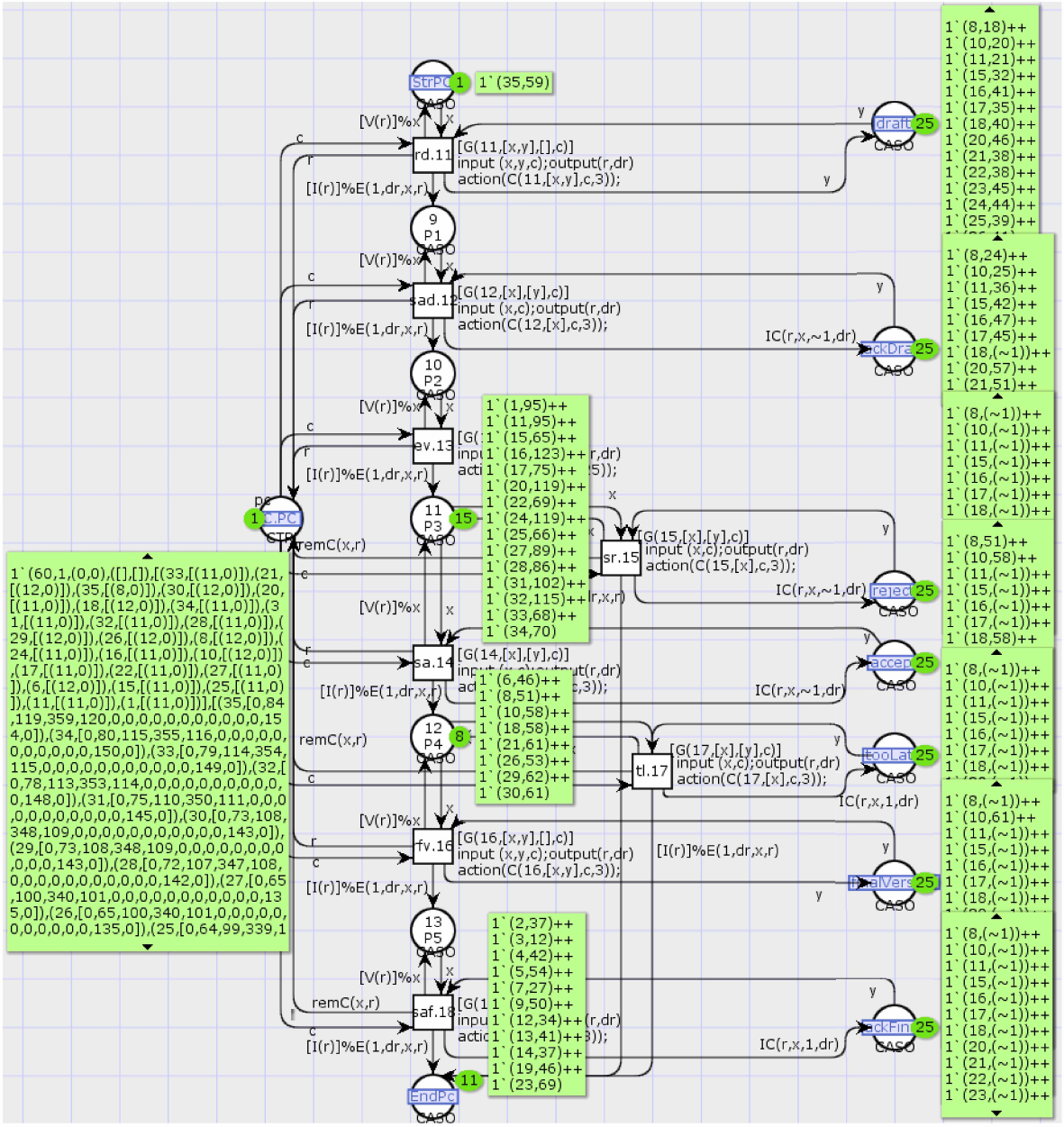


Figura 99 – Visualização da execução do modelo do processo PC no CPN Tools, mostrado na Figura 53, no tempo corrente $\tau = 60$.

[15] Objetos em lugares diferentes com mesmo identificador se referem a um mesmo Caso em execução.

5. o Caso com identificador 1 se localiza no modelo do processo AU tanto no lugar $A2$ quanto no lugar $A3$. Isto indica uma marcação imprecisa para este Caso, a qual é obtida através do disparo incerto realizado na transição ra. Tal disparo foi efetuado considerando que no tempo limite definido para este Caso nesta transição, através da função cIL, o evento relacionado a ra ainda é tido como falso (nos lugares de comunicação *accept* e $reject$ têm-se o seguinte objeto $(1, -1)$). Esta situação ocorre devido à falta de confirmação sobre o aceite ou a recusa do esboço do artigo;

6. o Caso com identificador 18 se localiza no modelo do processo AU nos lugares $A1$ e $A2$ de forma imprecisa devido ao disparo incerto da transição rad. Tal disparo é realizado em consequência da perda do evento referente à confirmação do recebimento do esboço do artigo, o que pode ser observado através do objeto $(18, -1)$ no lugar de comunicação $ackDraft$. Levando em consideração que o evento relacionado à confirmação sobre o aceite do esboço do artigo foi recebido (objeto $(18, 58)$ no lugar de comunicação *accept*) e que a transição ra está habilitada, então um disparo pode ser realizado em ra este Caso tal como destacado na figura;

Dado que a transição ra está habilitada por uma marcação imprecisa e sua função de autorização é avaliada como verdadeira, o algoritmo de defuzzificação é então chamado para posteriormente dispará-la de forma certa. A Figura 100 mostra, através da marcação do lugar de controle exibida na variável c, no tempo corrente 60, que o objeto de identificador 18 está em processo de defuzzificação por causa da transição de identificador 3.

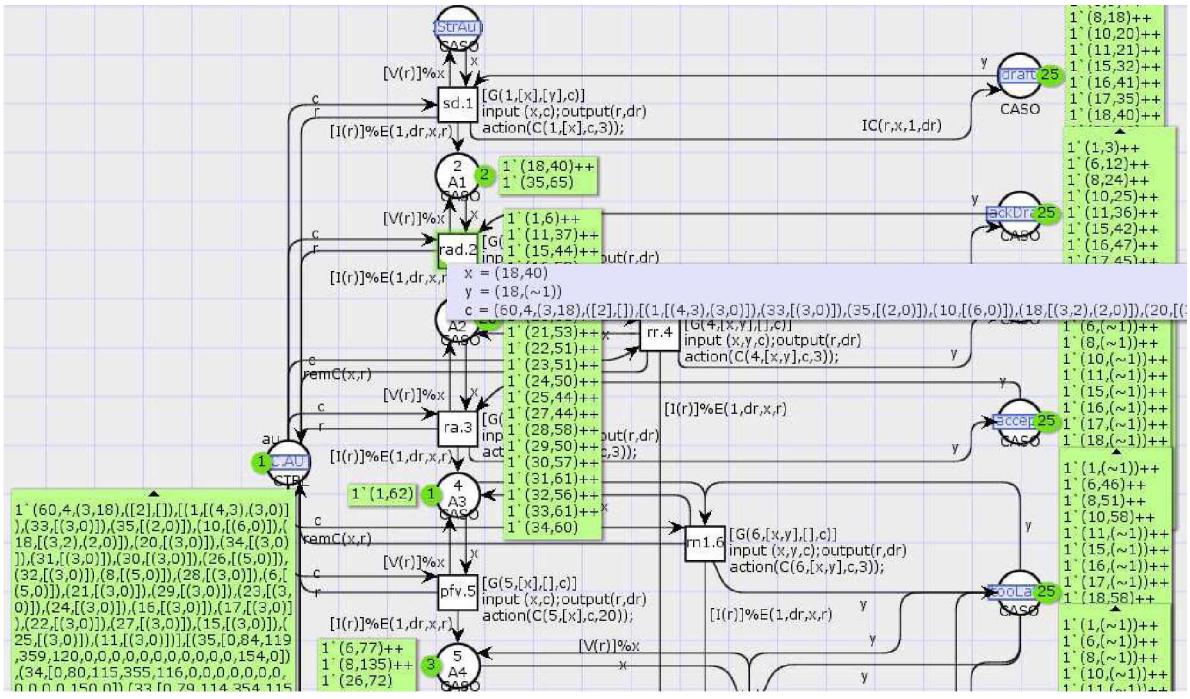


Figura 100 – Transição ra apta a ter o disparo incerto finalizado no tempo corrente $\tau = 60$ da execução do modelo do processo AU no CPN Tools, mostrado na Figura 53.

Além disto, o disparo incerto realizado na transição rad, representada pelo identificador 2 e destacada na figura, será finalizado e não existe qualquer outro disparo incerto a ser cancelado.

Ao observar a Figura 101, pode-se notar que o procedimento de defuzzificação foi finalizado e a transição ra, além de estar apta a ser disparada, está habilitada de forma precisa. Deste modo, a mesma será disparada de forma certa.

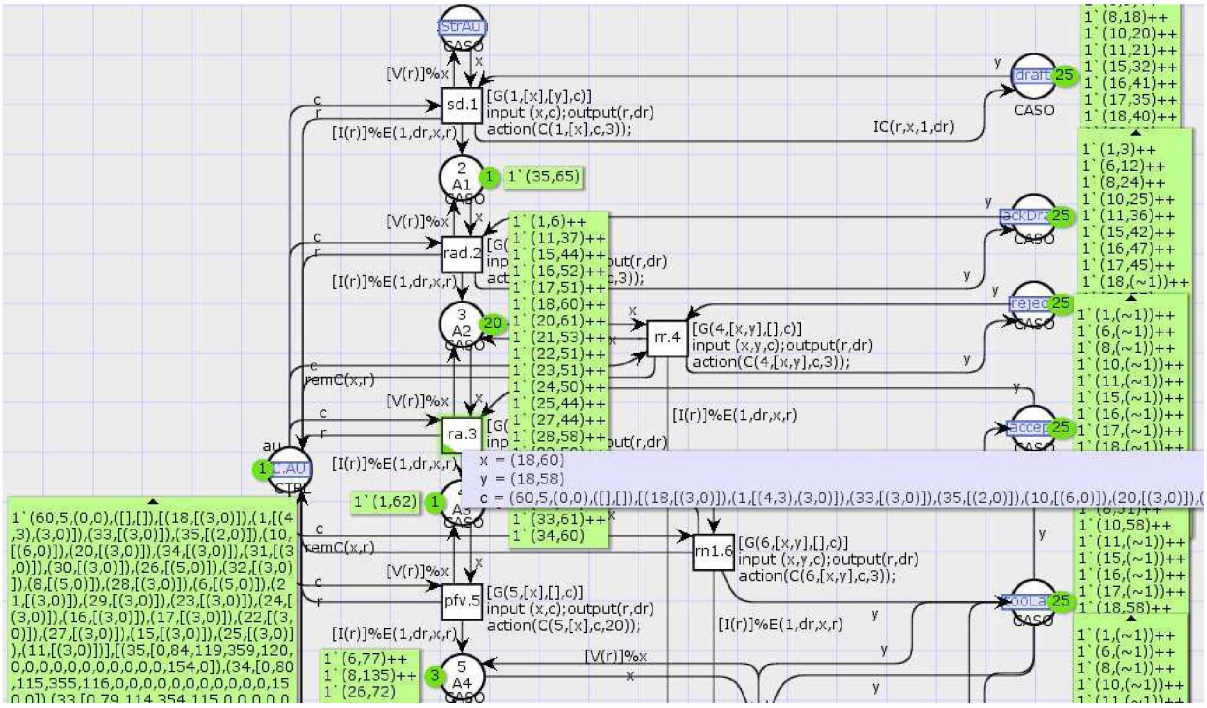


Figura 101 – Transição ra apta a ser disparada de forma certa no tempo corrente $\tau = 60$ da execução do modelo do processo AU no CPN Tools, mostrado na Figura 53.

As Figuras 102 e 103 mostram, respectivamente, a conclusão da execução dos 720 objetos produzidos no lugar $StrAU$ e $StrPC$. Nota-se que nos dois modelos, mesmo com ocorrências de perdas de eventos, nenhum Caso ficou sem finalizar a sua execução.

A fim de obter o identificador do Caso que teve pelo menos um evento perdido, um monitor é associado a cada transição pertencente ao modelo. Esta informação permite verificar a quantidade de Casos que tiveram eventos perdidos durante a execução tando do modelo do processo AU quanto do PC.

Ao analisar a Figura 104, verifica-se que um total de 24 Casos foram afetados pela perda de um evento no decorrer da execução. O evento que teve mais falhas foi o $ackDraft$ com 12 Casos afetados. Embora a quantidade de eventos perdidos seja considerada relativamente baixa, se nenhum tratamento for levado em consideração a execução destes Casos fica num estado de espera indefinida, caracterizando um estado de *deadlock*.

Para avaliar os resultados obtidos nesta execução, uma nova simulação é realizada onde as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

Os modelos utilizados nesta nova execução são os mesmos apresentados nas Figuras 53 e 54, porém com as interpretações associadas às transições podendo ser avaliadas apenas como verdadeira ou falsa, ou seja, nenhuma interpretação pode ser avaliada como incerta.

Uma vez desconsideradas as interpretações incertas e, consequentemente, os disparos

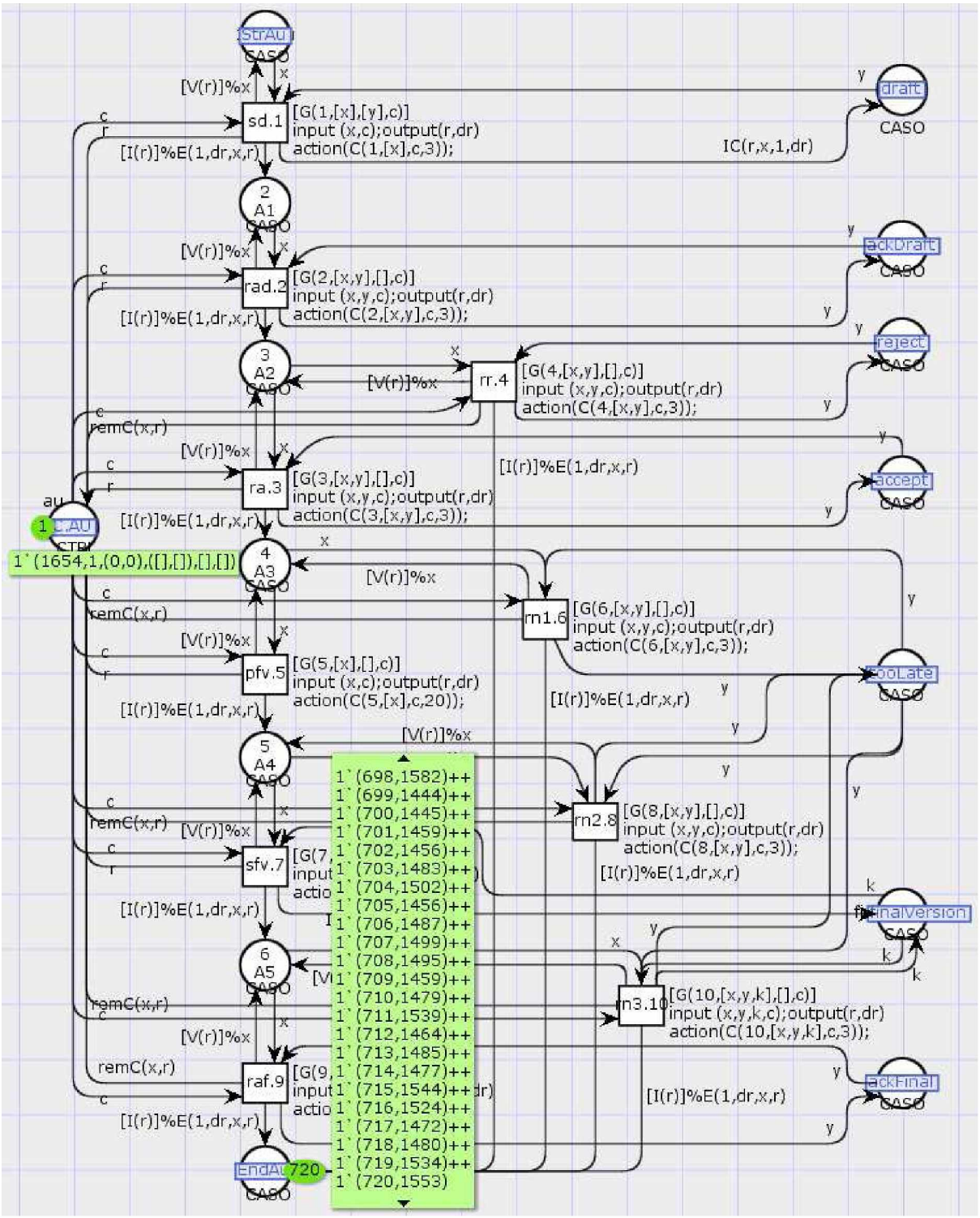


Figura 102 – Visualização da execução do modelo do processo *AU* no CPN Tools, mostrado na Figura 53, após a execução dos 720 objetos produzidos.

incertos, a execução de alguns Casos entram em um estado de espera indefinida. Assim sendo, a execução destes Casos pode não ser concluída, como mostrado na Figura 105, após a conclusão da execução dos 720 Casos produzidos nos lugares $StrAU$. Nota-se que

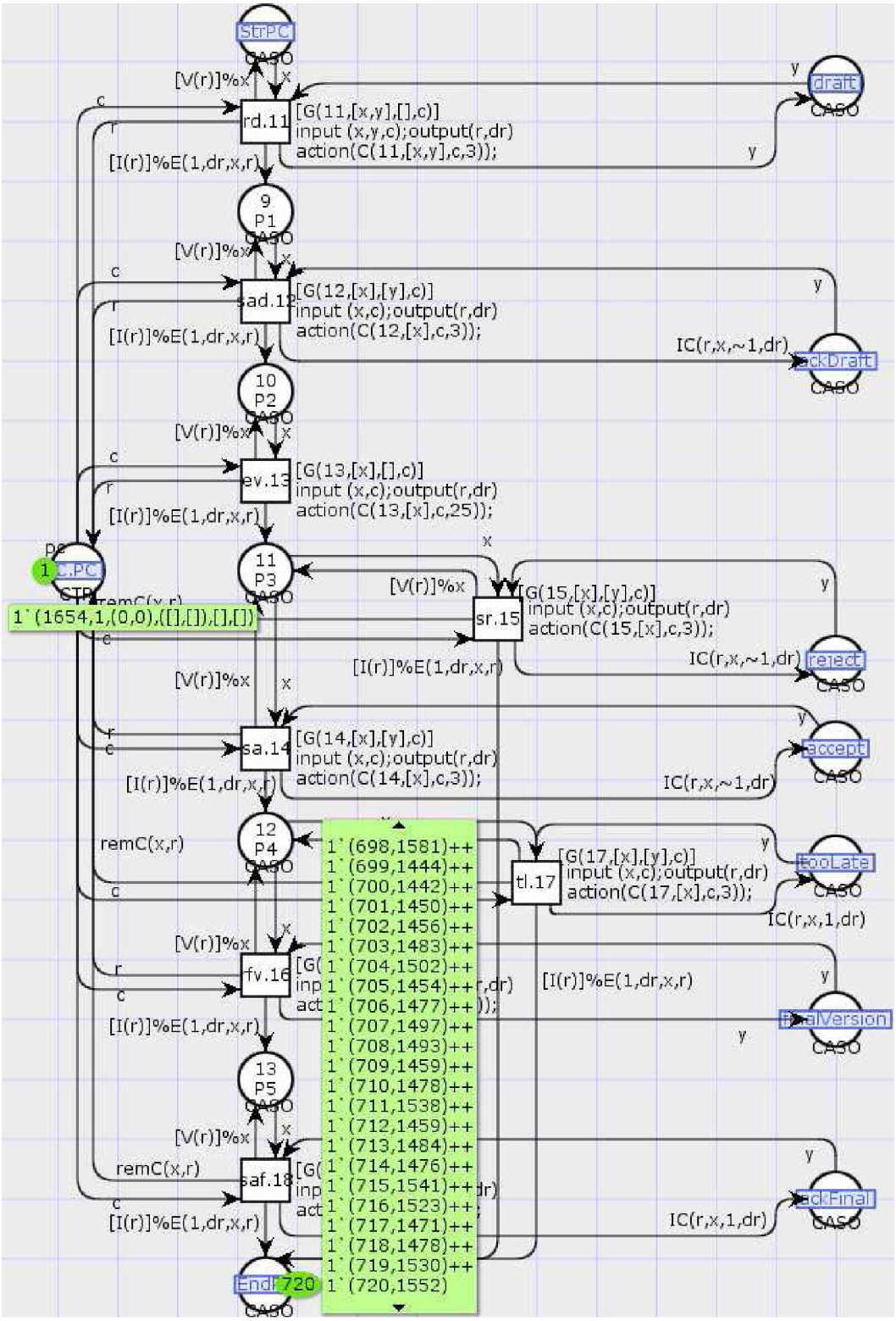


Figura 103 – Visualização da execução do modelo do processo PC no CPN Tools, mostrado na Figura 54, após a execução dos 720 objetos produzidos.

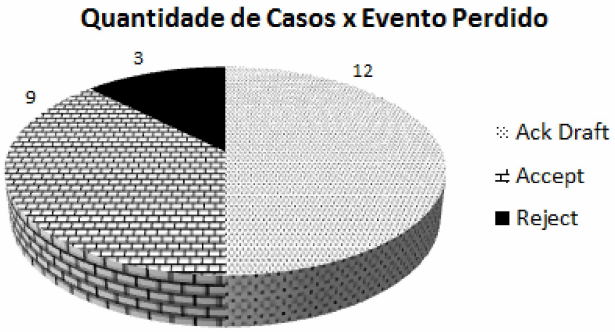


Figura 104 – Quantidade de Casos finalizados que tiveram um determinado evento perdido durante a execução do modelo do processo *AU* no CPN Tools, mostrado na Figura 53.

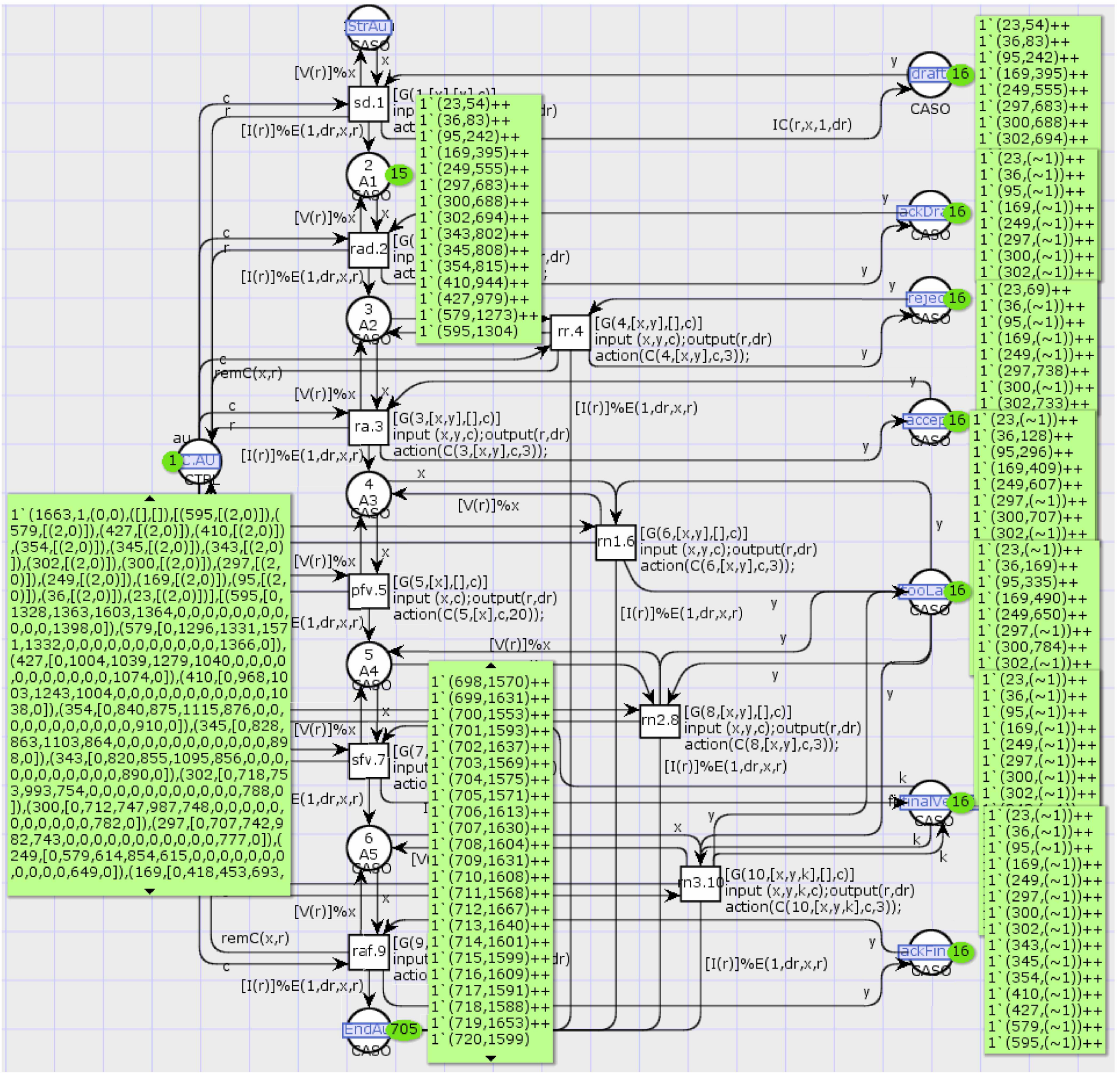


Figura 105 – Visualização da execução do modelo do processo *AU* no CPN Tools, mostrado na Figura 53, após a execução dos 720 objetos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

15 Casos não concluíram sua execução devido ao não recebimento de um evento esperado. Lembrando que a transição rr é disparada quando o tempo corrente alcança o tempo limite definido na função cIL, alguns eventos perdidos relacionados ao aceite ou a recusa do esboço do artigo (respectivamente, os lugares de comunicação *accept* e *reject*) foram tratados de maneira geral como uma rejeição do esboço.

6.3.5 Resultado da Simulação de um Processo modelado em IOWF-net Possibilística que aceita Desvios em casos de existência de *Deadlocks*

A Figura 106 mostra a inserção do lugar *next* e da transição *arv* no modelo do processo que precede a apresentação de um artigo em uma conferência (modelos nas Figuras 56, 57 e 58) apresentado na Figura 58. Observa-se que, tal como no modelo descrito na Figura 97, um objeto composto pelo identificador *id* e pelo valor negativo -1 é produzido em cada um dos lugares de comunicação e as sete transições são adicionadas no modelo descrito na Figura 106.

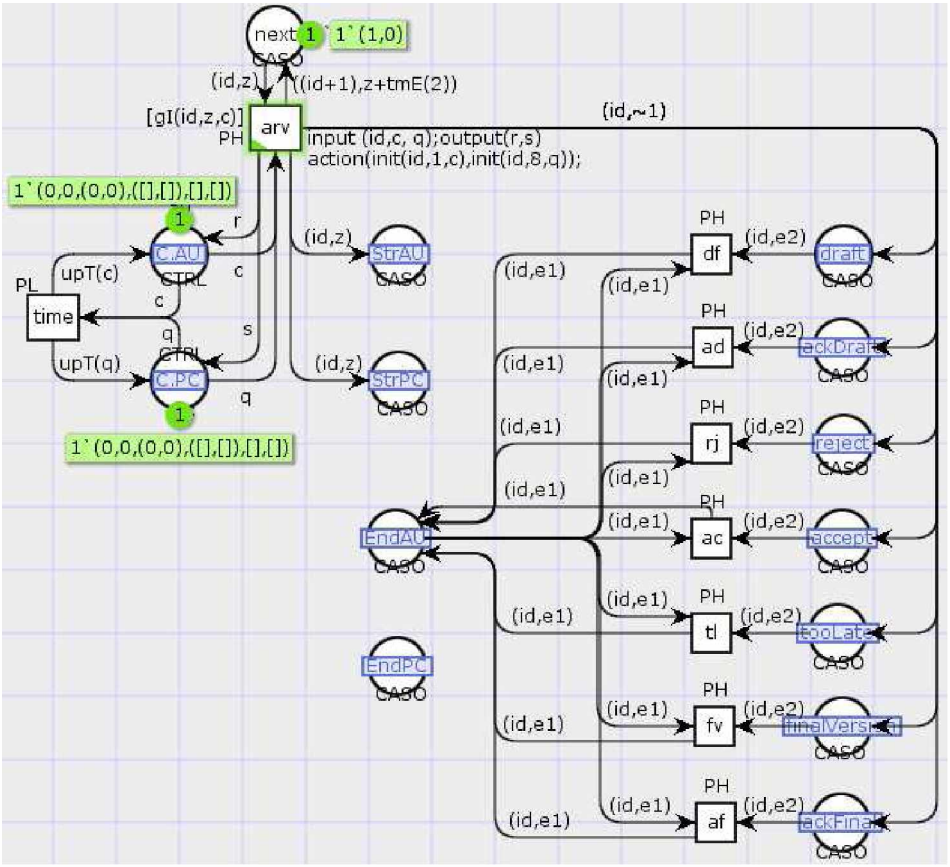


Figura 106 – Lugar *next* e transição *arv* em conjunto com as outras sete transições adicionais no modelo apresentado na Figura 58.

A declaração das variáveis id, z, $e1$ e $e2$ é idêntica à descrita na subseção anterior. A declaração da função cIL, responsável por determinar o tempo limite para o envio da

versão final do artigo, é descrita a seguir:

$$\mathrm{fun}\ \ \mathrm{cIL}(\mathrm{tm}) \ = \ [(95+\mathrm{tm})]$$

Por fim, os modelos descritos nas Figuras 56, 57 e 58, acrescido do lugar *next* e das oito transições (*arv*, *df*, *ad*, *rj*, *ac*, *tl*, *fv* e *af*), tal como mostrado na Figura 106, é executado e as Figuras 107 e 108 mostram, respectivamente, uma replicação registrada no CPN Tools do estado do modelo do processo *PC* no tempo corrente $\tau = 181$ e outra do processo *AU* no tempo corrente $\tau = 185$. Considerando estes dois estados em particular e lembrando que os lugares de comunicação representam a chegada ou envio de sinalizações de eventos entre os modelos dos processos, 6 observações podem ser realizadas. São elas:

1. no modelo do processo *PC*, 5 Casos estão esperando o envio do esboço do artigo

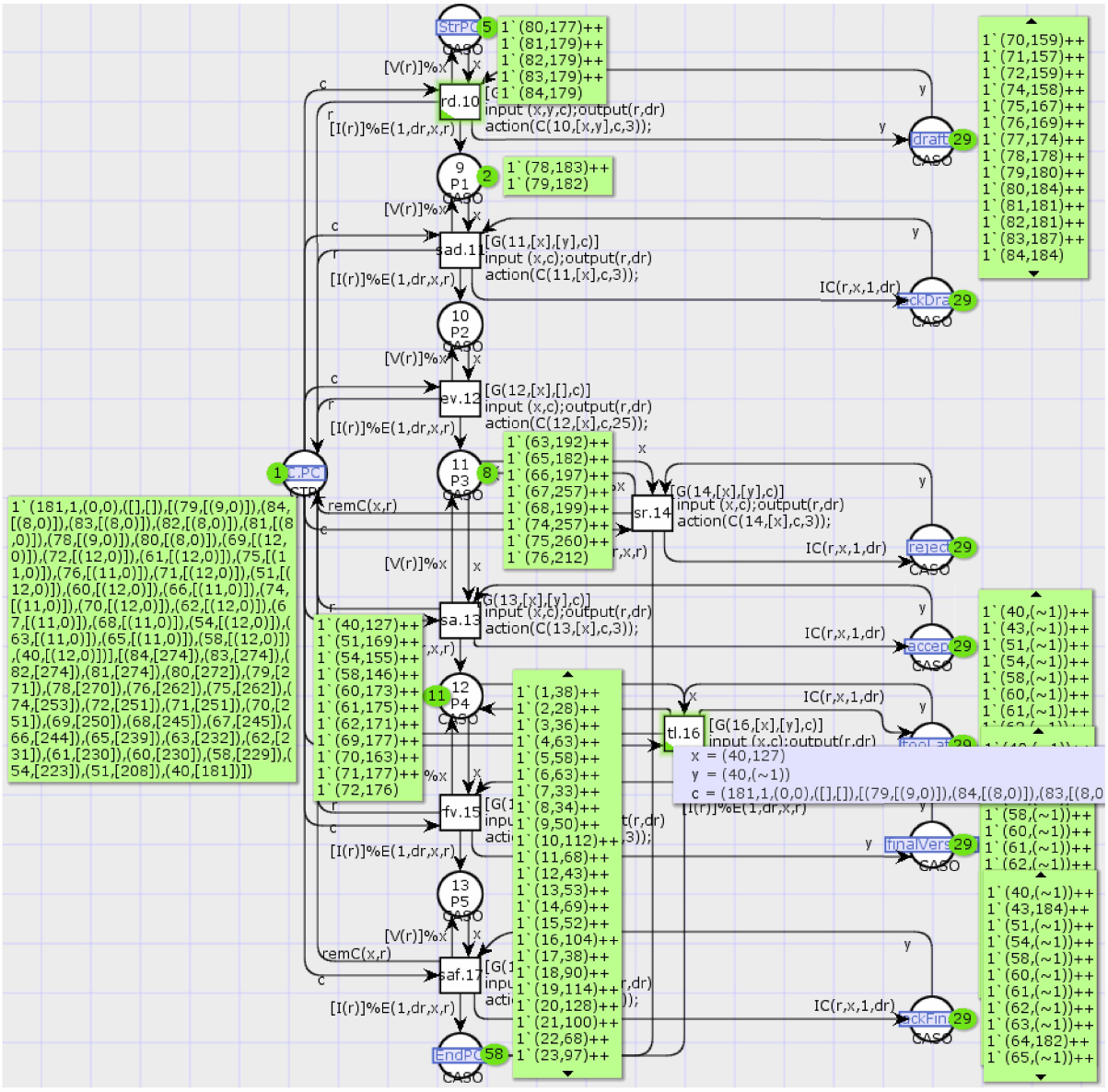


Figura 107 – Visualização da execução do modelo do processo *PC* no CPN Tools, mostrado na Figura 57, no tempo corrente $\tau = 181$.

para iniciar sua execução enquanto 21 já estão em execução[16]. Observe-se que tanto o Caso com $id = 81$ quanto o com $id = 82$ estão aptos a iniciarem a execução neste momento considerando que o esboço do artigo foi enviado (os objetos $(81, 181)$ e $(82, 181)$ se encontram no lugar de comunicação $draft$);

2. a execução de 58 Casos foram concluídas no modelo do processo PC, ou seja, o lugar $EndPC$ contém 58 Casos. Entretanto, ao observar os lugares de comunicação, pode-se notar que três dos 58 Casos já concluídos ainda não foram concluídos no modelo do processo AU, considerando $\tau = 181$;

3. o tempo limite para enviar a versão final do artigo para o modelo do processo PC foi alcançado para o Caso com identificador 40. Desta forma, a transição tl é disparada com certeza. Note, através da Figura 108, que a chegada desta informação no mode-

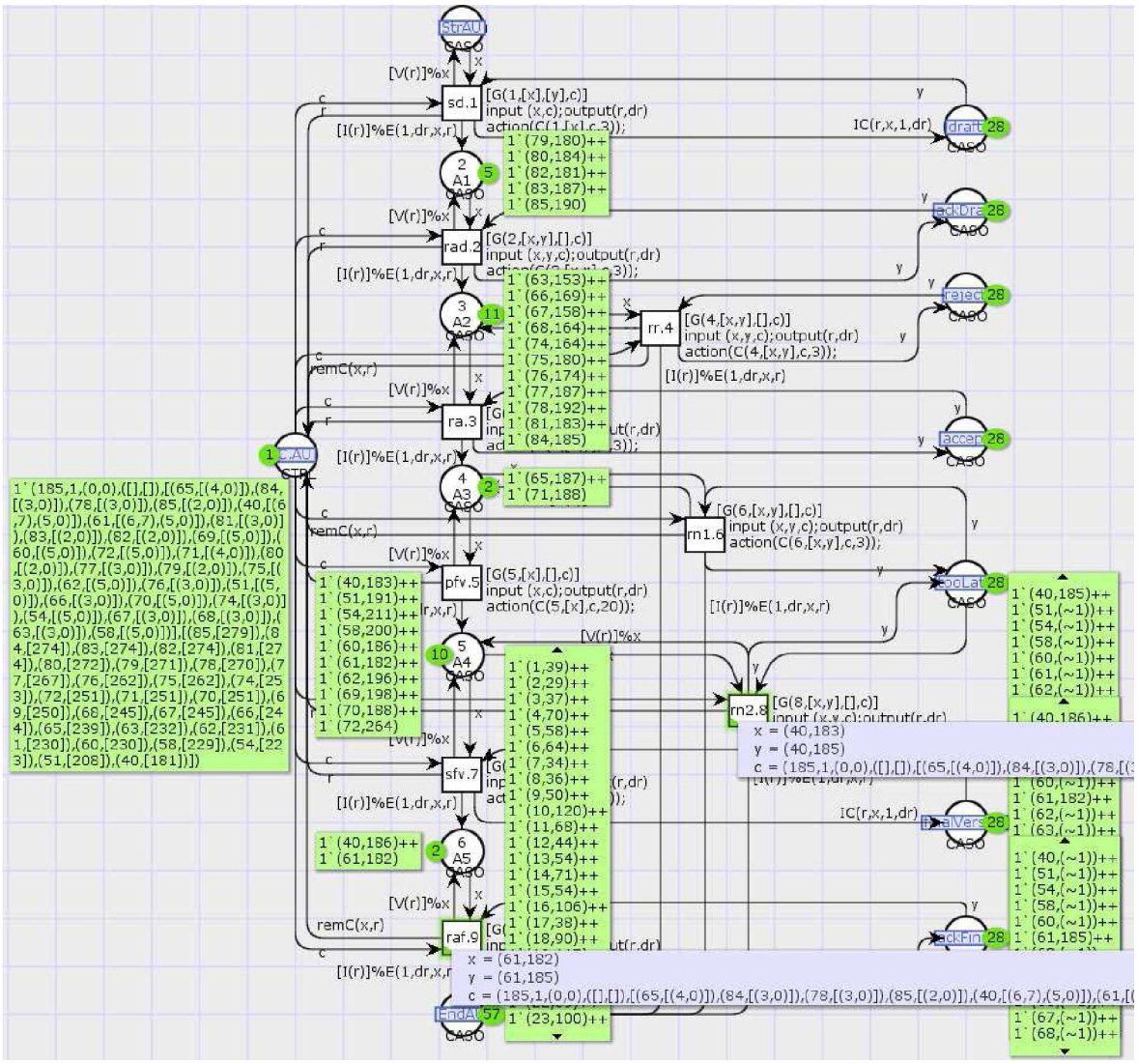


Figura 108 – Visualização da execução do modelo do processo AU no CPN Tools, mostrado na Figura 56, no tempo corrente $\tau = 185$.

[16] Objetos em lugares diferentes com mesmo identificador se referem a um mesmo Caso em execução.

lo do processo AU teve um *delay* de 4 minutos, ou seja, o objeto $(40, 185)$ presente no lugar de comunicação *tooLate* estará disponível para o modelo do processo AU somente quando o tempo corrente alcançar o valor 185. Assim sendo, a transição sfv responsável por enviar a versão final do artigo ao modelo do processo PC pode ser disparada de forma errônea antes do tempo corrente alcançar o valor 185.

4. considerando o modelo do processo AU no tempo corrente $\tau = 185$, 28 Casos estão em execução e a execução de 57 Casos foi concluída, ou seja, o lugar $EndAU$ contém 57 Casos;

5. no modelo do processo AU o Caso com identificador 61 recebeu o reconhecimento, por parte do modelo do processo PC, do envio da versão final do artigo. Desta forma, o disparo incerto realizado na transição sfv no tempo $\tau = 182$ pode ser finalizado para que a execução do Caso possa ser concluído;

6. o Caso com identificador 40 teve o tempo limite para enviar a versão final do artigo para o modelo do processo PC alcançado. Desta forma, a transição $rn2$ deve ser disparada para este Caso. Entretanto, devido ao *delay* de 4 minutos ocasionado no momento do envio da notificação por parte do modelo do processo PC à AU, a versão final do artigo foi enviada ao PC no tempo $\tau = 183$. Assim sendo, o disparo da transição sfv deve ser cancelado para que a transição $rn2$ possa ser disparada e a execução do Caso concluída.

Na Figura 108, tanto a transição $rn2$, para o Caso de $id = 40$, quanto a raf, para o Caso de $id = 61$, estão aptas a serem disparadas. Levando em consideração que ambas estão habilitadas por uma marcação imprecisa e as funções de autorização associadas a elas são avaliadas como verdadeiras, o algoritmo de defuzzificação é então chamado para posteriormente dispará-las de forma certa.

A Figura 109 mostra, através da marcação do lugar de controle exibida na variável c, no tempo corrente 185, que o objeto de identificador 40 está em processo de defuzzificação por causa da transição de identificador 8. Além disto, o disparo incerto realizado na transição sfv será cancelado através do disparo da transição raf representada pelo identificador 9 como destacado na figura e nenhum outro disparo incerto é finalizado.

Ao observar a Figura 110, pode-se notar que o procedimento de defuzzificação foi finalizado e a transição $rn2$, além de estar apta a ser disparada, está habilitada de forma precisa. Deste modo, a mesma será disparada de forma certa. Após o disparo da transição $rn2$, a transição raf é a próxima a ser disparada para o Caso de $id = 61$. Lembrando que a mesma se encontra com uma marcação imprecisa, o algoritmo de defuzzificação também é chamado para recuperar o estado do modelo do processo para este Caso.

A Figura 111 mostra, através da marcação do lugar de controle exibida na variável c, também no tempo corrente 185, que o objeto de identificador 61 está em processo de

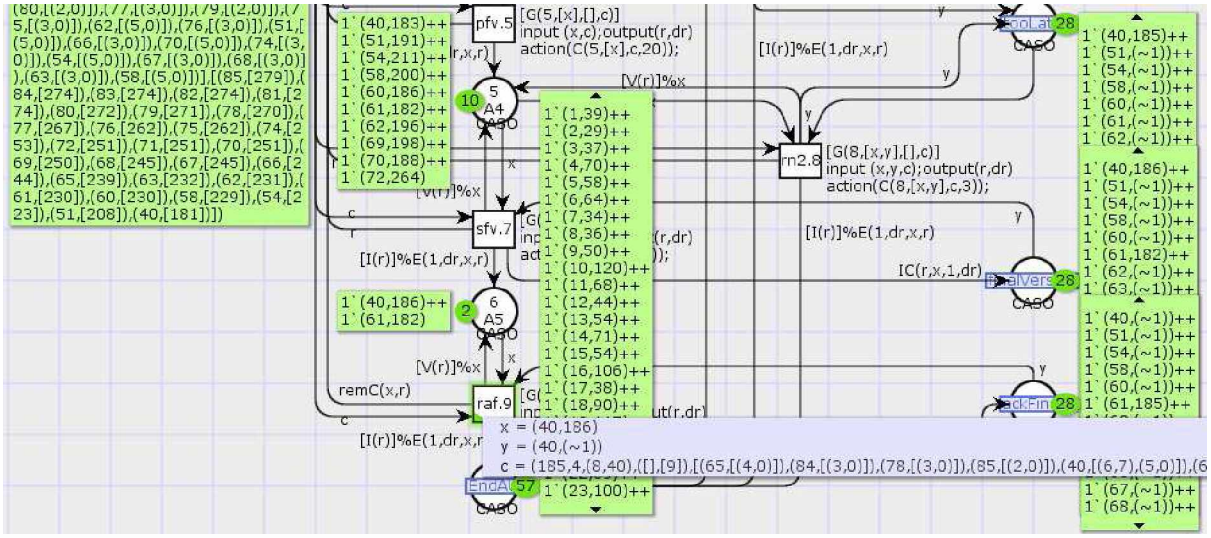


Figura 109 – Transição sfv apta a ter o disparo incerto cancelado através do disparo da transição raf no tempo corrente $\tau = 185$ da execução do modelo mostrado na Figura 56.

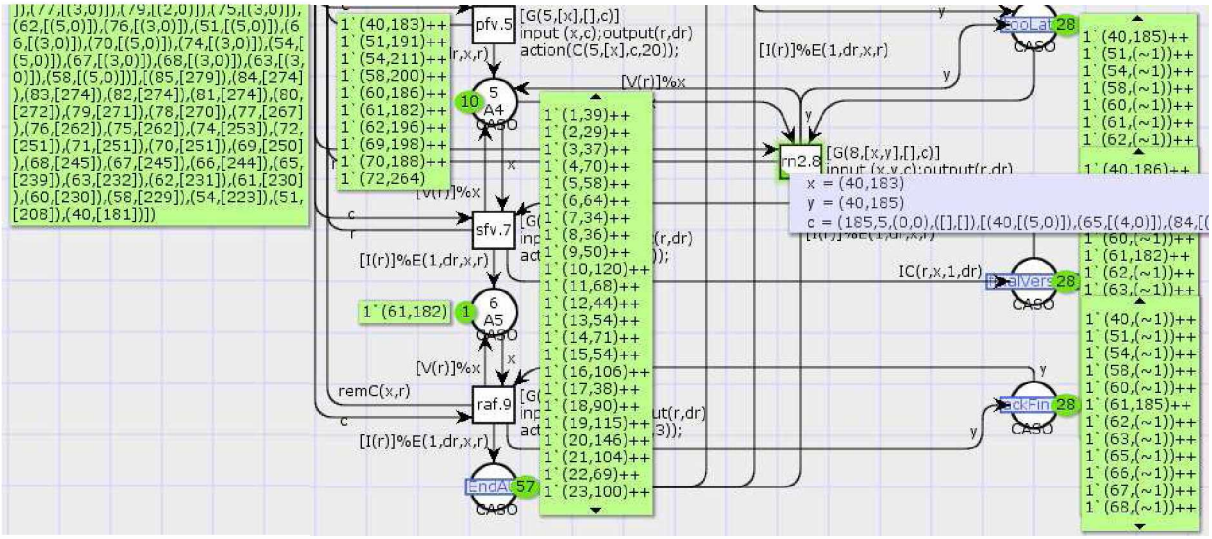


Figura 110 – Transição $rn2$ apta a ser disparada de forma certa no tempo corrente $\tau = 185$ da simulação do modelo do processo AU implementado no CPN Tools, mostrado na Figura 56.

defuzzificação por causa da transição de identificador 9. Além disto, o disparo incerto realizado na transição sfv, representada pelo identificador 7 e destacada na figura, será finalizado e não existe qualquer outro disparo incerto a ser cancelado.

Após o disparo incerto da transição sfv ser finalizado e o processo de defuzzificação concluído, a transição raf pode ser dispara de forma certa, tal como mostrado na Figura 112, dado que a mesma estará habilitada de forma precisa e a função de autorização associada a ela avaliada como verdadeira. Ao fim dos disparos realizados nas transições $ra2$ e raf, pode-se perceber que um possível estado de *deadlock* para o Caso com $id = 40$ foi evitado através do uso dos disparos incertos e a confirmação do recebimento do artigo

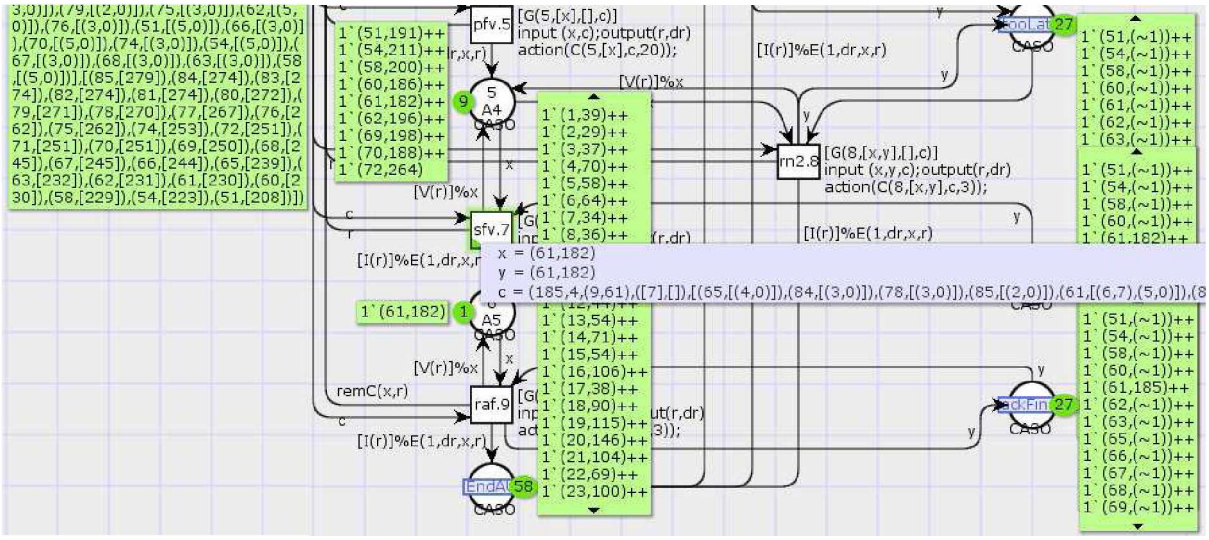


Figura 111 – Transição sfv apta a ter o disparo incerto finalizado no tempo corrente $\tau = 185$ da execução do modelo do processo AU no CPN Tools, mostrado na Figura 56.

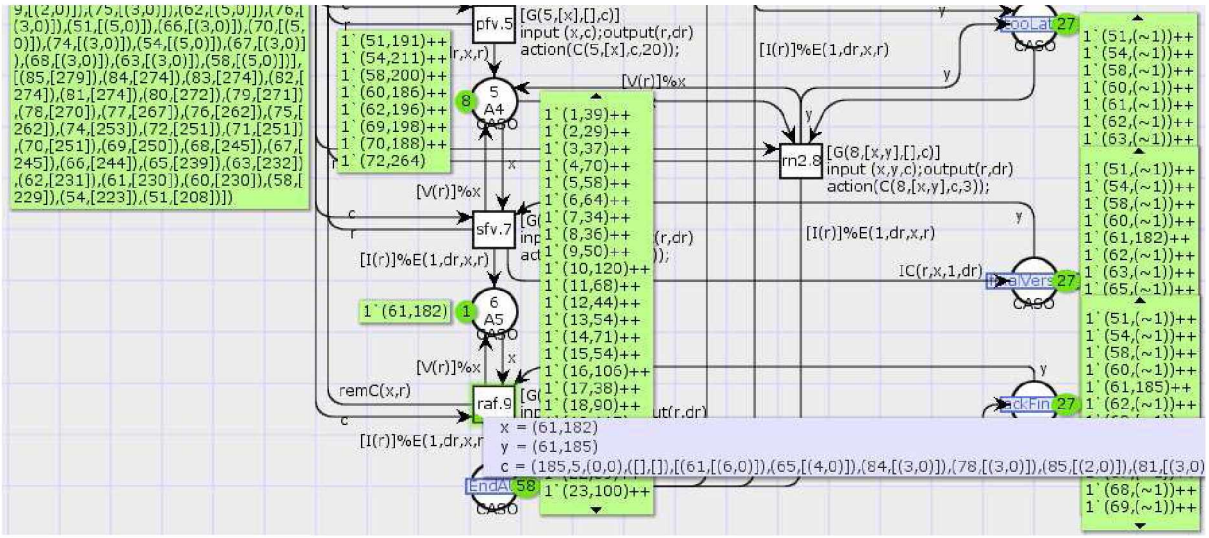


Figura 112 – Transição raf apta a ser disparada de forma certa no tempo corrente $\tau = 185$ da execução do modelo do processo AU no CPN Tools, mostrado na Figura 56.

para o Caso de $id = 61$ foi concluída sem nenhum transtorno.

As Figuras 113 e 114 mostram, respectivamente, a conclusão da execução dos 720 Casos produzidos no lugar $StrAU$ e $StrPC$. Note-se que nos dois modelos, mesmo com possíveis ocorrências de *deadlock*, nenhum Caso ficou sem finalizar a sua execução. Entretanto, quando os disparos incertos não são considerados, a transição sfv pode ser disparada de forma errônea impedindo assim a correta conclusão da execução do Caso em questão através da transição $rn2$.

Para analisar os resultados obtidos nesta execução, uma nova simulação é realizada onde as interpretações incertas e, consequentemente, os disparos incertos são desconsi-

derados. Os modelos utilizados nesta nova execução são os mesmos apresentados nas Figuras 56 e 57, porém com as interpretações associadas às transições podendo ser avaliadas apenas como verdadeira ou falsa, ou seja, nenhuma interpretação pode ser avaliada como incerta.

Uma vez desconsideradas as interpretações incertas e, consequentemente, os disparos

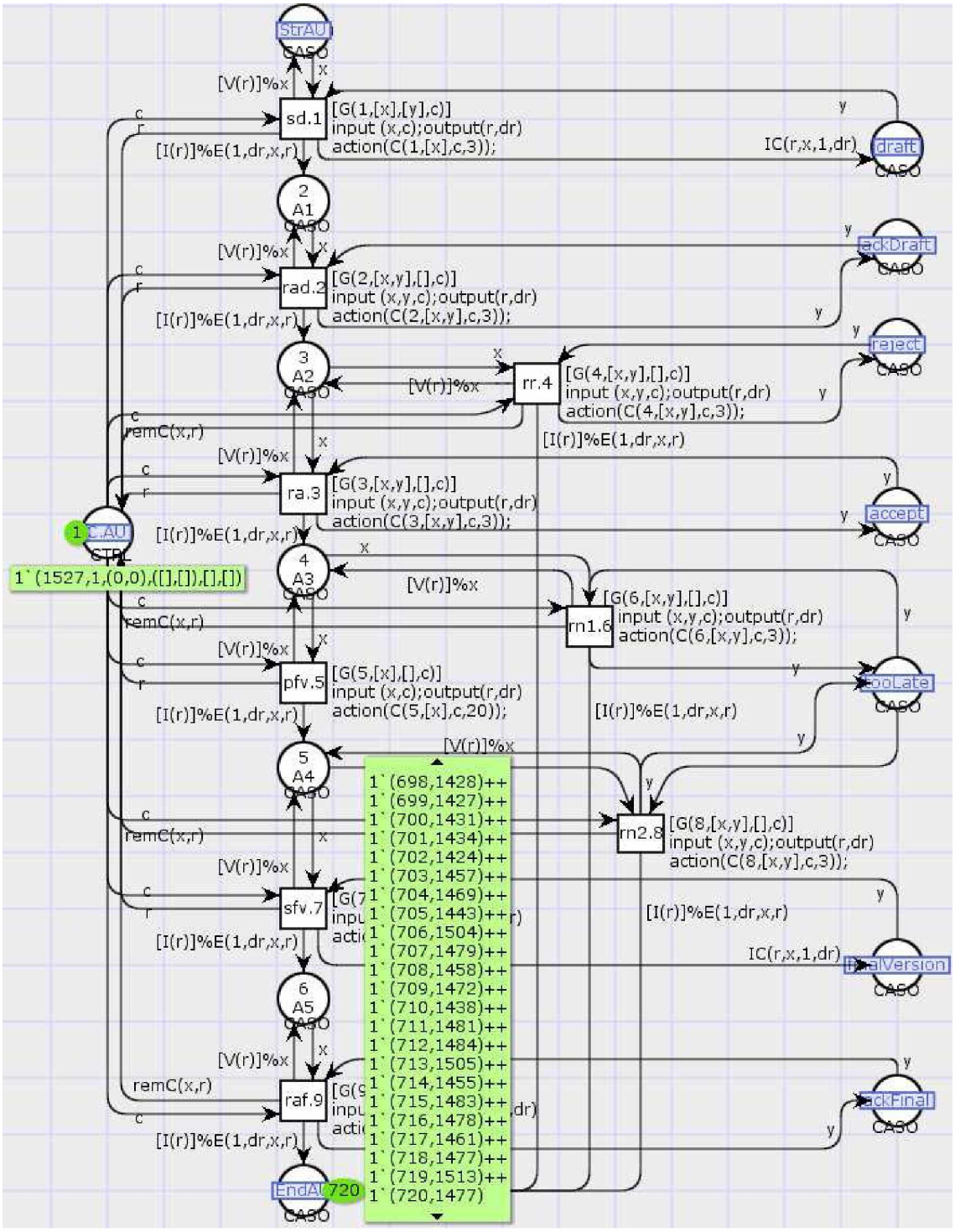


Figura 113 – Visualização da execução do modelo do processo *AU* no CPN Tools, mostrado na Figura 56, após a execução dos 720 objetos produzidos.

incertos, alguns Casos podem não ser concluídos devido o disparo errôneo das transições sfv e $rn2$. Na Figura 115, pode-se observar que 32 Casos não tiveram sua execução concluída quando as interpretações associadas às transições podem ser avaliadas apenas

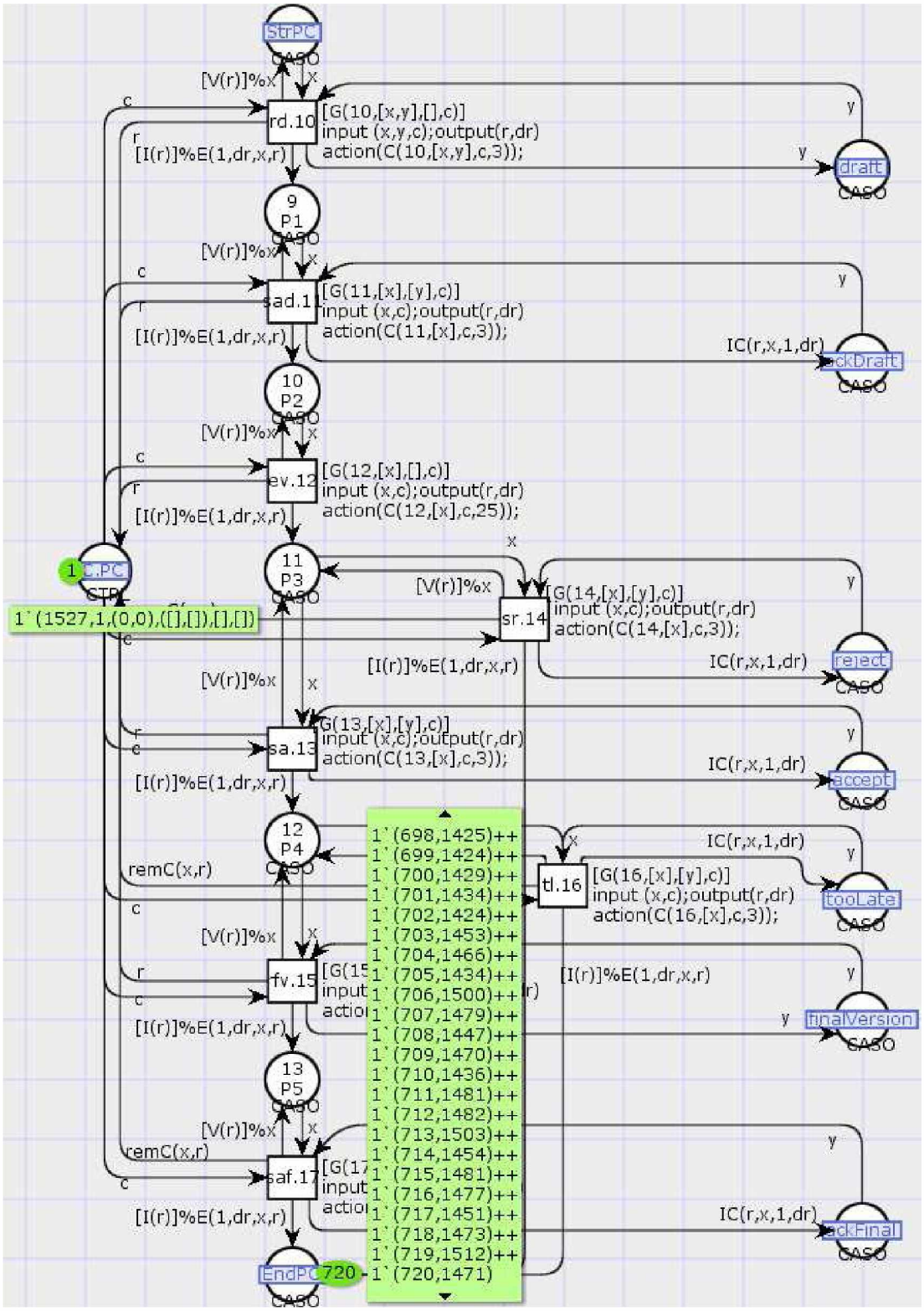


Figura 114 – Visualização da execução do modelo do processo PC no CPN Tools, mostrado na Figura 57, após a execução dos 720 objetos produzidos.

como verdadeira ou falsa, ou seja, 688 Casos se encontram no lugar $EndAU$ e 32 Casos no lugar $A5$. A existência destes 32 Casos no modelo do processo AU ocorre por causa do disparo errôneo realizado na transição sfv quando, o correto, era o disparo da transição $rn2$. Em contrapartida, no modelo do processo PC, todos os 720 Casos foram concluídos, como pode ser observado na Figura 116, ou seja, os 720 Casos produzidos no lugar $StrPC$ estão localizados no lugar $EndPC$ ao fim da execução.

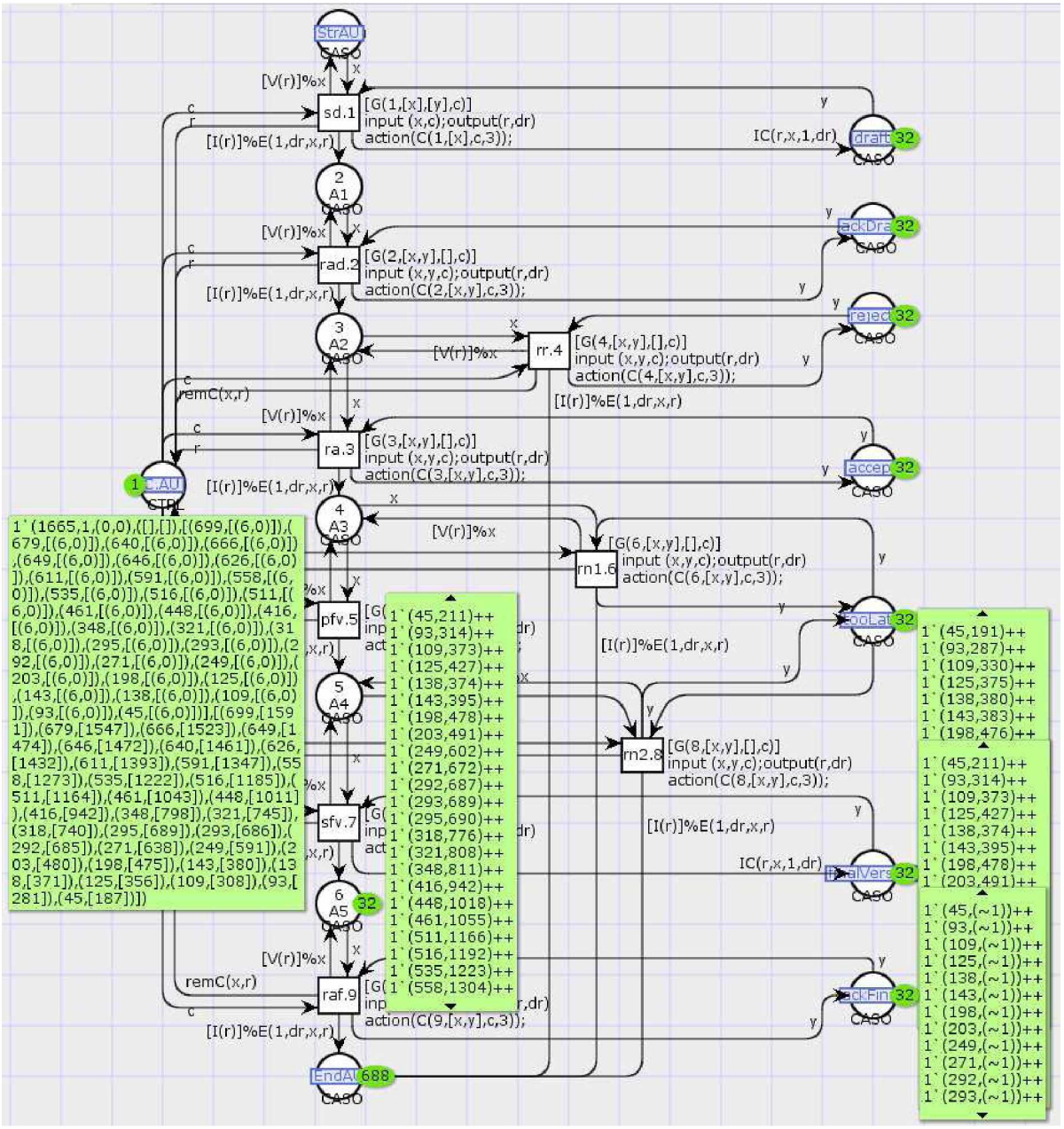


Figura 115 – Visualização da execução do modelo do processo PC no CPN Tools, mostrado na Figura 57, após a execução dos 720 objetos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

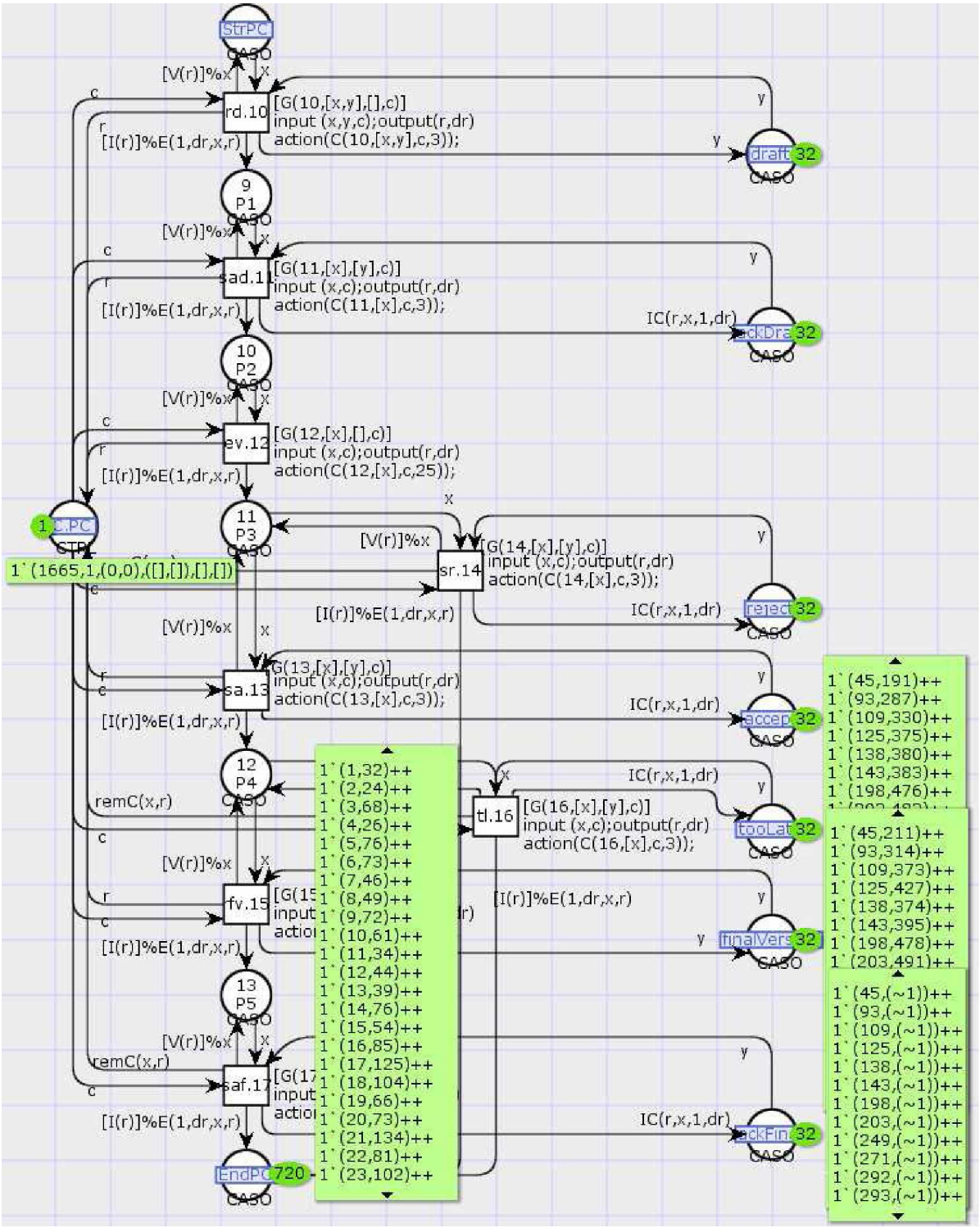


Figura 116 – Visualização da execução do modelo do processo *PC* no CPN Tools, mostrado na Figura 57, após a execução dos 720 objetos produzidos quando as interpretações incertas e, consequentemente, os disparos incertos são desconsiderados.

6.4 Considerações Finais

Os resultados obtidos ao fim de cada execução comprovam que cada uma das características esperadas, de acordo com o tipo de desvio considerado e o comportamento flexível produzido pelo "jogador", foi contemplada. Tais resultados demonstram que, mesmo com um modelo de processo estruturado de forma procedimental, é possível, a partir da noção de disparo incerto, contornar situações imprevistas sem que um estado inconsistente seja alcançado.

Além disso, em alguns casos, uma melhoria no desempenho dos processos executados foi obtida quando a noção de disparo incerto foi considerada. Por fim, as execuções realizadas nos cinco modelos de processos, cada um com características diferentes, comprovam que o "jogador" de WF-net possibilística apresentado na Seção 4.1 é passível de ser implementado em sistemas de gerenciamento de processos de negócios usados em ambientes administrativos reais.

Capítulo 7

Conclusão

Nos WfMS's, a tarefa de gerenciamento é feita a partir de modelos do processo ou da definição do *workflow*. Quando se considera os modelos até então definidos, ou eles envolvem apenas a ordenação rígida das atividades dos processos de negócios ou permitem certas alterações durante a execução do mesmo. A ordenação rígida dificulta a modificação dos modelos e, como consequência, são frequentemente sujeitos à intervenções manuais na medida que os funcionários humanos tentam manipulá-los de forma a adaptarem-os às mudanças das práticas de trabalho. Nos modelos de processos que permitem uma certa flexibilidade durante a sua execução, a decidibilidade das suas propriedades tendem se a perder.

Nesta tese, a WF-net possibilística e a IOWF-net possibilística propostas permitem lidar com problemas de desvios e existências de problemas estruturais no modelo dos processos na fase de execução dos mesmos. Diferentemente dos modelos já propostos, a noção de disparo incerto permite definir roteiros alternativos sem que o modelo do processo seja alterado fisicamente ou que se tornem complexos e extensos. Além disto, a decidibilidade das propriedades dos modelos não é perdida.

Neste capítulo, as principais contribuições, os trabalhos futuros e as produções bibliográficas obtidos com o desenvolvimento desta pesquisa são apresentados. As principais contribuições são apresentadas na Seção 7.1, os trabalhos futuros que poderão ser desenvolvidos considerando os resultados obtidos nesta pesquisa na Seção 7.2 e, finalmente, na Seção 7.3, a produção bibliográfica resultante no contexto da presente pesquisa são apresentadas.

7.1 Principais Contribuições

Este trabalho tem como principal contribuição duas novas técnicas de modelagem, nomeadas *WorkFlow net* possibilística e *WorkFlow net* interorganizacional possibilística (IOWF-net possibilística), juntamente com um novo mecanismo de inferência especializado, do tipo *token player*. Assim sendo, problemas estruturais como por exemplo *deadlock*

e situações de desvios, tais como falhas de comunicação, inconsistências/desvios/exceções e decisões humanas não completamente previstas, em processos de negócios modelados por uma WF-net são tratadas através da noção de disparo incerto com o intuito de proporcionar um certo grau de flexibilidade no comportamento dos modelos durante a sua execução. Em particular, diferentes situações de desvios foram consideradas e descritas através de quatro modelos de processos.

No caso mono-processo, o comportamento flexível do novo modelo foi usado para de incorporar o padrão de mudança "*Parallelize Process Fragments*" definido por Weber, Reichert e Rinderle-Ma (2008) e a noção de cancelamento, apontada por Aalst et al. (2003) como um importante mecanismo nos WfMS's, quando a execução de algumas atividades pode levar a finalização de outras em determinadas circunstâncias. No caso de processos interorganizacionais, o comportamento flexível do novo modelo foi usado para contornar situações de *deadlock* e lidar com possíveis problemas de comunicação, tais como mensagens perdidas, atrasadas ou inesperadas entre os modelos dos processos parceiros de forma a evitar situações de inconsistências no modelo dos processos.

Comparando o comportamento dos novos modelos com os modelos de processos tradicionais, a principal vantagem é que as situações de desvios nos comportamentos esperados são detectadas em tempo de execução e devidamente tratados sem necessidade de rever a estrutura dos modelos. Em particular, um desvio no comportamento esperado não se caracteriza mais como um estado de inconsistência e, em certos casos, através do raciocínio incerto os novos modelos permitem aos sistemas de gerenciamento de processos de negócios a adoção de soluções com uma melhoria significativa no desempenho. Além disto, o fato de trabalhar com um modelo formal de processo faz com que as "boas propriedades" dos sistemas modelados sejam mantidas mesmo após aplicação de uma nova semântica operacional associada à ocorrência de eventos incertos em sistemas administrativos.

Levando em consideração as Hipóteses desta pesquisa apresentadas na Seção 1.3 e as propostas expostas nos Capítulos 4 e 5, bem como os resultados dos experimentos mostrados no Capítulo 6, tem-se que:

1. a hipótese "o padrão de mudança '*Parallelize Process Fragments*' proposto por Weber, Reichert e Rinderle-Ma (2008) para paralelizar fragmentos do modelo do processo que, originalmente, eram puramente sequencial, pode ser incorporado nos modelos dos processos a partir do uso dos disparos incertos" foi validada através da aplicação dos disparos incertos nos modelos de mono-processo de forma a paralelizar algumas de suas atividades (Seção 4.2). Esta paralelização permite suportar certas sequências de eventos não previstas no modelo do processo quando os recursos envolvidos não seguem rigorosamente as sequências predeterminadas. Desta forma, a flexibilidade incorporada no modelo de processo proporciona uma melhoria na produtividade do sistema derivada de práticas de trabalhos na área administrativa além de evitar possíveis estados de inconsistência quando os recursos envolvidos não

respeitam a estrutura do modelo do processo;

2. a hipótese "os padrões de projeto 'Cancel activity' e 'Cancel case' propostos por Aalst et al. (2003), responsáveis por capturar a interferência de uma atividade na execução das outras atividades em certas circunstâncias, podem ser incorporados nos modelos dos processos a partir do uso dos disparos incertos sem perder a decidibilidade da propriedade *Soundness*" foi validada considerando a noção de cancelamento incorporado à WF-net possibilística (Seção 4.3). Levando em consideração que um pedido de cancelamento pode ser considerado como um desvio/distúrbio do processo e que, se não tratado adequadamente, um estado de inconsistência é alcançado, a flexibilidade fornecida pelo "jogador" de WF-net possibilística permite executar o pedido de cancelamento de forma a recuperar o estado normal de funcionamento do modelo do processo sem atingir qualquer tipo de inconsistência;

3. a hipótese "as falhas decorrentes da comunicação entre os processos parceiros podem ser contornadas através dos disparos incertos nos modelos de processos interorganizacionais" foi validada através da aplicação dos disparos incertos nos modelos de processo interorganizacional de forma a contornar possíveis falhas de comunicação ocasionadas durante a sua execução (Seção 5.2). A noção de disparo incerto neste problema é considerado com o intuito de incorporar uma flexibilidade ao modelo do processo de forma a torná-lo robusto quando não há um total controle sobre a geração dos eventos. Desta forma, situações imprevistas ocasionadas por desvios de comportamento durante a execução do modelo do processo interorganizacional podem ser contornadas. A diferença deste problema em relação ao tratado na dissertação de mestrado (REZENDE, 2013) se dá nos tipos de processos considerados, ou seja, na dissertação de mestrado (REZENDE, 2013) levou-se em consideração os mono-processos e, aqui, são considerados os processos interorganizacionais;

4. a hipótese "os estados de *deadlock* presentes nos modelos dos processos interorganizacionais devido à sincronização entre os processos paralelos podem ser evitados em tempo de execução através dos disparos incertos" foi validada considerando o tratamento à situação de *deadlock* ocasionada devido à sincronização de dois processos parceiros num modelo interorganizacional (Seção 5.3). Levando em consideração que uma situação de *deadlock* pode ser considerada como um desvio relacionado à comportamentos extremos, a flexibilidade obtida a partir da noção de disparos incertos permite evitar as sequências de disparos errôneos, isto é, as sequências que levam a uma situação de *deadlock*;

5. a hipótese "a ferramenta de modelagem *CPN Tools* permite, através do uso da linguagem de programação funcional *Standard ML*, implementar, simular e validar o uso dos disparos incertos a fim de avaliar a robustez das técnicas propostas" foi

validada com a construção do "jogador" de WF-net possibilística para a edição e execução de cinco modelos de processos usados nas duas propostas apresentadas (Capítulo 6). A partir das execuções realizadas, pôde-se concluir que todas as características esperadas, de acordo com o tipo de desvio considerado e o comportamento flexível produzido pelo "jogador", foram contempladas. Além disso, mesmo com um modelo de processo estruturado de forma procedimental, foi possível, a partir da noção de disparo incerto, contornar situações imprevistas sem que um estado inconsistente seja alcançado e, em alguns casos, melhorar o desempenho dos processos executados. Por fim, pôde-se comprovar também que o "jogador" de WF-net possibilística apresentado no Capítulo 4 é passível de ser implementado em sistemas de gerenciamento de processos de negócios usados em ambientes administrativos reais ainda com a possibilidade de implementar diferentes mecanismos de defuzzificação.

7.2 Trabalhos Futuros

Como trabalho futuro, considera-se interessante investigar as situações de conflitos existentes quando duas ou mais transições se tornam habilitadas para o mesmo Caso e onde o disparo incerto de uma transição impacta diretamente no disparo das outras. Este impacto pode ocasionar um estado de inconsistência ou gerar um retrabalho desnecessário. Um exemplo de inconsistência é quando, após a realização de uma sequência de disparos incertos, dois eventos se tornam verdadeiros permitindo o disparo certo de duas transições distintas, entretanto, ao executar o algoritmo de defuzzificação, o disparo de uma delas é impossibilitado devido a remoção da marcação nos lugares de entrada correspondentes.

Além deste conflito direto entre as transições, existe o conflito indireto ocasionado pelas situações de rotas seletivas, ou seja, quando existe uma escolha entre duas ou mais tarefas para um mesmo Caso. Da forma como as técnicas de modelagem e o mecanismo de inferência especializado foram definidos, quando existe uma escolha entre duas transições e ambas estão disponíveis para serem disparadas de forma incerta para um mesmo Caso, a escolha da qual será pseudo-disparada é feita de forma aleatória, sem nenhum tipo de preferência.

Um outro ponto importante a destacar é em relação às condições associadas às transições através das funções de autorização. O que se deseja é torná-las flexíveis de modo que possíveis alterações possam ser feitas durante a execução do Caso, seja dependendo, por exemplo, de um banco de dados. Desta forma novos disparos incertos podem ser considerados durante a execução do modelo com o intuito de tratar outras situações não previstas ou alterar o comportamento do mesmo.

Um aspecto a investigar nos modelos dos processos de negócio é a representação dos recursos necessários para executar certas atividades. O problema é que esses recursos são geralmente finitos e precisam ser compartilhados entre as diferentes instâncias do

modelo do processo de negócio que podem estar concorrentemente em execução (FREITAS, 2017). No caso dos WfMS's, recursos podem representar tanto equipamentos físicos quanto funcionários humanos. Quando se trata de recursos humanos, tem-se a existência de incerteza associada à execução da tarefa, pois cada funcionário tem seu próprio ritmo de trabalho (FREITAS, 2017). Além disso, é necessário evitar que muitas tarefas sejam atribuídas ao mesmo recurso, pois ele pode se tornar um gargalo para o sistema.

Levando em consideração os 20 padrões de *design* definidos por Aalst et al. (2003), é interessante incorporar os padrões referentes aos diferentes tipos de sincronização das atividades executadas em paralelo ou não, tanto à WF-net possibilística quanto à IOWF-net possibilística. Desta forma, diferentes alternativas de escolha possibilitam uma maior flexibilidade na execução do Caso em questão.

A fim de obter informações úteis para serem incorporadas nas futuras iterações do modelo do processo, arquivos de *log*[1] podem ser definidos com o intuito de obter tais informações. A partir destes arquivos, informações, tais como remoção, inserção ou redefinição de uma sequência de atividades, podem ser obtidas permitindo assim novos estudos sobre os processos envolvidos. Além disto, regras automáticas de pseudos-disparo poderiam ser geradas a partir destes arquivos de *log*. Esta abordagem considera o *machine learning* (MICHALSKI; CARBONELL; MITCHELL, 2013) e o *process mining* (AALST, 2016).

Por fim, é interessante aplicar os modelos propostos a outros tipos de sistemas quando a noção de distribuição de tarefas em relação a um conjunto de recurso fica incerta. Por exemplo, nos processos de desenvolvimento de projetos de software os disparos incertos podem gerar informações para o gerente de projeto com o intuito de melhorar ou redefinir a forma como o trabalho é executado.

7.3 Contribuições em Produção Bibliográfica

Até este momento de escrita deste texto as seguintes produções bibliográficas foram realizadas e apresentadas - no caso de conferências - (REZENDE; JULIA; CARDOSO, 2014), (REZENDE; JULIA, 2015a), (REZENDE; JULIA; CARDOSO, 2016a) e (REZENDE; JULIA; CARDOSO, 2016b).

O artigo (REZENDE; JULIA; CARDOSO, 2014) aplica à IOWF-net possibilística aos modelos dos processos de negócios interorganizacionais para recuperar o estado do modelo do processo após a ocorrência de falhas de comunicação tais como mensagens perdidas, atrasadas ou inesperadas entre os processos como mostrado na Seção 5.2. Este artigo foi publicado e apresentado no ICEIS (International Conference on Enterprise Information System) no ano de 2014.

[1] Um *log*, em termos gerais, é um registro de acontecimentos ou de atividades. Em se tratando de computadores, é um registro gerado por um serviço ou aplicativo específico (às vezes pelo próprio sistema operacional).

No artigo (REZENDE; JULIA, 2015a), a IOWF-net possibilística, a noção de disparo incerto é usada para se desviar, em tempo de execução, das sequências responsáveis pelos estados de *deadlock* nos modelos dos processos de negócio interorganizacionais como mostrado na Seção 5.3. Este artigo foi publicado e apresentado no ICEIS no ano de 2015.

Devido ao prêmio recebido de *best student paper* da conferência ICEIS pelo artigo (REZENDE; JULIA, 2015a), uma versão estendida com o título "Possibilistic WorkFlow Net for Deadlock Avoidance in Interorganizational Business Processes" foi incluída no periódico LPBIP (*Lecture Notes in Business Information Processing*) publicado pela Springer em 2015 (REZENDE; JULIA, 2015b).

Os artigos (REZENDE; JULIA; CARDOSO, 2016b) e (REZENDE; JULIA; CARDOSO, 2016a) foram publicados e apresentados no ICEIS no ano de 2016. O artigo (REZENDE; JULIA; CARDOSO, 2016b) aplica a WF-net possibilística aos processos de negócio em execução com roteiro puramente sequencial para permitir uma paralelização parcial na ordenação das atividades como mostrado na Seção 4.2. Este artigo também foi indicado a *best poster* da conferência.

O artigo (REZENDE; JULIA; CARDOSO, 2016a) apresenta o procedimento baseado nas regiões de cancelamento e nos disparos incertos mostrado na Seção 4.3. Este procedimento tem como objetivo incorporar a noção de cancelamento à WF-net possibilística de forma a considerar o cancelamento da execução de certas atividades e, em alguns casos, do processo.

Por fim, o artigo com o título "Possibilistic Workflow nets for deviation problems in Business Processes" foi submetido ao periódico "*International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*" e ainda está sob revisão. Este artigo considera a modelagem e a execução no CPN Tools, considerando o mecanismo de inferência especializado do tipo *token player* definido na Seção 4.1, do modelo do processo de tratamento de reclamações modelado por uma WF-net possibilística (modelo em 50) como apresentado na Seção 6.1.1.

Referências

AALST, W. M. P. van der. **Timed coloured Petri nets and their application to logistics**. Dissertação (Mestrado) — Technische Universiteit Eidhoven, 1992.

AALST, W. M. P. van der. The application of petri nets to workflow management. **Journal of Circuits Systems and Computers**, v. 8, p. 21 – 66, 1998.

_____. Modeling and analyzing interorganizational workflows. In: **International Conference on Application of Concurrency to System Design**. Fukushima, Japão: IEEE Computer Society, 1998. p. 262 – 272.

AALST, W. M. P. van der et al. Chapter 1: Introduction. In: **Modern Business Process Automation**. Berlim, Alemanha: Springer Berlin Heidelberg, 2010. p. 3 – 19.

AALST, W. M. P. van der; ADRIANSYAH, A.; DONGEN, B. van. Replaying history on process models for conformance checking and performance analysis. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 2, p. 182 – 192, 2012.

AALST, W. M. P. van der et al. Choreography conformance checking: An approach based on BPEL and petri nets. In: **The Role of Business Processes in Service Oriented Architectures**. Wadern, Alemanha: Internationales Begegnungs- und Forschungszentrum fuer Informatik, 2006. v. 06291.

_____. Soundness of workflow nets: classification, decidability, and analysis. **Formal Aspects of Computing**, v. 23, p. 333 – 363, 2011.

AALST, W. M. P. van der; HEE, K. v. **Workflow Management: Models, Methods, and Systems**. Cambridge, EUA: MIT Press, 2004.

AALST, W. M. P. van der et al. Workflow patterns. **Distributed and Parallel Databases**, v. 14, p. 5 – 51, 2003.

AALST, W. M. P. van der; HOFSTEDE, A. H. M. ter. Yawl: Yet another workflow language. **Information Systems**, v. 30, p. 245 – 275, 2005.

AALST, W. M. P. van der; JABLONSKI, S. Dealing with workflow change: Identification of issues and solutions. **International Journal of Computer Systems, Science, and Engineering**, v. 15, p. 267 – 276, 2000.

AALST, W. M. P. van der; PESIC, M.; SCHONENBERG, H. Declarative workflows: Balancing between flexibility and support. **Computer Science - Research and Development**, v. 23, p. 99 – 113, 2009.

AALST, W. M. P. van der; WESKE, M.; GRÜNBAUER, D. Case handling: a new paradigm for business process support. **Data & Knowledge Engineering**, v. 53, p. 129 – 162, 2005.

AALST, W. van der. **Process Mining**. Berlim, Alemanha: Springer Berlin Heidelberg, 2016.

ADAMS, M. **Facilitating Dynamic Flexibility and Exception Handling for Workflows**. Tese (Doutorado) — Queensland University of Technology, Brisbane, Austrália, 2007.

ADAMS, M. Chapter 4: Dynamic workflow. In: **Modern Business Process Automation**. Berlim, Alemanha: Springer Berlin Heidelberg, 2010. p. 123 – 145.

ADAMS, M. et al. Dynamic, extensible and context-aware exception handling for workflows. In: **On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS**. Vilamoura, Portugal: Springer Berlin Heidelberg, 2007. v. 4803, p. 95 – 112.

_____. Facilitating flexibility and dynamic exception handling in workflows through worklets. In: **CAiSE Short Paper Proceedings**. Porto, Portugal: CEUR-WS.org, 2005. v. 161, p. 45 – 50.

_____. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: **Proceedings of the 2006 Confederated International Conference on the Move to Meaningful Internet Systems**. Montpellier, França: Springer-Verlag, 2006. v. 4275, p. 291 – 308.

ADRIANSYAH, A.; DONGEN, B. F. van; AALST, W. M. P. van der. Towards robust conformance checking. In: **Business Process Management Workshops**. Berlim, Alemanha: Springer Berlin Heidelberg, 2010. v. 66, p. 122 – 133.

AFFLERBACH, P. et al. The business value of process flexibility. **Business & Information Systems Engineering**, v. 6, p. 203 – 214, 2014.

AHMAD, F.; HUANG, H.; WANG, X. Analysis of the petri net model of parallel manufacturing processes with shared resources. **Information Sciences**, v. 181, p. 5249 – 5266, 2011.

ALONSO, G. et al. Functionality and limitations of current workflow management systems. **IEEE Expert**, v. 12, 1997.

_____. Enhancing the fault tolerance of workflow management systems. **IEEE Concurrency**, v. 8, p. 74 – 81, 2000.

ARAKI, T.; KASAMI, T. Some decision problems related to the reachability problem for petri nets. **Theoretical Computer Science**, v. 3, p. 85 – 104, 1976.

_____. Some decision problems related to the reachability problem for petri nets. **Theoretical Computer Science**, v. 3, p. 85 – 104, 1976.

ASATO, O. L. et al. Process control system considering the machines functional flexibilities. In: **Technological Innovation for Value Creation**. Berlim, Alemanha: Springer Berlin Heidelberg, 2012. p. 133 – 142.

_____. Control of productive systems with functional flexibility level. In: **Symposium on Emerging Technologies and Factory Automation**. Toulouse, França: IEEE Computer Society, 2011. p. 1 – 4.

AWAD, A.; PUHLMANN, F. Structural detection of deadlocks in business process models. In: **Business Information Systems**. Innsbruck, Áustria: Springer Berlin Heidelberg, 2008. v. 7, p. 239 – 250.

BACALÁ JÚNIOR, S. **Arquitetura de software baseada numa abordagem UML/Redes de Petri com prevenção de bloqueio mortal em Sistemas de Tempo Real**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2003.

BAEK, S.-J.; HAN, J.-S.; CHUNG, K.-Y. Dynamic reconfiguration based on goal-scenario by adaptation strategy. **Wireless Pers Commun**, v. 73, p. 309 – 318, 2013.

BALESTRIN, A.; VERSCHOORE, J. R.; JUNIOR, E. R. O campo de estudo sobre redes de cooperação interorganizacional no brasil. **Revista de Administração Contemporânea**, v. 14, p. 458 – 477, 2010.

BANASZAK, Z.; KROGH, B. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. **IEEE Transactions on Robotics and Automation**, v. 6, p. 724 – 734, 1990.

BARKAOUI, K.; ABDALLAH, I. Deadlock avoidance in fms based on structural theory of petri nets. In: **Symposium on Emerging Technologies and Factory Automation**. Paris, França: IEEE Computer Society, 1995. v. 2, p. 499 – 510.

BARROS, A. P.; DECKER, G. Dynamic routing as paradigm for decentralized flexible process management. In: **EDOC Workshops**. Hong Kong, China: 10th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2006. p. 27.

BARUWA, O. T.; PIERA, M. A.; GUASCH, A. Deadlock-free scheduling method for flexible manufacturing systems based on timed colored petri nets and anytime heuristic search. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, p. 831 – 846, 2015.

BASIN, D. A. et al. Monpoly: Monitoring usage-control policies. In: **2nd International Conference on Runtime Verification**. São Francisco, EUA: Springer, 2011. v. 7186, p. 360 – 364.

BÉRARD, B. et al. The expressive power of time petri nets. **Theoretical Computer Science**, Elsevier BV, v. 474, p. 1 – 20, 2013.

BERTHOMIEU, B.; MENASCHE, M. An enumerative approach for analyzing time petri nets. In: **Proceedings of the International Federation for Information Processing**. Londres, Reino Unido: Elsevier Science Publishers, 1983. p. 41 – 46.

BEZERRA, E. **Princípios De Análise e Projeto De Sistemas Com {UML}**. Rio de Janeiro, Brasil: Elsevier Editora Ltda., 2007.

BHATT, G. Chapter 5: The role of dynamic organizational capabilities in creating, renewing, and leveraging information systems competencies. In: **Planning for Information Systems**. Londres, Reino Unido: Routledge, 2015. p. 96 – 107.

BOER, E. R.; MURATA, T. Generating basis siphons and traps of petri nets using the sign incidence matrix. **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications**, v. 41, p. 266 – 271, 1994.

BORREGO, D.; BARBA, I. Conformance checking and diagnosis for declarative business process models in data-aware scenarios. **Expert Systems with Applications**, v. 41, p. 5340 – 5352, 2014.

BOWDEN, F. A brief survey and synthesis of the roles of time in petri nets. **Mathematical and Computer Modelling**, Elsevier BV, v. 31, p. 55 – 68, 2000.

BUIJS, J. C. A. M. et al. Improving business process models using observed behavior. In: **International Symposium on Data-driven Process Discovery and Analysis**. Campione d'Italia, Itália: Springer Berlin Heidelberg, 2012. v. 162, p. 44 – 59.

CANTAMESSA, M.; CAPELLO, C. Flexibility in manufacturing - an empirical case-study research. In: **Design of Flexible Production Systems**. Berlim, Alemanha: Springer Berlin Heidelberg, 2009. p. 19 – 40.

CANTÚ, E. **Uma Abordagem para a Representação, Simulação e Implementação de Sistemas Baseada na Rede de Petri a Objetos**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 1990.

CARDOSO, J. Time fuzzy petri nets. In: **Studies in Fuzziness and Soft Computing**. Berlim, Alemanha: Physica-Verlag, 1999. v. 22, p. 115 – 145.

CARDOSO, J.; VALETTE, R. **Redes de Petri**. Florianópolis, Brazil: Editora da Petri, 1997. Disponível em: <http://valetterobert.free.fr/enseignement.d/livroweb101004.pdf>.

CARDOSO, J.; VALETTE, R.; DUBOIS, D. Monitoring manufacturing systems by means of petri nets with imprecise markings. **IEEE International Symposium on Intelligent Control**, p. 233 – 238, 1989.

_____. Petri nets with uncertain markings. In: **Advances in Petri Nets 1990**. Berlim, Alemanha: Springer Berlin Heidelberg, 1991. v. 483, p. 64 – 78.

_____. Possibilistic petri nets. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics**, v. 29, p. 573 – 582, 1999.

CARLSEN, S.; KROGSTIE, J.; LINDLAND, O. I. Evaluating flexible workflow systems. In: **Proceedings of the 30th Hawaii International Conference on System Sciences**. Wailea-Makena, Havaí: IEEE Computer Society, 1997. v. 2, p. 230 – 239.

CELKO, J. Time token design methodology. **Software: Practice and Experience**, Wiley-Blackwell, v. 12, p. 889 – 895, 1982.

CERONE, A.; MAGGIOLO-SCHETTINI, A. Time-based expressivity of time petri nets for system specification. **Theoretical Computer Science**, v. 216, p. 1 – 53, 1999.

CHANOPAS, A.; KRAIRIT, D.; KHANG, D. B. Managing information technology infrastructure: a new flexibility framework. **Management Research News**, v. 29, p. 632 – 651, 2006.

CHARFI, A.; MEZINI, M. Ao4bpel: An aspect-oriented extension to bpel. **World Wide Web**, v. 10, p. 309 – 344, 2007.

CHAROY, F.; GUABTNI, A.; FAURA, M. V. A dynamic workflow management system for coordination of cooperative activities. In: **Business Process Management Workshops**. Viena, Áustria: Springer, 2006. v. 4103, p. 205 – 216.

CHEN, Y.; LI, Z. Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems. **Automatica**, v. 47, p. 1028 – 1034, 2011.

CHEN, Y.; LI, Z.; ZHOU, M. Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 42, p. 615 – 629, 2012.

_____. Optimal supervisory control of flexible manufacturing systems by petri nets: A set classification approach. **IEEE Transactions on Automation Science and Engineering**, v. 11, p. 549 – 563, 2014.

CHU, F.; XIE, X.-L. Deadlock analysis of petri nets using siphons and mathematical programming. **IEEE Transactions on Robotics and Automation**, v. 13, p. 793 – 804, 1997.

CHUU, S.-J. Evaluating the flexibility in a manufacturing system using fuzzy multi-attribute group decision-making with multi-granularity linguistic information. **The International Journal of Advanced Manufacturing Technology**, v. 32, p. 409 – 421, 2006.

CÎMPAN, S.; OQUENDO, F. Dealing with software process deviations using fuzzy logic based monitoring. **ACM SIGAPP Applied Computing Review**, v. 8, p. 3 – 13, 2000.

COFFMAN, E. G.; ELPHICK, M.; SHOSHANI, A. System deadlocks. **ACM Computing Surveys**, v. 3, p. 67 – 78, 1971.

CORDONE, R.; PIRODDI, L. Parsimonious monitor control of petri net models of flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 43, p. 215 – 221, 2013.

CUGOLA, G. et al. A framework for formalizing inconsistencies and deviations in human-centered systems. **ACM Transactions on Software Engineering and Methodology**, v. 5, p. 191 – 230, 1996.

_____. How to deal with deviations during process model enactment. In: **Proceedings of the 17th International Conference on Software Engineering**. Seattle, EUA: ACM, 1995. p. 265 – 273.

DADAM, P.; REICHERT, M. The adept project: a decade of research and development for robust and flexible process support. **Computer Science - Research and Development**, v. 23, p. 81 – 97, 2009.

DAS, A.; CAPRIHAN, R. A rule-based fuzzy-logic approach for the measurement of manufacturing flexibility. **The International Journal of Advanced Manufacturing Technology**, v. 38, p. 1098 – 1113, 2007.

DAVID, R.; ALLA, H. **Discrete, Continuous, and Hybrid Petri Nets**. 2nd. ed. Berlim, Alemanha: Springer-Verlag Berlin Heidelberg, 2010.

DEHNERT, J.; RITTGEN, P. Relaxed soundness of business processes. In: **Advanced Information Systems Engineering**. Berlim, Alemanha: Springer Berlin Heidelberg, 2001. v. 2068, p. 157 – 170.

DESEL, J.; ERWIN, T. Modeling, simulation and analysis of business processes. In: **Business Process Management, Models, Techniques, and Empirical Studies**. Londres, Reino Unido: Springer-Verlag, 2000. p. 129 – 141.

DIAZ, M. **Petri Nets: Fundamental Models, Verification and Applications**. Hoboken, EUA: Wiley-IEEE Press, 2009.

DINGLE, N. J.; KNOTTENBELT, W. J.; SUTO, T. Pipe2: A tool for the performance evaluation of generalised stochastic petri nets. **SIGMETRICS Performance Evaluation Review**, v. 36, p. 34 – 39, 2009.

DOANE, D.; SEWARD, L. **Estatística Aplicada à Administração e à Economia**. São Paulo, Brasil: Cengage, 2011.

DOTOLI, M. et al. A timed petri nets model for intermodal freight transport terminals. **Proceedings of the International Federation of Automatic Control**, Elsevier BV, v. 47, p. 176 – 181, 2014.

DUBOIS, D.; PRADE, H. **Théorie des possibilités: applications à la représentation des connaissances en informatique**. Paris, França: Masson, 1985. (Méthode + programmes).

_____. Possibility theory. In: **Encyclopedia of Complexity and Systems Science**. Nova Iorque, EUA: Springer New York, 2009. p. 6927 – 6939.

DUFOURD, C.; FINKEL, A.; SCHNOEBELEN, P. Reset nets between decidability and undecidability. In: **Proceedings of the 25th International Colloquium on Automata, Languages and Programming**. Aalborg, Dinamarca: Springer Science + Business Media, 1998. v. 1443, p. 103 – 115.

DUFOURD, C.; JANČAR, P.; SCHNOEBELEN, Ph. Boundedness of reset P/T nets. In: **Proceedings of the 26th International Colloquium on Automata, Languages and Programming**. Prague, Czech Republic: Springer, 1999. v. 1644, p. 301 – 310.

ELLIS, C.; KEDDARA, K.; ROZENBERG, G. Dynamic change within workflow systems. In: **Proceedings of Conference on Organizational computing systems**. Nova Iorque, EUA: ACM Press, 1995. p. 10 – 21.

ESHUIS, H. **Semantics and Verification of UML Activity Diagrams for Workflow Modelling**. Tese (Doutorado) — University of Twente, Enschede, Holanda, 2002.

EZPELETA, J.; COLOM, J. M.; MARTÍNEZ, J. A petri net based deadlock prevention policy for flexible manufacturing systems. **IEEE Transactions on Robotics and Automation**, v. 11, p. 173 – 184, 1995.

EZPELETA, J.; COUVREUR, J.-M.; SILVA, M. A new technique for finding a generating family of siphons, traps and st-components. application to colored petri nets. In: **International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets**. Londres, Reino Unido: Springer-Verlag, 1993. p. 126 – 147.

FAHLAND, D. et al. Analysis on demand: Instantaneous soundness checking of industrial business process models. **Data and Knowledge Engineering**, v. 70, p. 448 – 466, 2011.

FREITAS, J. C. J. de. **Modelagem e Simulação de Sistemas de Gerenciamento de Processos de Negócios baseadas em Workflow net Temporais com Mecanismos de Alocação de Recursos Híbridos Fuzzy**. Tese (Doutorado) — Universidade Federal de Uberlândia, Uberlândia, Brasil, 2017.

GANG, X.; MING, W. Z. Systemic solutions to deadlock in fms. In: **American Control Conference**. Boston, EUA: IEEE Computer Society, 2004. v. 6, p. 5740 – 5745.

GENRICH, H. J. Predicate/transition nets. In: **Proceedings of the Advances in Petri Nets, Part I on Petri Nets: Central Models and Their Properties**. Londres, Inglaterra: Springer-Verlag, 1987. p. 207 – 247.

GHARIB, M.; GIORGINI, P. Modeling and reasoning about information quality requirements in business processes. In: **Enterprise, Business-Process and Information Systems Modeling**. Estocolmo, Suécia: Springer International Publishing, 2015. p. 231 – 245.

GIBLIN, C.; MüLLER, S.; PFITZMANN, B. **From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation**. R'uschlikon, Suíça, 2006.

GOLDEN, W.; POWELL, P. Towards a definition of flexibility: in search of the holy grail? **Omega**, v. 28, p. 373 – 384, 2000.

GOMEZ-LOPEZ, M. T.; GASCA, R. M.; RINDERLE-MA, S. Explaining the incorrect temporal events during business process monitoring by means of compliance rules and model-based diagnosis. In: **17th IEEE International Enterprise Distributed Object Computing Conference Workshops**. Vancouver, Canadá: IEEE Computer Society, 2013. p. 163 – 172.

GORGÔNIO, K. C. **Adaptação de Modelos em Redes de Petri Coloridas**. Dissertação (Mestrado) — Universidade Federal da Paraíba, 2001.

GÜNSEL, A. et al. The role of flexibility on software development performance: An empirical study on software development teams. **Procedia - Social and Behavioral Sciences**, v. 58, p. 853 – 860, 2012.

HACK, M. **Analysis production schemata by Petri nets**. Dissertação (Mestrado) — Massachusetts Institute of Technology, 1972.

HALLE, S.; VILLEMAIRE, R. Runtime monitoring of message-based workflows with data. In: **12th International IEEE Enterprise Distributed Object Computing Conference**. Munique, Alemanha: IEEE Computer Society, 2008. p. 63 – 72.

HARTEL, P. H. et al. Declarative languages in education. In: **Encyclopaedia of Microcomputers**. Nova Iorque, EUA: Marcel Dekker Inc., 2001. v. 27, p. 97 – 102.

HEINL, P. et al. A comprehensive approach to flexibility in workflow management systems. ACM, São Francisco - EUA, p. 79 – 88, 1999.

HERLE, S. et al. Exploiting parallel processing and optimal paths in a flexible manufacturing system. In: **IEEE International Conference on Automation, Quality and Testing, Robotics**. Cluj-Napoca, Romênia: IEEE Computer Society, 2010. v. 3, p. 1 – 6.

HOFSTEDE, A. H. M. t. et al. (Ed.). **Modern Business Process Automation**. Berlim, Alemanha: Springer Berlin Heidelberg, 2010.

HOLLINGSWORTH, D. **The workflow reference model**. Bruxelas, Bélgica, 1994.

HSIEH, F.-S.; LIN, J.-B. A self-adaptation scheme for workflow management in multi-agent systems. **Journal of Intelligent Manufacturing**, v. 27, p. 1 – 18, 2013.

_____. Development of context-aware workflow systems based on petri net markup language. **Computer Standards & Interfaces**, v. 36, p. 672 – 685, 2014.

HU, H.; ZHOU, M. A petri net-based discrete-event control of automated manufacturing systems with assembly operations. **IEEE Transactions on Control Systems Technology**, v. 23, p. 513 – 524, 2015.

HUANG, Y. et al. Deadlock prevention policy based on petri nets and siphons. **International Journal of Production Research**, v. 39, p. 283 – 305, 2001.

HUANG, Y.-S.; PAN, Y.-L.; ZHOU, M. Computationally improved optimal deadlock control policy for flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 42, p. 404 – 415, 2012.

IORDACHE, M.; MOODY, J.; ANTSAKLIS, P. Synthesis of deadlock prevention supervisors using petri nets. **IEEE Transactions on Robotics and Automation**, v. 18, p. 59 – 68, 2002.

JALOTE, P. Fault tolerant processes. **Distributed Computing**, v. 3, p. 187 – 195, 1989.

JENG, M. der; PENG, M. Y. Generating minimal siphons and traps for petri nets. In: **IEEE International Conference on Systems, Man, and Cybernetics**. Pequim, Japão: IEEE Computer Society, 1996. v. 4, p. 2996 – 2999.

JENSEN, K. Coloured petri nets and the invariant-method. **Theoretical Computer Science**, v. 14, p. 317 – 336, 1981.

JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri Nets**. Berlim, Alemanha: Springer Berlin Heidelberg, 2009.

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**, v. 9, p. 213 – 254, 2007.

JESKE, J. C.; JULIA, S.; VALETTE, R. Fuzzy continuous resource allocation mechanisms in workflow management systems. **Brazilian Symposium on Software Engineering**, v. 0, p. 236 – 251, 2009.

JONES, P.; L., S.; WADLER, P. Imperative functional programming. In: **Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages**. Nova Iorque, EUA: ACM, 1993. p. 71 – 84.

KAPURUGE, M.; HAN, J.; COLMAN, A. Support for business process flexibility in service compositions: An evaluative survey. In: **Australian Software Engineering Conference**. Auckland, Nova Zelândia: IEEE Computer Society, 2010. p. 97 – 106.

KHALID, M. N. A.; YUSOF, U. K.; SABUDIN, M. Solving flexible manufacturing system distributed scheduling problem subject to maintenance using harmony search algorithm. In: **4th Conference on Data Mining and Optimization**. Langkawi, Malásia: IEEE Computer Society, 2012. p. 73 – 79.

KHOMYAKOV, M.; BIDER, I. Achieving workflow flexibility through taming the chaos. In: **Proceedings of the 6th International Conference on Object Oriented Information Systems**. Londres, Reino Unido: Springer London, 2001. p. 85 – 92.

KINUYAMA M. E MURATA, T. Generating siphons and traps by petri nets representation of logic equations. In: **Proceedings of the Conference of the Net Theory**. Osaka, Japão: IECE, 1986. p. 93 – 100.

KOHLER, M.; SCHAAD, A. Avoiding policy-based deadlocks in business processes. In: **International Conference on Availability, Reliability and Security**. Barcelona, Espanha: IEEE Computer Society, 2008. p. 709 – 716.

KOTB, Y. T.; BADREDDIN, E. Synchronization among activities in a workflow using extended workflow petri nets. In: **Proceedings of the 7th IEEE International Conference on E-Commerce Technology**. Hong Kong, China: IEEE Computer Society, 2005. p. 548 – 551.

KUMAR, A.; YAO, W. Design and management of flexible process variants using templates and rules. **Computers in Industry**, v. 63, p. 112 – 130, 2012.

KUMAR, K.; NARASIPURAM, M. M. Defining requirements for business process flexibility. In: **Proceedings of the Workshop on Business Process Modelling, Development, and Support**. Cidade de Luxemburgo, Luxemburgo: CAISE'06, 2006. v. 236, p. 137 – 148.

LAHAYE, S.; KOMENDA, J.; BOIMOND, J.-L. Modeling of timed petri nets using deterministic (max, +) automata. **Proceedings of the International Federation of Automatic Control**, Elsevier BV, v. 47, p. 471 – 476, 2014.

LEONI, M. de; AALST, W. M. P. van der. Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: **Lecture Notes in Computer Science**. Berlim, Heidelberg: Springer Science + Business Media, 2013. p. 113 – 129.

LI, Z. et al. Deadlock prevention based on structure reuse of petri net supervisors for flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 42, p. 178 – 191, 2012.

LI, Z.; ZHOU, M. Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 34, p. 38 – 51, 2004.

LIU, D.; LI, Z.; ZHOU, M. Hybrid liveness-enforcing policy for generalized petri net models of flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 43, p. 85 – 97, 2013.

LIU, G. et al. Robustness of deadlock control for a class of petri nets with unreliable resources. **Information Sciences**, v. 235, p. 259 – 279, 2013.

LIU, H. et al. Deadlock prevention for flexible manufacturing systems via controllable siphon basis of petri nets. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, p. 519 – 529, 2015.

LOH, S.; CASTILHO, J. M. V. de. O uso de uma linguagem semi-formal no processo de formalização de especificacoes de software. **Simpósio Brasileiro de Engenharia de Software**, v. 5, p. 207 – 220, 1991.

LOMAZOVA, I. A. Interacting workflow nets for workflow process re-engineering. **Fundamenta Informaticae**, v. 101, p. 59 – 70, 2010.

LOONEY, C. G. Fuzzy petri nets for rule-based decisionmaking. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 18, p. 178 – 183, 1988.

LUO, J. et al. Deadlock-free scheduling of automated manufacturing systems using petri nets and hybrid heuristic search. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, p. 530 – 541, 2015.

LY, L. T. et al. Compliance monitoring in business processes: Functionalities, application, and tool-support. **Information Systems**, v. 54, p. 209 – 234, 2015.

MAHDAVI, I.; AZAR, A. H. F. M.; BAGHERPOUR, M. Applying fuzzy rule based to flexible routing problem in a flexible manufacturing system. In: **IEEE International Conference on Industrial Engineering and Engineering Management**. Hong Kong, China: IEEE Computer Society, 2009. p. 2358 – 2364.

MAHMUDY, W. F.; MARIAN, R. M.; LUONG, L. H. S. Hybrid genetic algorithms for part type selection and machine loading problems with alternative production plans in flexible manufacturing system. **ECTI Transactions on Computer and Information Technology**, v. 8, p. 80 – 93, 2014.

MARUTA, T. et al. A deadlock detection algorithm for business processes workflow models. In: **IEEE International Conference on Systems, Man, and Cybernetics**. San Diego, Califórnia: IEEE Computer Society, 1998. v. 1, p. 611 – 616.

MEMBERS, W. M. C. **Glossary: A Workflow Management Coalition Specification**. Bruxelas, Bélgica, 1994.

MERLIN, P. M. **A Study of the Recoverability of Computing Systems**. Tese (Doutorado) — University of California, Irvine, EUA, 1974.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine Learning: An Artificial Intelligence Approach**. Berlim, Alemanha: Springer Publishing Company, Incorporated, 2013.

MILNER, R.; TOFTE, M.; MACQUEEN, D. **The Definition of Standard ML**. Cambridge, EUA: MIT Press, 1997.

MISHRA, R.; PUNDIR, A. K.; GANAPATHY, L. Manufacturing flexibility research: A review of literature and agenda for future research. **Global Journal of Flexible Systems Management**, v. 15, p. 101 – 112, 2014.

MOHAMMED, K.; REDOUANE, L.; BERNARD, C. A deviation-tolerant approach to software process evolution. In: **9th International Workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting**. Dubrovnik, Croácia: ACM, 2007. p. 75 – 78.

MOHANTY, M.; KUMARA, P. Deadlock prevention in process control computer system. **IJCA Proceedings on International Conference in Distributed Computing and Internet Technology**, ICDCIT, p. 12 – 16, 2013.

MUNOZ-GAMA, J. **Algorithms for Process Conformance and Process Refinement**. Dissertação (Mestrado) — Universitat Politècnica de Catalunya (UPC), Barcelona, Espanha, 2010.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, v. 77, p. 541 – 580, 1989.

_____. Temporal uncertainty and fuzzy-timing high-level petri nets. In: **Proceedings of International Conference Application and Theory of Petri Nets**. Osaka, Japão: Springer Berlin / Heidelberg, 1996. v. 1091, p. 11 – 28.

MURATA, T.; SUZUKI, T.; SHATZ, S. Fuzzy-timing high-level petri nets (fthns) for time-critical systems. In: **Fuzziness in Petri Nets**. Nova Iorque, EUA: Springer Verlag, 1999. v. 22, p. 88 – 114.

OMG. **Business Process Model and Notation (BPMN), Version 2.0**. [S.l.], 2011. Disponível em: <http://www.omg.org/spec/BPMN/2.0>.

_____. **OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1**. [S.l.], 2011. Disponível em: <http://www.omg.org/spec/UML/2.4.1>.

PADUA, S. I. D. et al. O potencial das redes de petri em modelagem e análise de processos de negócio. **Gestão & Produção**, v. 11, p. 109 – 119, 2004.

PASSOS, L. M. S.; JULIA, S. Qualitative analysis of workflow nets using linear logic: Soundness verification. In: **IEEE International Conference on Systems, Man and Cybernetics**. Herndon, EUA: IEEE Computer Society, 2009. p. 2843 – 2847.

PASSOS, L. M. S.; JULIA, S. Linear logic as a tool for deadlock-freeness scenarios detection in interorganizational workflow processes. In: **26th International Conference on Tools with Artificial Intelligence**. Limassol, Chipre: IEEE Computer Society, 2014. p. 316 – 320.

PESIC, M. **Constraint-based Workflow Management Systems: Shifting Control to Users**. Tese (Doutorado) — Eindhoven University of Technology, Eindhoven, Países Baixos, 2008.

PESIC, M.; AALST, W. M. P. van der. A declarative approach for flexible business processes management. In: **Proceedings of the 2006 International Conference on Business Process Management Workshop**. Viena, Áustria: Springer Berlin Heidelberg, 2006. v. 4103, p. 169 – 180.

PESIC, M. et al. Constraint-based workflow models: Change made easy. In: **On the Move to Meaningful Internet Systems**. Vilamoura, Portugal: Springer Berlin Heidelberg, 2007. v. 4803, p. 77 – 94.

PETERSON, J. **Petri Net Theory and the Modelling of Systems**. Nova Jersey, EUA: Prentice Hall, 1983.

PETRI, C. A. **Kommunikation mit Automaten**. Tese (Doutorado) — Institut für Instrumentelle Mathematik, Bonn, Alemanha, 1962.

PLA, A. et al. Petri net-based process monitoring: a workflow management system for process modelling and monitoring. **Journal of Intelligent Manufacturing**, v. 25, p. 539 – 554, 2012.

PUNDIR, A. K.; MISHRA, R.; GANAPATHY, L. An exploration of firm level competitiveness through choices in manufacturing strategy: The case of indian four wheeler passenger vehicle companies. **EuroEconomica**, v. 32, p. 49 – 61, 2013.

QI, Q.; LUO, G. Constructing enterprise flexibility competence on the basis of product platform and collaborative product commerce. In: **International Conference on Automation and Logistics**. Jinan, China: IEEE Computer Society, 2007. p. 2857 – 2861.

RAMCHANDANI, C. **Analysis of Asynchronous Concurrent Systems by Timed Petri Nets**. Cambridge, EUA, 1974.

REZENDE, L. P.; JULIA, S.; CARDOSO, J. Possibilistic workflow nets to deal with non-conformance in process execution. In: **International Conference on Systems, Man, and Cybernetics**. Seul, Korea: IEEE Computer Society, 2012. p. 1219 – 1224.

REZENDE, L. P. de. **WorkFlow net Possibilística para Problemas de não Conformidade em Processos de Negócios**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2013.

REZENDE, L. P. de; JULIA, S. Deadlock avoidance in interorganizational business processes using a possibilistic workflow net. In: **Proceedings of the 17th International Conference on Enterprise Information Systems**. Barcelona, Espanha: SciTePress, 2015. v. 1, p. 429 – 439.

_____. Possibilistic workflow net for deadlock avoidance in interorganizational business processes. In: **Enterprise Information Systems**. Cham, Suíça: Springer International Publishing, 2015. v. 241, p. 168 – 191.

REZENDE, L. P. de; JULIA, S.; CARDOSO, J. Inconsistency recovery in business processes using a possibilistic workflow net. In: **International Conference of the Chilean Computer Science Society**. Valparaíso, Chile: IEEE Computer Society, 2012. p. 41 – 50.

_____. Possibilistic interorganizational workflow net for the recovery problem concerning communication failures. In: **Proceedings of the 16th International Conference on Enterprise Information Systems**. Lisboa, Portugal: SciTePress, 2014. v. 1, p. 432 – 439.

_____. Possibilistic workflow nets for dealing with cancellation regions in business processes. In: **Proceedings of the 18th International Conference on Enterprise Information Systems**. Roma, Itália: SciTePress, 2016. v. 2, p. 126 – 133.

_____. Uncertain marking for dealing with partial parallelization in business processes. In: **Proceedings of the 18th International Conference on Enterprise Information Systems**. Roma, Itália: SciTePress, 2016. v. 2, p. 118 – 125.

RILLO, M. **Aplicações de Redes de Petri em sistemas de Manufatura**. Tese (Doutorado) — Escola Politénica da Universidade de São Paulo, EPUSP, São Paulo, Brazil, 1988.

RINDERLE, S.; REICHERT, M.; DADAM, P. Correctness criteria for dynamic changes in workflow systems: a survey. **Data & Knowledge Engineering**, v. 50, p. 9 – 34, 2004.

ROSA, M. L. et al. **Business process variability modeling: A survey**. Brisbane, Austrália, 2013.

ROZINAT, A.; AALST, W. M. P. van der. Conformance checking of processes based on monitoring real behavior. **Information Systems**, v. 33, p. 64 – 95, 2008.

SADIQ, W.; ORLOWSKA, M. E. Analyzing process models using graph reduction techniques. **Information Systems**, v. 25, p. 117 – 134, 2000.

SANTOS, E. A. P. et al. Modeling business rules for supervisory control of process-aware information systems. In: **International Business Process Management Workshops**. Clermont-Ferrand, França: Springer Berlin Heidelberg, 2012. p. 447 – 458.

SCHEFFER, T. Algebraic foundation and improved methods of induction of ripple down rules. In: **Proceedings of the 2nd Pacific Rim Workshop on Knowledge Acquisition**. Sydney, Austrália: CiteSeerX, 1996. p. 279 – 292.

SCHONENBERG, H. et al. Process flexibility: A survey of contemporary approaches. In: **Advances in Enterprise Engineering I**. Montpellier, França: Springer-Verlag Berlin, 2008. v. 10, p. 16 – 30.

_____. Towards a taxonomy of process flexibility. In: **Proceedings of the Forum at the CAiSE'08 conference**. Montpellier, França: CEUR Workshop Proceedings, 2008. v. 344, p. 81 – 84.

SEBAHI, S. **Monitoring business process compliance: a view based approach**. Tese (Doutorado) — Université Claude Bernard - Lyon I, Lyon, França, 2012.

SHARP, A.; MCDERMOTT, P. **Workflow Modeling: Tools for Process Improvement and Application Development**. 1st. ed. Norwood, EUA: Artech House, Inc., 2001.

SIBERTIN-BLANC, C. Cooperative objects: Principles, use and implementation. In: **Concurrent Object-Oriented Programming and Petri Nets**. Berlim, Alemanha: Springer Berlin Heidelberg, 2001. p. 216 – 246.

SILVA, L. de F. et al. Siphon-based deadlock prevention policy for interorganizational workflow net design. In: **IEEE International Conference on Information Reuse and Integration**. São Francisco, EUA: IEEE Computer Society, 2013. p. 293 – 300.

SILVA, L. F. **Detecção e Correção de Situações de *Deadlock* em *WorkFlow nets* Interorganizacionais**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, Uberlândia, Brasil, 2014.

SILVA, M. A. A. da et al. Early deviation detection in modeling activities of mde processes. In: **Proceedings of the 13th international conference on Model driven engineering languages and systems: Part II**. Berlim, Alemanha: Springer Berlin Heidelberg, 2010. p. 303 – 317.

SILVA, R. M. da et al. A method to design a manufacturing control system considering flexible reconfiguration. In: **IEEE International Conference on Industrial Informatics**. Porto Alegre, Brazil: IEEE Computer Society, 2014. p. 82 – 87.

SINGH, D.; OBEROI, J. S.; AHUJA, I. S. Assessing strategic flexibility of manufacturing organisations using AHP. **International Journal of Agile Systems and Management**, v. 5, p. 319 – 329, 2012.

SMANCHAT, S.; LING, S.; INDRAWAN, M. A survey on context-aware workflow adaptations. In: **Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia**. Nova Iorque, EUA: ACM, 2008. p. 414 – 417.

SNOWDON, R. A. et al. On the architecture and form of flexible process support. **Software Process: Improvement and Practice**, v. 12, p. 21 – 34, 2007.

SOFFER, P. On the notion of flexibility in business processes. In: **Proceedings of the Workshop on Business Process Modeling, Design and Support**. Porto, Portugal: CAiSE'05, 2005. v. 5, p. 35 – 42.

TANENBAUM, A. **Sistemas operacionais modernos**. 3. ed. São Paulo, Brasil: Pearson, 2009.

TANG, F. et al. An efficient deadlock prevention approach for service oriented transaction processing. **Computers & Mathematics with Applications**, v. 63, p. 458 – 468, 2012.

THOMPSON, S.; TORABI, T. An observational approach to practical process non-conformance detection. In: **Second International Conference on the Applications of Digital Information and Web Technologies**. Londres, Reino Unido: Institute of Electrical & Electronics Engineers (IEEE), 2009. p. 62 – 67.

THULLNER, R. et al. Proactive business process compliance monitoring with event-based systems. In: **15th International Enterprise Distributed Object Computing Conference Workshops**. Helsínquia, Finlândia: IEEE Computer Society, 2011. p. 429 – 437.

TOLIO, T. (Ed.). **Design of Flexible Production Systems**. Berlim, Alemanha: Springer Berlin Heidelberg, 2009.

TOMIYAMA, M. N. **Simulação e Modelagem de Processos Biológicos usando Redes de Petri Predicado Transição Diferenciais**. Dissertação (Mestrado) — Universidade Federal de Uberlândia, Uberlândia, Brasil, 2007.

TRICAS, F.; MARTÍNEZ, J. An extension of the liveness theory for concurrent sequential process competing for shared resources. In: **Proceedings of the IEEE International Conference on Systems, Man and Cybernetics**. Vancouver, Canada: IEEE Computer Society, 1995. v. 4, p. 4119 – 4124.

TüYSüZ, F.; KAHRAMAN, C. Modeling a flexible manufacturing cell using stochastic petri nets with fuzzy parameters. **Expert Systems with Applications**, Elsevier BV, v. 37, p. 3910 – 3920, 2010.

URTASUN-ALONSO, A. et al. Manufacturing flexibility and advanced human resource management practices. **Production Planning & Control**, v. 25, p. 303 – 317, 2012.

UZAM, M.; ZHOU, M. An iterative synthesis approach to petri net-based deadlock prevention policy for flexible manufacturing systems. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 37, p. 362 – 371, 2007.

VALETTE, R. Analysis of petri nets by stepwise refinements. **Journal of Computer and System Sciences**, v. 18, p. 35 – 46, 1979.

VALETTE, R.; ATABAKHCHE, H. Petri nets for sequence constraint propagation in knowledge based approaches. In: **Concurrency and Nets: Advances in Petri Nets**. Berlim, Alemanha: Springer Berlin Heidelberg, 1987. p. 555 – 569.

VALETTE, R.; COURVOISIER, M.; MAYEUX, D. Control of flexible production systems and petri nets. In: **European Workshop on Applications and Theory of Petri Nets**. Varenna, Itália: Springer Berlin Heidelberg, 1982. p. 264 – 277.

VERBEEK, H.; AALST, W. M. P. van der; HOFSTEDE, A. H. M. t. Verifying workflows with cancellation regions and or-joins: An approach based on relaxed soundness and invariants. **The Computer Journal**, v. 50, p. 294 – 314, 2007.

VERBEEK, H. et al. Reduction rules for reset/inhibitor nets. **Journal of Computer and System Sciences**, v. 76, p. 125 – 143, 2010.

VILLANI, E. **Modelagem e análise de sistemas supervisórios híbridos**. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2004.

WANG, J.; TEPFENHART, W. M.; ROSCA, D. Emergency response workflow resource requirements modeling and analysis. **IEEE Transactions on Systems, Man, and Cybernetics, Part C**, v. 39, p. 270 – 283, 2009.

WANG, Y.; LU, P. DDS: A deadlock detection-based scheduling algorithm for workflow computations in HPC systems with storage constraints. **Parallel Computing**, v. 39, p. 291 – 305, 2013.

_____. Maximizing active storage resources with deadlock avoidance in workflow-based computations. **IEEE Transactions on Computers**, v. 62, p. 2210 – 2223, 2013.

WARD, P. T.; DURAY, R. Manufacturing strategy in context: environment, competitive strategy and manufacturing strategy. **Journal of Operations Management**, v. 18, p. 123 – 138, 2000.

WEBER, B.; REICHERT, M.; RINDERLE-MA, S. Change patterns and change support features - enhancing flexibility in process-aware information systems. **Data and Knowledge Engineering**, v. 66, p. 438 – 466, 2008.

WESKE, M. **Business Process Management: Concepts, Languages, Architectures**. Berlim, Alemanha: Springer-Verlag Berlin Heidelberg, 2007.

WYNN, M. et al. Soundness-preserving reduction rules for reset workflow nets. **Information Sciences**, v. 179, p. 769 – 790, 2009.

ZADEH, L. A. Fuzzy sets. **Information and Control**, v. 8, p. 338 – 353, 1965.

_____. Fuzzy logic. **IEEE Computer**, v. 21, p. 83 – 93, 1988.

_____. Fuzzy sets as a basis for a theory of possibility. **Fuzzy Sets and Systems**, v. 100, Supplement 1, p. 9 – 34, 1999.

ZAIRI, S.; ZOUARI, B.; PITRAC, L. A formal approach for the specification, verification and control of flexible manufacturing systems. In: **IEEE Conference on Emerging Technologies & Factory Automation**. Patras, Grécia: IEEE Computer Society, 2007. p. 1031 – 1038.

ZAZWORKA, N.; BASILI, V. R.; SHULL, F. Tool supported detection and judgment of nonconformance in process execution. In: **3rd International Symposium on Empirical Software Engineering and Measurement**. Lake Buena Vista, EUA: IEEE Computer Society, 2009. p. 312 – 323.

ZAZWORKA, N. et al. Are developers complying with the process: an xp study. In: **Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. Bolzano, Itália: ACM, 2010. p. 1 – 10.

ZHONG, C.; LI, Z. Petri net based deadlock prevention approach for flexible manufacturing systems. In: **Applications for Flexibility and Agility**. Hershey, EUA: IGI Global, 2011. p. 416 – 433.

ZISMAN, M. D. **Representation, Specification and Automation of Office Procedures**. Tese (Doutorado) — University of Pennsylvania, Pensilvânia, EUA, 1977.

ZUBEREK, W. Timed petri nets definitions, properties, and applications. **Microelectronics Reliability**, Elsevier BV, v. 31, p. 627 – 644, 1991.

Apêndices

APÊNDICE **A**

Funcionamento do Algoritmo para a Detecção dos Sifões

Para ilustrar o funcionamento do Algoritmo 2, a rede de Petri apresentada na Figura 117 será utilizada.

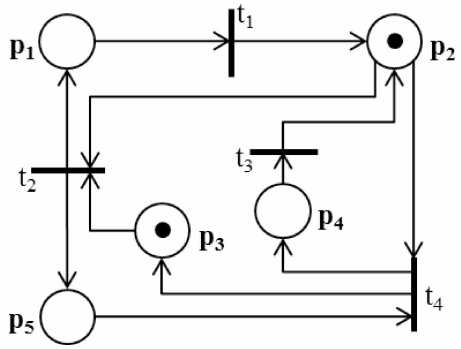


Figura 117 – Modelo representado por uma rede de Petri utilizado para ilustrar o funcionamento do Algoritmo 2.

A matriz de incidência de sinais D obtida a partir da rede de Petri apresentada na Figura 117 é dada por:

$$\mathrm{D} = \begin{array}{c} \begin{array}{ccccc} D_1 & D_2 & D_3 & D_4 & D_5 \end{array} \\ \begin{bmatrix} - & + & 0 & 0 & 0 \\ + & - & - & 0 & + \\ 0 & + & 0 & - & 0 \\ 0 & - & + & + & - \end{bmatrix} \end{array}$$

Inicialmente, a **1ª ETAPA** do Algoritmo 2 é executada até que todos os possíveis sifões ($LEAFs$) sejam identificados. Os passos executados nesta etapa são descritos abaixo:

Passo 1: Definindo as X_{jr} e P_{jr}:

$$D_1 = \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix}, \quad X_{11}^1 = \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} \quad \text{e} \quad P_{11}^1 = \{p_1\}$$

$$D_2 = \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix}, \quad X_{21}^1 = \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} \quad \text{e} \quad P_{21}^1 = \{p_2\}$$

$$D_3 = \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix}, \quad X_{31}^1 = \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} \quad \text{e} \quad P_{31}^1 = \{p_3\}$$

$$D_4 = \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix}, \quad X_{41}^1 = \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} \quad \text{e} \quad P_{41}^1 = \{p_4\}$$

$$D_5 = \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix}, \quad X_{51}^1 = \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} \quad \text{e} \quad P_{51}^1 = \{p_5\}$$

onde os índices de X e P indicam o $SEED_j$ (ou seja, os lugares p_j da rede) e o nível de recursão (neste caso, primeiro) e, os expoentes os diferentes vetores X ou conjuntos P de índices idênticos.

Passo 2: Neutralização dos valores + presentes nos X_{j1} definidos no "Passo 1":

- X_{11}^1: o valor + presente na segunda linha ($l = 2$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 3 (p_3) de D. Esta operação resulta em:

$$X_{12}^1 = X_{11}^1 \oplus D_2 = \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \quad P_{12}^1 = \{p_1, p_2\}$$

$$X_{12}^2 = X_{11}^1 \oplus D_3 = \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} = \begin{bmatrix} - \\ \pm \\ 0 \\ + \end{bmatrix} \quad P_{12}^2 = \{p_1, p_3\}$$

- X_{21}^1: o valor + presente na primeira linha ($l = 1$) é neutralizado através da adição da coluna 1 (p_1) de D. Esta operação resulta em:

$$X_{22}^1 = X_{21}^1 \oplus D_1 = \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} \oplus \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \quad P_{22}^1 = \{p_2, p_1\}$$

- X_{31}^1: o valor + presente na quarta linha ($l = 4$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 5 (p_5) de D. Esta operação resulta em:

$$X_{32}^1 = X_{31}^1 \oplus D_2 = \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} + \\ - \\ + \\ \pm \end{bmatrix} \quad P_{32}^1 = \{p_3, p_2\}$$

$$X_{32}^2 = X_{31}^1 \oplus D_5 = \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} \oplus \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} = \begin{bmatrix} 0 \\ \pm \\ 0 \\ \pm \end{bmatrix} \quad P_{32}^2 = \{p_3, p_5\}$$

- X_{41}^1: o valor + presente na quarta linha ($l = 4$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 5 (p_5) de D. Esta operação resulta em:

$$X_{42}^1 = X_{41}^1 \oplus D_2 = \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} + \\ - \\ \pm \\ \pm \end{bmatrix} \quad P_{42}^1 = \{p_4, p_2\}$$

$$X_{42}^2 = X_{41}^1 \oplus D_5 = \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} \oplus \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} = \begin{bmatrix} 0 \\ + \\ - \\ \pm \end{bmatrix} \quad P_{42}^2 = \{p_4, p_5\}$$

- X_{51}^1: o valor + presente na segunda linha ($l = 2$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 3 (p_3) de D. Esta operação resulta em:

$$X_{52}^1 = X_{51}^1 \oplus D_2 = \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} + \\ \pm \\ + \\ - \end{bmatrix} \quad p_{52}^1 = \{p_5, p_2\}$$

$$X_{52}^2 = X_{51}^1 \oplus D_3 = \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} \oplus \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} = \begin{bmatrix} 0 \\ \pm \\ 0 \\ \pm \end{bmatrix} \quad P_{52}^2 = \{p_5, p_3\}$$

Passo 3: Uma vez que alguns dos vetores X_{j2} produzidos no passo anterior contêm um valor + em uma de suas linhas, o "Passo 2" será executado novamente para eles.

Passo 2: Neutralização dos valores + presentes nos X_{j2}:

- X_{12}^1: o valor + presente na terceira linha ($l = 3$) é neutralizado através da adição da coluna 4 (p_4) de D. Esta operação resulta em:

$$X_{13}^1 = X_{12}^1 \oplus D_4 = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{13}^1 = \{p_1, p_2, p_4\}$$

- X_{12}^2: o valor + presente na quarta linha ($l = 4$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 5 (p_5) de D. Esta operação resulta em:

$$X_{13}^2 = X_{12}^2 \oplus D_2 = \begin{bmatrix} - \\ \pm \\ 0 \\ + \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ + \\ \pm \end{bmatrix} \quad P_{13}^2 = \{p_1, p_3, p_2\}$$

$$X_{13}^3 = X_{12}^2 \oplus D_5 = \begin{bmatrix} - \\ \pm \\ 0 \\ + \end{bmatrix} \oplus \begin{bmatrix} 0 \\ + \\ 0 \\ - \end{bmatrix} = \begin{bmatrix} - \\ \pm \\ 0 \\ \pm \end{bmatrix} \quad P_{13}^3 = \{p_1, p_3, p_5\}$$

- X_{22}^1: o valor + presente na terceira linha ($l = 3$) é neutralizado através da adição da coluna 4 (p_4) de D. Esta operação resulta em:

$$X_{23}^1 = X_{22}^1 \oplus D_4 = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{23}^1 = \{p_2, p_1, p_4\}$$

- X_{32}^1: o valor + presente na primeira linha ($l = 1$) é neutralizado através da adição da coluna 1 (p_1) de D. Esta operação resulta em:

$$X_{33}^1 = X_{32}^1 \oplus D_1 = \begin{bmatrix} + \\ - \\ + \\ \pm \end{bmatrix} \oplus \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ + \\ \pm \end{bmatrix} \quad P_{33}^1 = \{p_3, p_2, p_1\}$$

- X_{42}^1: o valor + presente na primeira linha ($l = 1$) é neutralizado através da adição da coluna 1 (p_1) de D. Esta operação resulta em:

$$X_{43}^1 = X_{42}^1 \oplus D_1 = \begin{bmatrix} + \\ - \\ \pm \\ \pm \end{bmatrix} \oplus \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{43}^1 = \{p_4, p_2, p_1\}$$

- X_{42}^2: o valor + presente na segunda linha ($l = 2$) é neutralizado através da adição da coluna 2 (p_2) ou da coluna 3 (p_3) de D. Esta operação resulta em:

$$X_{43}^2 = X_{42}^2 \oplus D_2 = \begin{bmatrix} 0 \\ + \\ - \\ \pm \end{bmatrix} \oplus \begin{bmatrix} + \\ - \\ + \\ - \end{bmatrix} = \begin{bmatrix} + \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{43}^2 = \{p_4, p_5, p_2\}$$

$$X_{43}^3 = X_{42}^2 \oplus D_3 = \begin{bmatrix} 0 \\ + \\ - \\ \pm \end{bmatrix} \oplus \begin{bmatrix} 0 \\ - \\ 0 \\ + \end{bmatrix} = \begin{bmatrix} 0 \\ \pm \\ - \\ \pm \end{bmatrix} \quad P_{43}^3 = \{p_4, p_5, p_3\}$$

- X_{52}^1: o valor + presente na primeira linha ($l = 1$) é neutralizado através da adição da coluna 1 (p_1) de D. Esta operação resulta em:

$$X_{53}^1 = X_{52}^1 \oplus D_1 = \begin{bmatrix} + \\ \pm \\ + \\ - \end{bmatrix} \oplus \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \quad P_{53}^1 = \{p_5, p_2, p_1\}$$

Passo 3: Uma vez que alguns dos vetores X_{j3} produzidos no passo anterior ainda contêm um valor + em uma de suas linhas, o "Passo 2" será executado novamente para eles.

Passo 2: Neutralização dos valores + presentes nos X_{j3}:

- X_{13}^2: o valor + presente na terceira linha ($l = 3$) é neutralizado através da adição da coluna 4 (p_4) de D. Esta operação resulta em:

$$X_{14}^{1} = X_{13}^{2} \oplus D_4 = \begin{bmatrix} \pm \\ \pm \\ + \\ \pm \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{14}^{1} = \{p_1, p_3, p_2, p_4\}$$

- X_{33}^{1}: o valor + presente na terceira linha ($l = 3$) é neutralizado através da adição da coluna 4 (p_4) de D. Esta operação resulta em:

$$X_{34}^{1} = X_{33}^{1} \oplus D_4 = \begin{bmatrix} \pm \\ \pm \\ + \\ \pm \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{34}^{1} = \{p_3, p_2, p_1, p_4\}$$

- X_{43}^{2}: o valor + presente na primeira linha ($l = 1$) é neutralizado através da adição da coluna 1 (p_1) de D. Esta operação resulta em:

$$X_{44}^{1} = X_{43}^{2} \oplus D_1 = \begin{bmatrix} + \\ \pm \\ \pm \\ \pm \end{bmatrix} \oplus \begin{bmatrix} - \\ + \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{44}^{1} = \{p_4, p_5, p_2, p_1\}$$

- X_{53}^{1}: o valor + presente na terceira linha ($l = 3$) é neutralizado através da adição da coluna 4 (p_4) de D. Esta operação resulta em:

$$X_{54}^{1} = X_{53}^{1} \oplus D_4 = \begin{bmatrix} \pm \\ \pm \\ + \\ - \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ - \\ + \end{bmatrix} = \begin{bmatrix} \pm \\ \pm \\ \pm \\ \pm \end{bmatrix} \quad P_{54}^{1} = \{p_5, p_2, p_1, p_4\}$$

Passo 3: Como nenhum vetor X_{j4} produzido no passo anterior contém um valor + em uma de suas linhas, a 1ª ETAPA do Algoritmo 2 é finalizada com a identificação dos seguintes possíveis sifões: P_{32}^{2}, P_{52}^{2}, P_{13}^{1}, P_{13}^{3}, P_{23}^{1}, P_{43}^{1}, P_{43}^{3}, P_{14}^{1}, P_{34}^{1}, P_{44}^{1}, P_{54}^{1}.

Após a execução da 1ª ETAPA do Algoritmo 2 faz-se necessário verificar se os sifões encontrados são mínimos. Para isto a **2ª ETAPA** é executada e seus passos são descritos abaixo:

SEED$_1$: com $PLANT_1^0$ inicialmente vazio, adicionar a ele todos os sifões mínimos resultantes da 1ª ETAPA obtidos a partir de p_1:

- $P_{13}^{1} = \{p_1, p_2, p_4\}$:
  dado que $\forall P' \in PLANT_1^0 \, (P_{13}^{1} \not\subset P' \, e \, P' \not\subset P_{13}^{1})$ então $PLANT_1^1 = \{P_{13}^{1}\}$;

- $P_{13}^3 = \{p_1, p_3, p_5\}$:
  dado que $\forall P' \in PLANT_1^1 \, (P_{13}^3 \not\subset P' \, e \, P' \not\subset P_{13}^3)$ então $PLANT_1^2 = PLANT_1^1 \cup \{P_{13}^3\}$;

- $P_{14}^1 = \{p_1, p_2, p_3, p_4\}$:
  dado que $\forall P' \in PLANT_1^2 \, (P_{14}^1 \not\subset P')$ e $\exists P' \in PLANT_1^2 | (P' \subset P_{14}^1)$ então $PLANT_1^3 = PLANT_1^2$.

SEED$_2$: com $PLANT_2^0$ inicialmente vazio, adicionar a ele todos os sifões mínimos resultantes da 1ª ETAPA obtidos a partir de p_2:

- $P_{23}^1 = \{p_1, p_2, p_4\}$:
  dado que $\forall P' \in PLANT_2^0 \, (P_{23}^1 \not\subset P' \, e \, P' \not\subset P_{23}^1)$ então $PLANT_2^1 = \{P_{23}^1\}$.

SEED$_3$: com $PLANT_3^0$ inicialmente vazio, adicionar a ele todos os sifões mínimos resultantes da 1ª ETAPA obtidos a partir de p_3:

- $P_{32}^2 = \{p_3, p_5\}$:
  dado que $\forall P' \in PLANT_3^0 \, (P_{32}^2 \not\subset P' \, e \, P' \not\subset P_{32}^2)$ então $PLANT_3^1 = \{P_{32}^2\}$;

- $P_{34}^1 = \{p_1, p_2, p_3, p_4\}$:
  dado que $\forall P' \in PLANT_3^1 \, (P_{34}^1 \not\subset P' \, e \, P' \not\subset P_{34}^1)$ então $PLANT_3^2 = PLANT_3^1 \cup \{P_{34}^1\}$.

SEED$_4$: com $PLANT_4^0$ inicialmente vazio, adicionar a ele todos os sifões mínimos resultantes da 1ª ETAPA obtidos a partir de p_4:

- $P_{43}^1 = \{p_1, p_2, p_4\}$:
  dado que $\forall P' \in PLANT_4^0 \, (P_{43}^1 \not\subset P' \, e \, P' \not\subset P_{43}^1)$ então $PLANT_4^1 = \{P_{43}^1\}$;

- $P_{43}^3 = \{p_3, p_4, p_5\}$:
  dado que $\forall P' \in PLANT_4^1 \, (P_{43}^3 \not\subset P' \, e \, P' \not\subset P_{43}^3)$ então $PLANT_4^2 = PLANT_4^1 \cup \{P_{43}^3\}$;

- $P_{44}^1 = \{p_1, p_2, p_4, p_5\}$:
  dado que $\forall P' \in PLANT_4^2 \, (P_{44}^1 \not\subset P')$ e $\exists P' \in PLANT_4^2 | (P' \subset P_{44}^1)$ então $PLANT_4^3 = PLANT_4^2$.

SEED$_5$: com $PLANT_5^0$ inicialmente vazio, adicionar a ele todos os sifões mínimos resultantes da 1ª ETAPA obtidos a partir de p_5:

- $P_{52}^2 = \{p_3, p_5\}$:
  dado que $\forall P' \in PLANT_4^0 \, (P_{52}^2 \not\subset P' \, e \, P' \not\subset P_{52}^2)$ então $PLANT_5^1 = \{P_{52}^2\}$;

- $P_{54}^1 = \{p_1, p_2, p_4, p_5\}$:
  dado que $\forall P' \in PLANT_5^1 \, (P_{54}^1 \not\subset P' \, e \, P' \not\subset P_{54}^1)$ então $PLANT_5^2 = PLANT_5^1 \cup \{P_{54}^1\}$.

Após a execução da 2ª ETAPA do Algoritmo 2, faz-se necessário verificar se $LEAFs$ idênticos foram gerados a partir de diferentes $SEEDs$. Para isto a **3ª ETAPA** é executada e seus passos são descritos abaixo:

$PLANT_2^1 = \{P_{23}^1\}$: com $SB^0 = PLANT_1 = \{P_{13}^1, P_{13}^3\}$

- P_{23}^1: dado que $\exists P' \in SB^0 \,|\, (P_{23}^1 = P')$ então $SB^1 = SB^0$.

$PLANT_3^1 = \{P_{32}^2, P_{34}^1\}$: com $SB^1 = \{P_{13}^1, P_{13}^3\}$

- P_{32}^2: dado que $\forall P' \in SB^1 (P_{32}^2 \neq P')$ então $SB^2 = SB^1 \cup \{P_{32}^2\}$;
- P_{34}^1: dado que $\forall P' \in SB^2 (P_{34}^1 \neq P')$ então $SB^3 = SB^2 \cup \{P_{34}^1\}$.

$PLANT_4^1 = \{P_{43}^1, P_{43}^3\}$: com $SB^3 = \{P_{13}^1, P_{13}^3, P_{32}^2, P_{34}^1\}$

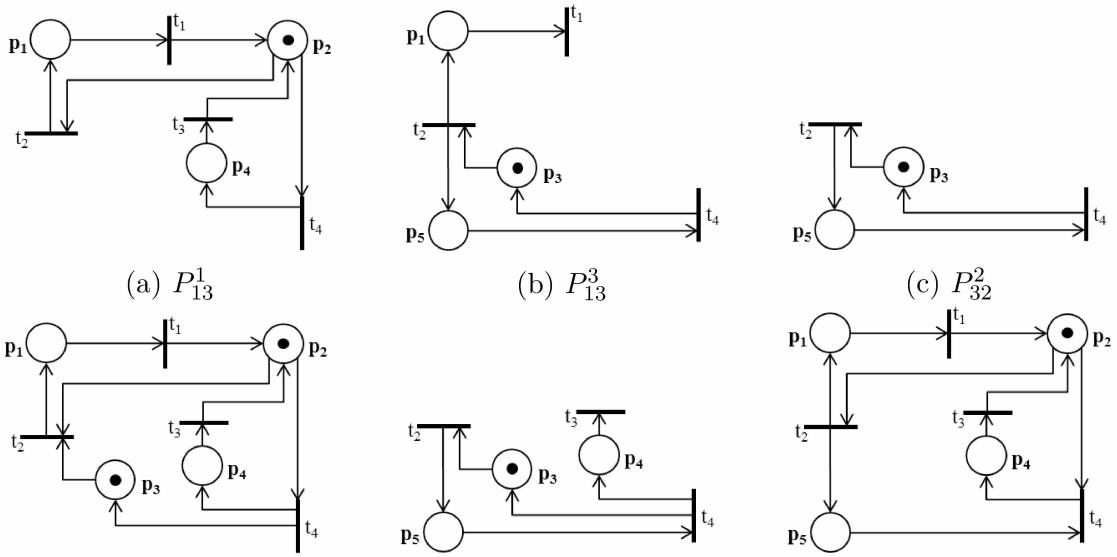
- P_{43}^1: dado que $\exists P' \in SB^3 \,|\, (P_{43}^1 = P')$ então $SB^4 = SB^3$;
- P_{43}^3: dado que $\forall P' \in SB^4 (P_{43}^3 \neq P')$ então $SB^5 = SB^4 \cup \{P_{43}^3\}$.

$PLANT_5^1 = \{P_{52}^2, P_{54}^1\}$: com $SB^5 = \{P_{13}^1, P_{13}^3, P_{32}^2, P_{34}^1, P_{43}^3\}$

- P_{52}^2: dado que $\exists P' \in SB^5 \,|\, (P_{52}^2 = P')$ então $SB^6 = SB^5$;
- P_{54}^1: dado que $\forall P' \in SB^6 (P_{54}^1 \neq P')$ então $SB^7 = SB^6 \cup \{P_{54}^1\}$.

Ao fim da execução da 3ª ETAPA do Algoritmo 2, o conjunto final SB de sifões básicos da rede de Petri mostrada na Figura 117 é obtido. Tal conjunto consiste de seis sifões básicos: $P_{13}^1 = \{p_1, p_2, p_4\}$, $P_{13}^3 = \{p_1, p_3, p_5\}$, $P_{32}^2 = \{p_3, p_5\}$, $P_{34}^1 = \{p_1, p_2, p_3, p_4\}$, $P_{43}^3 = \{p_3, p_4, p_5\}$ e $p_{54}^1 = \{p_1, p_2, p_4, p_5\}$. A representação gráfica deles é mostrada na Figura 118.

Dos sifões básicos obtidos pelo algoritmo, apenas os sifões P_{13}^1 e P_{32}^2 são classificados como sifões mínimos dado que eles são subconjunto próprio dos outros restantes. Levando em consideração estes dois sifões mínimos e que ambos possuem uma *trap*, pode-se concluir que a rede de Petri mostrada na Figura 117 é viva e, consequentemente, nenhuma situação de *deadlock* se fará presente.



(a) P^1_{13} (b) P^3_{13} (c) P^2_{32}

(d) P^1_{34} (e) P^3_{43} (f) P^1_{54}

Figura 118 – Modelos de sifões básicos representados graficamente.

APÊNDICE **B**

Funções do Jogador de WF-net possibilística

Quando se considera a ferramenta computacional CPN Tools, diversas funções são necessárias para remodelar o jogador de CPN característico do CPN Tools para o jogador de WF-net possibilística descrito na Figura 29. Destas funções, cinco ou seis, no caso das IOWF-net possibilística, são inseridas no modelo através do conceito de função de guarda, de expressões associadas aos arcos do modelo e de segmentos de códigos diversos. O emprego de cada uma das cinco funções é descrito a seguir:

1. a função V é associada aos arcos de saída direcionados para os lugares de entrada de cada transição através da expressão de arco permitindo a modelagem do disparo incerto. Desta forma, é possível "permanecer" com os objetos nos lugares de entrada de uma transição quando a mesma é pseudo-disparada;

2. a função I é associada aos arcos de saída direcionados para os lugares de saída de cada transição através da expressão de arco permitindo a modelagem do procedimento de defuzzificação;

3. a função E é associada aos arcos de saída direcionados para os lugares de saída de cada transição através da expressão de arco de forma a permitir a associação dos tempos de chegadas dos eventos relacionados aos objetos produzidos nestes lugares de acordo com o tipo de disparo;

4. a função IC é associada aos arcos de saída direcionados para os lugares de comunicação através da expressão de arco de forma a permitir a associação dos tempos de chegadas dos eventos relacionados aos objetos produzidos nestes lugares de acordo com o tipo de disparo. Diferentemente da função E, esta função é exclusiva para os objetos que se localizam nos lugares de comunicação de uma IOWF-net possibilística;

5. a função G é associada a todas as transições do modelo por meio do conceito de função de guarda com o objetivo de gerenciar a permissão do disparo das mesmas;

6. a função C é associada a todas as transições do modelo por intermédio do segmento de código $input()output()action()$. Através desta função, o objeto localizado no lugar de controle, o qual é responsável por substituir o jogador padrão do CPN Tools pelo jogador de WF-net possibilística, é atualizado e a duração necessária para a chegada de um determinado evento é definida.

Antes de explicar cada uma das funções definidas para redefinir o jogador de CPN, as declarações utilizadas nestas funções e, consequentemente, nos modelos, são descritas a seguir. Com o intuito de facilitar o entendimento das funções definidas neste trabalho, tais declarações são agrupadas de acordo com a cor (tipo e estrutura de dado).

As declarações necessárias para definir a classe do objeto $CASO$ são descritas a seguir.

```
colset idCASO = INT;
colset chEv = INT;
colset CASO = product idCASO * chEv;
```

Note-se que estas declarações são as básicas obrigatórias para que se possa redefinir a semântica operacional do jogador inerente ao CPN Tools. Entretanto, se um modelo de processo necessite de outras declarações devido ao seu comportamento especifico, as mesmas deverão ser acrescidas à declaração da cor $CASO$. O significado das cores (*colset*) acima definidas é dado a seguir:

- idCASO: define o identificador do Caso como um inteiro;
- chEv: define o tempo de chegado de um evento;
- CASO: representa a classe do objeto C_{aso} utilizada na definição da WF-net Possibilística. Nesta classe, o conjunto de atributos é formado pelo identificador do Caso e pelo tempo de chegada de um evento associado ao Caso.

Para especificar as funções de autorização associadas às transições, uma lista de inteiros foi definida para cada Caso do tipo $CASO$. O significado desta lista muda de acordo com o processo modelado. Assim sendo, tal significado será dado no momento da definição do modelo. A declaração da lista bem como a da combinação dela com o identificador do Caso são descritas abaixo:

```
colset intLIST = list INT;
colset cteCaso = product idCASO * intLIST;
```

Levando em consideração que o jogador padrão do CPN Tools está sendo redefinido para contemplar a semântica operacional própria ao jogador de WF-net possibilística, faz-se necessário representar as listas de Pre e de $Post$ de cada lugar e de cada transição

pertencente ao modelo. Estas listas são essenciais para que se possa configurar o algoritmo de defuzzificação. Assim, as declarações necessárias para representar tais listas são dadas a seguir:

```
colset Pre = list INT;
colset Pos = list INT;
colset CNL = list INT;
colset Elem = product Pre * Pos * CNL;
colset PN = list Elem;
```

O significado de cada cor acima definida é:

- Pre: corresponde à lista dos lugares (ou das transições) de entrada de uma determinada transição (ou lugar);
- Pos: corresponde à lista dos lugares (ou das transições) de saída de uma determinada transição (ou lugar);
- CNL: corresponde à lista de transições responsáveis por cancelar o disparo incerto de uma determinada transição;
- Elem: representa os elementos de entrada e saída de uma transição (ou de um lugar) e a lista de transições responsáveis por cancelar o disparo incerto dela (se for lugar, tal lista é vazia);
- PN: representa os elementos dos lugares ou das transições pertencentes ao modelo definido.

Por fim, o objeto localizado no lugar de controle responsável por gerenciar os dados necessários para a execução do modelo é definido pela cor *CTRL*. As declarações necessárias para defini-la são descritas abaixo:

```
colset TM = INT;
colset TyF = INT;
colset DF = product INT * idCASO;
colset TF = list INT;
colset TC = list INT;
colset tDF = product TF * TC;
colset PxT = product INT * INT;
colset LPxT = list PxT;
colset CxLPT = product idCASO * LPxT;
colset LM = list CxLPT;
colset ILPN = list cteCaso;
colset CTRL = product tm * tyF * DF * tDF * LM * ILPN;
```

O significado de cada cor que compõe a cor $CTRL$ é dado abaixo:

- TM: representa o tempo corrente como inteiro;
- TyF: representa o tipo de disparo realizado (1: disparo certo, 2: primeiro disparo incerto, 3: disparo incerto, 4: início da execução do procedimento de defuzzificação, 5: término da execução do procedimento de defuzzificação, 6: finalização de um disparo incerto, 7: cancelamento de um disparo incerto);
- DF: indica a transição responsável pelo início da execução do procedimento de defuzzificação para o Caso determinado por $idCASO$;
- TF: representa a lista de transições que terão o disparo incerto finalizado;
- TC: representa a lista de transições responsáveis por cancelar determinados disparos incertos;
- tDF: define as transições que terão o disparo incerto finalizado ou cancelado para o Caso indicado pelo identificador;
- PxT: define a relação lugar × transição;
- LPxT: define uma lista da relação lugar × transição;
- CxLPT: define, para o Caso indicado por um identificador, a marcação dos objetos no modelo do processo;
- LM: define a marcação de todos os Casos em execução no modelo através de uma lista constituída por produtos do tipo CxLPT. Nesta lista, quando um disparo é realizado, a marcação do Caso em questão é atualizada;
- ILPN: define para cada Caso em execução uma lista de constantes inteiras usadas nas funções de autorização associadas às transições;
- CTRL: define o tipo do objeto localizado no lugar de controle.

Além das declarações dos conjuntos de cores acima descritos, as funções fazem uso de três constantes. São elas:

- qtdeTr: indica a quantidade de transições pertencentes ao modelo do processo. Na descrição abaixo, esta quantidade é igual a 3:

```
val qtdeTr = 3: INT;
```

- pRP: descreve a lista de transições de entrada e de saída para cada lugar p pertencente ao modelo do processo. Como a constante pRP é do tipo PN, a lista referente às transições a cancelar é vazia, ou seja, é sempre descrita como []. Na descrição a seguir, a lista de Pre e $Post$ é descrita para uma rede de quatro lugares:

```
val pRP = [([],[1],[]),
           ([1],[2,3],[]),
           ([2],[],[]),
           ([3],[],[])]  : PN;
```

- tRP: descreve para cada transição t pertencente ao modelo do processo a lista de lugares de entrada e de saída bem como a lista de transições responsáveis por cancelar o disparo incerto de t. Na descrição abaixo, as listas de *Pre* e *Post* são especificadas para uma rede com três transições. Observe-se que as listas de transições a cancelar é diferente de vazia apenas para a primeira transição, isto é, nenhum disparo incerto pode ser realizado nas outras duas restantes:

```
val tRP = [([1],[2],[3]),
           ([2],[3],[]),
           ([2],[4],[])]  : PN;
```

Os valores associados a estas três constantes diferem de acordo com o modelo implementado. Assim sendo, estes valores serão especificados no Apêndice C para cada modelo implementado no CPN Tools apresentado no Capítulo 6,

Após descrever todas as declarações fundamentais para o entendimento das funções, as subseções B.1, B.2, B.3, B.4, B.5, B.6 e B.7 descrevem as funções necessárias para configurar o jogador de WF-net possibilística no CPN Tools.

B.1 Funções de Âmbito Geral

Nesta subseção, sete funções são descritas. A primeira é usada em diversas funções posteriormente especificadas. As duas a seguir são usadas em funções diretamente ou indiretamente relacionadas ao Jogador de WF-net possibilística. A quarta e a quinta são definidas para adicionar expressividade ao modelo durante a simulação do mesmo através da atribuição de um tempo de chegada, exponencialmente distribuído, aos eventos associados aos objetos do tipo *CASO*. As duas restantes são usadas em outras duas funções: uma que gerencia a permissão de disparo de uma transição e outra que atualiza o objeto que se localiza no lugar de controle após o disparo de uma transição. A especificação de cada uma das sete funções é dada a seguir:

- **nth**: esta função retorna o enésimo elemento da lista definida por $hd :: TL$ (hd é o primeiro elemento da lista e TL são todos os elementos restantes). Observe-se que o índice se inicia no número 1, ou seja, o primeiro elemento da lista é identificado pelo índice 1 e o último pelo índice n;

```
fun nth(1, hd::TL) = hd |
    nth(n, hd::TL) = nth(n-1, TL);
```

- **LIST**: esta função retorna uma lista vazia ou a correspondente ao identificador dado por *id*, isto é, se existir algum produto na lista $(id1, z) :: TL$ com o conteúdo do primeiro elemento idêntico ao conteúdo de *id* ($id1 = id$), então a lista retornada é especificada pelo segundo elemento deste produto, ou seja, z. Em contrapartida, a lista retornada é vazia;

```
fun LIST(id, []) = [] |
    LIST(id, (id1,z)::TL) = if (id1 = id)
                            then z
                            else LIST(id,TL)
```

- **posPre**: esta função é responsável por retornar uma lista composta por todas os lugares/transições de entrada (se índice 1), ou de saída (se índice 2) ou pelas transições responsáveis por cancelar o disparo incerto de uma transição (se índice 3) de cada transição/lugar pertencente à lista $hd :: TL$;

```
fun posPre(n, [], Pn:PN) = [] |
    posPre(1, hd::TL, Pn) = #1(nth(hd, Pn))^^
                             posPre(1, TL, Pn) |
    posPre(2, hd::TL, Pn) = #2(nth(hd, Pn))^^
                             posPre(2, TL, Pn) |
    posPre(3, hd::TL, Pn) = #3(nth(hd, Pn))^^
                             posPre(3, TL, Pn);
```

- **tmE**: esta função é responsável por definir a duração necessária para a chegada de um determinado evento durante a simulação do modelo. Esta duração é exponencialmente distribuída de acordo com uma duração média de chegada do evento informada pela variável *mean*;

```
fun tmE(mean:INT) =
  let val realMean = Real.fromInt mean
      val rv = exponential((1.0/realMean))
  in floor (rv + 0.5)
  end;
```

- **E**: esta função é responsável por definir o tempo de chegada do evento associado ao Caso identificado por *idC*. A fim de simular a probabilidade de um determinado evento não chegar, a função *bernoulli* foi usada. Entretanto, tal função só é utilizada quando o valor da variável n é negativo. Esta variável, quando negativa, indica que a chegada do evento em questão pode ser perdida. Assim sendo, a função *bernoulli*(1.0/70.0) calcula o sucesso e o fracasso desta probabilidade, ou seja, a

probabilidade do evento não chegar é 1 a cada 70 ocorrências[1]. Se a probabilidade do evento não chegar é verdadeira, ou seja, $bernoulli(1.0/70.0) = 1$, o valor negativo -1 é atribuído à variável que representa o seu tempo de chegada. Caso contrário, este tempo é igual ao tempo corrente acrescido da duração necessária, determinada pela variável d, para a chegada do evento em questão;

```
fun E(n, d, (idC,aEv), ctrl:CTRL) =
  if (n < 0) andalso (bernoulli(1.0/70.0) = 1)
  then (idC, ~1)
  else (idC, #1(ctrl)+d)
```

- **IC**: esta função é responsável por definir o tempo de chegada do evento associado ao objeto do tipo *CASO* identificado pelo identificador *idC*. Diferentemente da função E especificada anteriormente, esta função é exclusiva para os objetos que se localizam nos lugares de comunicação de uma IOWF-net possibilística. Desta mameira, a variável aEv, que indica o momento que o evento chega, é atualizada de forma diferente dos outros lugares quando um disparo é finalizado ou cancelado, ou o procedimento de defuzzificação é iniciado ou concluído. Ou seja:

  - se o disparo é finalizado ($tyF = 6$) ou o procedimento de defuzzificação é iniciado ($tyF = 4$) ou concluído ($tyF = 5$), o tempo de chegada do evento associado ao objeto se mantém inalterado;
  - se o disparo é cancelado ($tyF = 7$), o tempo de chegada do evento associado ao objeto é alterado para um valor negativo, indicando que tal evento ainda não chegou;
  - por fim, se o disparo é certo ($tyF = 1$) ou incerto ($tyF = 2$ ou $tyF = 3$), o tempo de chegada do evento associado ao objeto é definido pela função E especificada anteriormente.

```
fun IC(n, d, (idC,aEv), ctrl:CTRL) =
  let val tyF = #2(ctrl)
  in if (tyF = 4) orelse (tyF = 5) orelse (tyF = 6)
     then (idC,aEv)
     else if (tyF = 7)
          then (idC, ~1)
          else E(n,d,(idC,aEv),ctrl)
  end
```

[1] valor suficiente para considerar 1 evento perdido a cada 10 Casos, dado que, em média, cada Caso recebe 7 eventos referentes à conclusão das atividades durante sua execução.

- **MinMax**: esta função é responsável por determinar o tempo de chegada mínimo e máximo dos eventos pertencentes à lista de objetos do tipo $CASO$;

```
fun MinMax((min, max), []) = (min, max) |
    MinMax((min, max), (id, tm)::TL) =
      if (tm < min)
      then MinMax((tm, max), TL)
      else if(tm > max)
           then MinMax((min, tm), TL)
           else MinMax((min, max), TL)
```

- **allID**: esta função é responsável por verificar se o identificador de todos os objetos pertencentes à lista $(hd : CASO) :: TL$ é idêntico ao identificador definido por idC;

```
fun allID(idC,[]) = true |
    allID(idC,(hd:CASO)::TL) =
      let val idHD = #1(hd)
      in if (idC = idHD)
         then allID(idC, TL)
         else false
      end
```

B.2 Funções de Remoção em Lista

Nesta subseção, quatro funções são descritas. Todas tem como característica em comum a remoção de, pelo menos, um elemento em uma lista. A especificação de cada uma das três funções é dada a seguir:

- **rElem**: esta função é responsável por remover, se existir, a primeira ocorrência do elemento indicado pela variável z na lista;

```
fun rElem(z, []) = [] |
    rElem(z, hd::TL) = if (z = hd)
                       then TL
                       else hd::rElem(z,TL);
```

- **cls1**: esta função é responsável por remover, se existir, a primeira ocorrência do produto $(z1, z2)$ na lista quando o valor do primeiro elemento deste produto é idêntico ao valor da variável z;

```
fun cls1(z, []) = [] |
    cls1(z, (z1,z2)::TL) = if (z = z1)
```

```
                                then TL
                                else (z1,z2)::cls1(z,TL);
```

- **cls2**: esta função é responsável por remover, se existir, todas as ocorrências do produto $(z1, z2)$ na lista quando o valor do segundo elemento deste produto é idêntico ao valor da variável z;

```
fun cls2(z, [])      = []   |
    cls2(z, (z1,z2)::TL) = if (z = z2)
                           then cls2(z,TL)
                           else (z1,z2)::cls2(z,TL);
```

- **cls**: esta função é responsável por remover da lista LPT a primeira ocorrência (se índice 1) ou todas ocorrências (se índice 2) de cada elemento pertencente à lista $hd :: TL$;

```
fun cls(n,    [],    LPT) = LPT  |
    cls(1, hd::TL, LPT) = cls(1, TL, cls1(hd, LPT))  |
    cls(2, hd::TL, LPT) = cls(2, TL, cls2(hd, LPT))  ;
```

B.3 Funções de Verificação Relacionado à Marcação da Rede

Nesta subseção, com a necessidade de verificar se um determinado lugar está marcado de forma precisa ou não, ou que uma transição foi pseudo-disparada ou está habilitada por uma marcação imprecisa, cinco funções são definidas. Observe-se que todas estas funções fazem uso de uma lista do tipo $LP \times T$. Esta lista, como explicado anteriormente, caracteriza a marcação de um Caso em execução no modelo. Nela, quando o produto (p, t) tem o segundo elemento igual a 0, o lugar indicado por p está marcado de forma precisa. Caso contrário, quando o valor do segundo elemento é diferente de 0, ou seja, igual ao valor do identificador da transição pseudo-disparada, o lugar indicado por p está marcado de forma imprecisa. A descrição de cada função é dada a seguir:

- **pM**: esta função verifica, independentemente se é de forma precisa ou imprecisa, se o lugar p está marcado;

```
fun pM(p, [])      = false  |
    pM(p, (p1,t1)::TL) = if (p = p1)
                         then true
                         else pM(p,TL)
```

- **pUm**: esta função verifica se o lugar p está marcado de forma imprecisa. Note-se que, para que o lugar p esteja marcado de forma imprecisa, o valor de $t1$ deverá ser maior que 0;

```
fun pUm(p, [])  = false |
    pUm(p, (p1,t1)::TL) = if (p = p1)
                          then if (t1 > 0)
                               then true
                               else false
                          else pUm(p,TL)
```

- **tM**: esta função verifica se a transição t foi pseudo-disparada. Para que isto seja verificado, deve existir pelo menos um produto $(p1, t1)$ que contenha o valor de $t1$ igual ao valor de t;

```
fun tM(t, [])  = false |
    tM(t, (p1,t1)::TL) = if (t = t1)
                         then true
                         else tM(t,TL)
```

- **eLPT**: esta função é responsável por verificar se existe pelo menos um lugar p marcado independentemente do tipo de marcação (se índice 1), ou marcado de forma imprecisa (se índice 2) ou se existe pelo menos uma transição t pseudo-disparada (se o índice 3);

```
fun eLPT(n,  [],   LPT) = false |
    eLPT(1, p::TL, LPT) = if pM(p, LPT)
                          then true
                          else eLPT(1, TL, LPT)|
    eLPT(2, p::TL, LPT) = if pUm(p, LPT)
                          then true
                          else eLPT(2, TL, LPT)|
    eLPT(3, t::TL, LPT) = if tM(t, LPT)
                          then true
                          else eLPT(3, TL, LPT)
```

- **uM**: esta função é responsável por verificar se a transição t está habilitada por uma marcação imprecisa. Para isto é verificado, inicialmente, se pelo menos um dos lugares de entrada da transição t está marcado de forma imprecisa. Caso isto se confirme, o retorno é verdadeiro. Caso contrário, será verificado se a transição t foi pseudo-disparada. Se confirmado, o retorno é verdadeiro. Se não, falso;

```
fun uM(t, LPT) =
        let val pPre = #1(nth(t, tRP))
        in if eLPT(2, pPre, LPT)
           then true
           else eLPT(3, posPre(2,pPre,pRP), LPT)
        end
```

B.4 Funções de Manipulação da Lista de Marcação

Como explicado anteriormente, uma lista do tipo LM é constituída por produtos compostos por duas variáveis. A primeira variável deste produto representa o identificador de um Caso e, a segunda, uma outra lista do tipo $LP \times T$, a qual representa a relação lugar $\times$ transição ($P \times T$). Desta forma, uma lista do tipo LM representa a atual marcação de todos os Casos em execução no modelo. Nesta subseção, três funções que a manipulam são descritas a seguir:

- **pLM**: esta função retorna a marcação da rede em três partes. A parte central corresponde ao produto $(idC1, LPT1)$, o qual se refere à atual marcação do Caso determinado pelo identificador dado pela variável idC. A primeira e a terceira parte retornada, respectivamente hd e $TL1$, correspondem à marcação dos outros Casos em execução no modelo;

```
fun pLM(hd, (idC,LPT), TL, []) = (hd, (idC, LPT), TL) |
    pLM(hd, (idC,LPT), TL, (idC1,LPT1)::TL1) =
        if (idC = idC1)
        then pLM(hd, (idC1,LPT1), TL1, [])
        else pLM(hd^^[(idC1, LPT1)], (idC,LPT), [], TL1)
```

- **mLt**: esta função retorna uma lista composta por uma lista de produtos do tipo (p, t). Quando um disparo é realizado, o identificador de cada lugar de saída p da transição disparada t é combinado com o identificador de t, formando assim a lista de produtos a ser retornada;

```
fun mLt( [], t) = [] |
    mLt(p::TL, t) = (p,t)::mLt(TL,t);
```

- **aLM**: esta função é responsável por atualizar a lista de marcação após a realização de um disparo (mais especificamente, a lista do tipo $LP \times T$ do Caso correspondente ao identificador dado pela variável idC). Para isto, a lista do tipo LM é informada em três partes, ou seja, hd, (idC, LPT) e TL. Para atualizar a lista LPT, as listas dos lugares de entrada e de saída, respectivamente $pPre$ e $pPos$, da transição

disparada são inicialmente obtidas. Após isto, a lista LPT é atualizada de acordo com o tipo de disparo, ou seja:

- se o disparo foi certo, isto é, a variável $tyF = 1$, a lista referente aos lugares de saída definida através da função $mLt(pPos, 0)$ é concatenada à lista obtida após a remoção de todos os lugares de entrada de t da lista LPT através da função $cls(1, pPre, LPT)$. Observe-se que nesta situação, o identificador referente à transição disparada, informado na função mLt, é caracterizado pelo inteiro zero. Isso indica que a marcação é precisa;
- se o disparo foi incerto, isto é, a variável $tyF = 2$ ou $tyF = 3$, a lista referente aos lugares de saída definida através da função $mLt(pPos, t)$ é concatenada à lista LPT. Observe-se que nenhuma alteração foi realizada na lista LPT, ou seja, nenhum lugar de entrada de t foi removido dela. Além disto, o identificador referente à transição pseudo-disparada é informado na função mLt, caracterizando assim uma marcação é imprecisa;
- se o disparo incerto é finalizado, isto é, a variável $tyF = 6$, a lista referente aos lugares de saída definida através da função $mLt(pPos, 0)$ é concatenada à lista obtida após a remoção de todos os lugares de entrada e de saída de t da lista LPT através da função $cls(1, pPre\hat{}\hat{}pPos, LPT)$. Observe-se que os lugares de saída são removidos e inseridos novamente devido ao valor do identificador da transição. Desta forma, a marcação imprecisa indicada pela presença do identificador da transição associado aos lugares de saída de t é alterada para uma precisa após a troca do identificador por zero;
- por fim, se o disparo incerto é cancelado, isto é, a variável $tyF = 7$, os lugares de entrada de t são removidos de LPT através da função $cls(1, pPre, LPT)$. Note que o óbvio seria remover de LPT os lugares de saída, mas, a fim de simplificar o modelo, quando o disparo incerto de uma transição é cancelado, os objetos são removidos dos lugares de saída desta transição pseudo-disparada através do disparo de uma das transições de saída destes lugares.

```
fun aLM(tyF, t, hd, (idC, LPT), TL) =
  let val t1 = if (t < 0) then (qtdeTr-t) else t
      val pPre= #1(nth(t1, tRP))
      val pPos = #2(nth(t1, tRP))
  in if (tyF = 1)
     then [(idC, mLt(pPos,0)^^cls(1, pPre, LPT))]^^hd^^TL
     else if (tyF = 2 orelse tyF = 3)
          then [(idC, mLt(pPos,t)^^LPT)]^^hd^^TL
          else if (tyF = 6)
               then [(idC, mLt(pPos,0)^^
```

```
                                    cls(1, pPre^^pPos, LPT))]
                              ^^hd^^TL
                        else [(idC, cls(1, pPre, LPT))]^^hd^^TL
        end
```

Observe que o identificador da transição na função aLM pode ser um valor negativo. Isto ocorre quando um lugar de saída da transição pseudo-disparada não contém uma transição de saída que possa ser disparada e, consequentemente, retirar o objeto dele. Para solucionar isto, uma transição específica para cancelar o disparo incerto da transição é então definida e o identificador da mesma é negativo de forma a distingui-la das outras transições do modelo.

B.5 Funções responsáveis pelo algoritmo de Defuzzificação

Esta subseção define três funções responsáveis por definir, baseado no algoritmo de defuzzificação descrito no Algoritmo 1, a lista de transições que terão o disparo incerto finalizado e a lista de transições encarregadas de cancelar um disparo incerto.

- **LTPF**: esta função retorna uma lista composta pelas transições pertencentes à lista $t :: TL$ que foram pseudo-disparadas;

```
fun LTPF( [], LPT) = [] |
    LTPF(t::TL, LPT) = if tM(t,LPT)
                       then t::LTPF(TL, LPT)
                       else LTPF(TL, LPT);
```

- **aLPF**: esta função retorna um produto composto por duas listas: uma composta pelas transições que terão o disparo incerto finalizado e outra pelas transições responsáveis por cancelar um disparo incerto. Para isto, é necessário obter: as listas das transições de entrada e de saída do lugar p ($tPre$ e $tPos$ respectivamente); as duas listas compostas pelas transições pertencentes à $tPre$ e $tPos$ que foram pseudo-disparadas ($tPrePF$ e $tPosPF$ respectivamente); e as duas listas, $pPre$ e $pPos$, compostas, respectivamente, pelos lugares de entrada e de saída das transições pertencentes à lista $tPrePF$ e $tPosPF$. Após isto, esta função é novamente chamada com os argumentos atualizados enquanto existir possíveis lugares marcados de forma imprecisa. Tais argumentos são atualizados da seguinte forma:

  - as listas $pPre$ e $pPos$ são concatenadas aos elementos restantes da lista de lugares (TL) para verificar se outras transições, anteriores ou posteriores a p, foram pseudo-disparadas;

- remover da lista LPT os produtos que contenham como segundo elemento uma das transições pertencentes às listas $tPrePF$ e $tPosPF$;
- concatenar a lista $tPrePF$ à lista de transições a terem o disparo incerto finalizado;
- concatenar a lista de transições obtidas a partir da função $posPre(3, tPosPF, tRP)$ à lista das transições responsáveis por cancelar um disparo incerto.

```
fun aLPF( [], LPT, (Tf,Tc)) = (Tf,Tc) |
    aLPF(p::TL, LPT, (Tf,Tc)) =
       let val tPre = #1(nth(p, pRP));
           val tPrePF = LTPF(tPre, LPT);

           val tPos = #2(nth(p, pRP));
           val tPosPF = LTPF(tPos, LPT);

           val pPre = posPre(1, tPrePF, tRP);
           val pPos = posPre(2, tPosPF, tRP);
       in aLPF(TL^^pPre^^pPos,
               cls(2,tPrePF^^tPosPF, LPT),
               (tPrePF^^Tf,posPre(3, tPosPF, tRP)^^Tc))
       end
```

❑ **cLPF**: esta função inicializa os argumentos necessários para executar a função $aLPF$ possibilitando, assim, a definição das listas de transições usadas durante a execução do procedimento de defuzzificação. A função $cLPF$ é necessária devido à possibilidade da transição t, responsável por iniciar o procedimento de defuzzificação, ter sido pseudo-disparada anteriormente. Caso isto tenha acontecido, a lista de lugares deverá ser composta tanto pelos lugares de entrada quanto pelos lugares de saída. Caso contrário, a lista de lugares deverá ser composta apenas pelos lugares de entrada. Além disto, o disparo incerto tem de ser considerado na lista de transições responsáveis por cancelar um disparo incerto e, todos os produtos pertencentes à lista LPT que contenham tal transição devem ser removidos.

```
fun cLPF(t,LPT) =
  let val pPre = #1(nth(t, tRP));
  in if tM(t, LPT)
     then let val pPos = #2(nth(t, tRP));
              val tC = #3(nth(t, tRP));
          in aLPF(pPre^^pPos, cls2(t, LPT), ([],tC))
          end
```

```
        else aLPF(pPre, LPT, ([],[]))
    end
```

B.6 Funções relacionadas ao Jogador de WF-net possibilística

Nesta subseção, a fim de reconfigurar o jogador de CPN inerente do CPN Tools, são definidas duas funções responsáveis por determinar se uma determinada transição pode ser disparada ou não e mais duas responsáveis por atualizar o objeto que se localiza no lugar de controle de acordo com o tipo de disparo realizado. A partir destas quatro funções, uma função de guarda e uma outra de controle são definidas a fim de agrupar as informações obtidas a partir destas anteriores e gerenciar a execução do modelo baseado numa WF-net possibilística. Estas duas funções contêm uma pequena parte que se altera de acordo com o modelo implementado, as quais serão detalhadas de acordo com o modelo considerado. Além destas seis funções, mais duas, uma responsável por atualizar o tempo corrente no objeto que se localiza no lugar de controle, e outra por remover deste objeto os dados de um Caso quando a execução do mesmo é totalmente concluída, são especificadas a seguir.

Por fim, devido a presença dos lugares de comunicação nas IOWF-net possibilística, a função de guarda definida para as WF-net possibilística é levemente alterada. Desta forma, duas funções de guarda são definidas, uma para as WF-net possibilística e outra para as IOWF-net possibilística.

- **GFiring**: esta função é responsável por permitir ou não o disparo certo ou incerto de uma transição t. Se a função de autorização associada a t é avaliada como falsa, isto é, $av = 3$, o disparo de t é proibido. Caso contrário, isto é, quando $av = 1$ ou $av = 2$, o disparo é permitido, porém com uma ressalva quando $av = 2$. Para que um disparo incerto seja permitido é obrigatório a inexistência de objetos do tipo $CASO$ com o mesmo identificador dado por idC nos lugares de saída de t;

```
fun GFiring(t, av, idC, Lm) =
 if (av = 3)
 then false
 else if (av = 1)
      then true
      else let val LPT = LIST(idC, Lm);
                val pPos = #2(nth(t, tRP));
                val marked = eLPT(1, pPos, LPT);
           in if (av = 2) andalso not marked
```

```
            then true
            else false
    end
```

- **GDf**: esta função é responsável por permitir o disparo das transições pertencentes ao procedimento de defuzzificação quando o mesmo está em execução. Ou seja, no momento em que um Caso inicia um procedimento de defuzzificação, o disparo de uma transição é permitido apenas se for para finalizar ou cancelar um disparo incerto; nenhum outro tipo de disparo é então permitido até que o procedimento seja concluído;

```
fun GDf(t, idC, tD, idCD, Tf, Tc) =
  if (idC = idCD)
  then if (Tf = []) andalso (Tc = [])
       then if (t = tD)
            then true
            else false
       else if (Tc = [])
            then hd(Tf) = t
            else hd(Tc) = t
  else false
```

- **G**: fazendo uso das duas funções anteriormente especificadas, esta função é responsável por permitir ou não o disparo de uma transição em uma WF-net possibilística. Para isto, inicialmente é verificado, através da função $allID$, se todos os objetos que habilitam a transição t contêm o mesmo identificador. Se sim, duas possibilidades são verificadas:

  - se nenhum procedimento de defuzzificação estiver em execução (ou seja, $tD = 0$), então deve-se verificar, através da função $GFiring$, se a transição pode ou não ser disparada de forma certa ou incerta, ou se o procedimento de defuzzificação pode ser iniciado. Para isto, inicialmente, tem de se executar a função de autorização através da função $funAut$ para, posteriormente, executar a função $GFiring$. Observe-se que na função $funAut$ a lista de parâmetros é indicada com três pontos. Isto ocorre devido à necessidade de inserir ou remover um determinado parâmetro de acordo com o modelo implementado. Assim sendo, a lista de parâmetros a ser usada juntamente com outras possíveis funções, necessárias para se obter determinados parâmetros, é especificada separadamente, no Apêndice C, para cada modelo implementado no CPN Tools apresentado no Capítulo 6;

– se um procedimento de defuzzificação está em curso, a função GDf é chamada para verificar se a transição t pertence ao conjunto de transições a terem o disparo incerto finalizado ($t \in Tf$) ou cancelado ($t \in Tc$), ou representa a transição responsável por finalizar o procedimento de defuzzificação ($t = tD$). Se t não pertence a tal conjunto ou o identificador do Caso a ser analisado não corresponde ao do Caso em defuzzificação ($idC \neq idCD$), a transição t não pode ser disparada.

```
fun G(t, (idC, aEv)::TL,
      (tm,tyF,(tD,idCD),(Tf,Tc),Lm,ILPn):CTRL)=
  if allID(idC, TL)
  then if (tD = 0)
       then let val av = funAut( ... );
            in GFiring(t, av, idC, Lm)
            end
       else GDf(t, idC, tD, idCD, Tf, Tc)
  else false
```

❑ **G**: devido à característica particular dos lugares de comunicação, esta função G deve diferenciar os lugares de comunicação dos outros lugares de saída de uma determinada transição t em uma IOWF-net possibilística. Assim sendo, duas listas de objetos são especificadas na lista de parâmetros desta função. Uma para representar os lugares de comunicação que caracterizam o envio de um evento (CP) e outra para representar todos os outros lugares de entrada de t, inclusive os lugares de comunicação que caracterizam o recebimento de um evento ($(idC, aEv) :: TL$). A lista CP, através da execução da função $allID$, garante que todos os objetos que habilitam a transição t contêm o mesmo identificador. Desta forma, é possível garantir que qualquer atualização de valor nas variáveis pertencentes a um destes objetos será realizada considerando o mesmo Caso em execução. O restante desta função é idêntica a função acima descrita. O código é descrito a seguir:

```
fun G(t, (idC,aEv)::TL, CP,
      (tm,tyF,(tD,idCD),(Tf,Tc),Lm,ILPn):CTRL) =
  if allID(idC, TL^^CP)
  then if (tD = 0)
       then let val av = interpretation(...)
            in GFiring(t, av, idC, Lm)
            end
       else GDf(t, idC, tD, idCD, Tf, Tc)
  else false
```

- **TPlayerFiring**: esta função é responsável por atualizar o objeto localizado no lugar de controle quando um disparo certo ou incerto é realizado, ou quando o procedimento de defuzzificação é iniciado. Para cada um destas três opões, tal objeto é atualizado de forma diferente, ou seja:

  - quando a transição t está habilitada por uma marcação imprecisa e a função de autorização associada a t é avaliada como verdadeira ($av = 1$), o procedimento de defuzzificação é iniciado. No momento em que isto ocorre, as seguintes modificações no objeto que se localiza no lugar de controle são realizadas: o valor 4 é atribuído à variável tyF, a qual define o tipo de disparo; o produto do tipo DF, encarregado de indicar a transição responsável pelo início da execução do procedimento de defuzzificação para o Caso, é atualizado para (t, idC); e as listas que definem as transições pseudo-disparadas a finalizar e a cancelar são definidas através da função $cLPF(t, LPT)$. As outras variáveis pertencentes a tal objeto permanecem inalteradas;
  - quando a função de autorização associada à transição t é avaliada como incerta ($av = 2$), um disparo incerto é realizado independentemente do tipo de marcação. Sempre que isto ocorre, duas modificações no objeto que se localiza no lugar de controle são realizadas: o valor 2 (caso seja o primeiro disparo incerto) ou valor 3 (caso não seja o primeiro disparo incerto) é atribuído à variável TyF e a lista de marcações é atualizada através da função $aLM(2, t, hd, (idC, LPT), TL)$ ou $aLM(3, t, hd, (idC, LPT), TL)$. As outras variáveis pertencentes a tal objeto permanecem inalteradas;
  - por fim, quando a transição t está habilitada por uma marcação precisa e a função de autorização associada a t é avaliada como verdadeira ($av = 1$), um disparo certo é realizado. Quando isto ocorre, duas modificações no objeto que se localiza no lugar de controle são realizadas: o valor 1 é atribuído à variável tyF e a lista de marcações é atualizada através da função $aLM(1, t, hd, (idC, LPT), TL)$. As outras variáveis pertencentes a tal objeto permanecem inalteradas.

```
fun TPlayerFiring(t,av,idC,hd,LPT,TL,Lm,tm,ILPn) =
  if uM(t,LPT)
  then if (av = 1)
       then (tm, 4, (t,idC), cLPF(t,LPT), Lm,ILPn)
       else (tm, 3, (0,0), ([],[]),
             aLM(3,t,hd,(idC, LPT), TL),ILPn)

  else if (av = 1)
       then (tm, 1, (0,0), ([],[]),
```

```
                aLM(1,t,hd,(idC, LPT), TL),ILPn)
        else (tm, 2, (0,0), ([],[]),
                aLM(2,t,hd,(idC, LPT), TL),ILPn)
```

- **TPlayerDf**: esta função é responsável por atualizar o objeto que se localiza no lugar de controle quando um disparo incerto é finalizado ou cancelado, ou quando o procedimento de defuzzificação é concluído. Para cada uma destas três opões, tal objeto é atualizado de forma diferente, ou seja:
  - se o procedimento de defuzzificação é concluído, ou seja, todos os disparos incertos foram finalizados ou cancelados, então as seguintes modificações no objeto que se localiza no lugar de controle são realizadas: o valor 5 é atribuído à variável tyF; as variáveis do produto (t, idC) do tipo DF são zeradas; e as listas que definem as transições pseudo-disparadas a finalizar e a cancelar são definidas como vazias. As outras variáveis pertencentes a tal objeto permanecem inalteradas;
  - se todos os disparos incertos a serem cancelados já tenham sido efetivados, os disparos incertos a serem finalizados serão então efetuados. Ao finalizar o disparo incerto de uma transição t, as seguintes modificações no objeto que se localiza no lugar de controle são realizadas: o valor 6 é atribuído à variável tyF; a transição t é removida da lista das transições a terem o seu disparo incerto finalizado através da função $rElem(t, Tf)$; e a lista de marcações é atualizada através da função $aLM(6, t, hd, (idC, LPT), TL)$. As outras variáveis pertencentes a tal objeto permanecem inalteradas;
  - por fim, quando uma transição t é a responsável pelo cancelamento de um disparo incerto, as seguintes modificações no objeto que se localiza no lugar de controle são realizadas: o valor 7 é atribuído à variável tyF; a transição t é removida da lista das transições responsáveis por cancelar um disparo incerto através da função $rElem(t, Tc)$; e a lista de marcações é atualizada através da função $aLM(7, t, hd, (idC, LPT), TL)$. As outras variáveis pertencentes a tal objeto permanecem inalteradas.

```
fun TPlayerDf(t,idC,hd,LPT,TL,Tf,Tc,tD,tm,ILPn) =
  if (Tf = []) andalso (Tc = [])
  then (tm,5,(0,0),([],[]),[(idC, LPT)]^^hd^^TL,ILPn)
  else if (Tc = [])
       then (tm, 6, (tD,idC), (rElem(t,Tf),Tc),
               aLM(6,t,hd,(idC,LPT),TL), ILPn)
       else (tm, 7, (tD,idC), (Tf,rElem(t,Tc)),
               aLM(7,t,hd,(idC, LPT),TL), ILPn)
```

- **C**: fazendo uso das duas funções anteriormente especificadas, esta função é responsável por atualizar o objeto que se localiza no lugar de controle e por definir a duração necessária para a chegada de um determinado evento. Para atualizar o objeto que se localiza no lugar de controle, o qual é necessário para adequar o jogador de CPN ao jogador de WF-net possibilística, a lista de marcação da rede é inicialmente particionada em três partes: no produto (idC, LPT), que representa a atual marcação do Caso indicado pelo identificador idC, e nas duas variáveis hd e $TL1$, que representam a marcação dos outros Casos em execução no modelo. A duração necessária para a chegada do evento em questão é obtida por meio da função tmE considerando a duração média informada pela variável d. A atualização do objeto que se localiza no lugar de controle é realizada da seguinte forma:
  - se nenhum procedimento de defuzzificação estiver em execução (ou seja, $tD = 0$), então a função de autorização é executada através da função $funAut$. Posteriormente, a função $TPlayerFiring$ é executada a fim de atualizar o objeto que se localiza no lugar de controle de acordo com o tipo do disparo. Idem à função G, os parâmetros das função $funAut$ juntamente com outras possíveis funções, necessárias para se obter determinados parâmetros, serão especificados separadamente, no Apêndice C, para cada modelo implementado no CPN Tools apresentado no Capítulo 6;
  - se um procedimento de defuzzificação está em curso, então a função $TPlayerDf$ é executada, atualizando o objeto que se localiza no lugar de controle.

```
fun C(t,(idC,aEv)::TL,
      (tm,tyF,(tD,idCD),(Tf,Tc),Lm,ILPn):CTRL,d) =
  let val (hd,(idC,lpt),TL1) = pLM([],(idC,[]),[],Lm);
      val dur = tmE(d);
  in if (tD = 0)
     then let val av = funAut( ... );
              val r = TPlayerFiring(t,av,idC,hd,lpt,TL1,
                                    lm,tm,ILPn);
          in (r, dur)
          end
     else let val r = TPlayerDf(t,idC,hd,lpt,TL1,Tf,Tc,
                                tD,tm,ILPn);
          in (r, dur)
          end
  end
```

- **upT**: esta função é responsável por incrementar o tempo corrente em uma unidade quando nenhuma transição pode ser disparada;

```
fun upT(ctrl:CTRL) = let val tm = #1(ctrl) + 1;
                     in CTRL.set_1(ctrl) tm
                     end
```

- **remC**: esta função remove do objeto que se localiza no lugar de controle, através das funções $cls1(idC, Lm)$ e $cls1(idC, ILPn)$, os dados de um Caso quando a execução do mesmo é finalizada;

```
fun remC((idC,aEv),(tm, tyF, Df, tD, Lm, ILPn):CTRL) =
  if (tyF = 1)
  then (tm, tyF, Df, tD, cls1(idC,Lm), cls1(idC,ILPn))
  else (tm, tyF, Df, tD, Lm, ILPn)
```

B.7 Funções relacionadas às condições de ida/volta de um objeto

Nesta subseção são descritas duas funções responsáveis por gerenciar a permanência ou a produção dos objetos nos lugares de entrada ou de saída quando: uma transição é disparada, seja de forma certa ou incerta, ou um disparo incerto é finalizado ou cancelado, ou o procedimento de defuzzificação está se iniciando ou se finalizando.

- **I**: esta função é responsável por permitir ou não a produção de um objeto nos lugares de saída da transição disparada. Note-se que apenas quando um disparo certo ou incerto é realizado que tal produção é permitida. Nas outras situações, isto é, quando o procedimento de defuzzificação está se iniciando ou se finalizando, ou quando um disparo incerto é finalizado ou cancelado, tal produção é proibida;

```
fun I(ctrl:CTRL) =
  let val tyF = #2(ctrl)
  in if (tyF = 1) orelse (tyF = 2) orelse (tyF = 3)
     then true
     else false
  end
```

- **V**: esta função é responsável por "permanecer" com os objetos nos lugares de entrada da transição disparada. Note-se que apenas quando um disparo incerto é realizado ou quando o procedimento de defuzzificação está se iniciando ou se finalizando que os objetos permanecem nos lugares de entrada da transição em questão. Nas outras situações, isto é, quando um disparo incerto é finalizado ou cancelado ou quando um disparo certo é realizado, tal permanência não é permitida;

```
fun V(ctrl:CTRL) =
  let val tyF = #2(ctrl)
  in if (tyF = 2) orelse (tyF = 3) orelse
        (tyF = 4) orelse (tyF = 5)
     then true
     else false
  end
```

APÊNDICE C

Funções específicas aos modelos implementados no CPN Tools

As funções de autorização referente aos cinco modelos implementados no CPN Tools e mostrados no Capítulo 6, juntamente com a descrição da parte que se modifica, devido à característica particular de cada modelo, nas funções G e C, declaradas na Seção B.6, são explicadas nas Seções C.1, C.2, C.3, C.4 e C.5. Além disto, os valores das três constantes, que as funções declaradas no Apêndice B fazem uso, também são informados de acordo com o modelo descrito.

C.1 WF-net Possibilística apresentada na Seção 6.1.1

Nesta Seção, os valores das três constantes $qtdeTr$, pRP e tRP, as funções de autorização e a parte que se modifica nas funções G e C [1], referente ao modelo implementado no CPN Tools do processo de tratamento de reclamações definido por (AALST; HEE, 2004) e apresentado na Figura 50, são dados a seguir.

Primeiramente, os valores específicos, para este modelo, das constantes $qtdeTr$, pRP e tRP, declaradas no Apêndice B, são especificados considerando os identificadores dos lugares e das transições mostrados nos nomes dos lugares e das transições do modelo apresentado na Figura 50. Tais constantes são responsáveis por definir, respectivamente, a quantidade de transições pertencentes ao modelo do processo, a lista de transições de entrada e de saída para cada lugar p pertencente ao modelo do processo, e a lista de lugares de entrada e de saída de cada transição t juntamente com a lista de transições responsáveis por cancelar o disparo incerto de t. Os valores atribuídos a elas são:

```
val qtdeTr = 16: INT;
```

[1] A função G é responsável por permitir ou não o disparo de uma transição e a função C por atualizar o objeto que se localiza no lugar de controle e definir a duração necessária para a chegada de um determinado evento. Para maiores detalhes, vide Seção B.6.

```
val pRP = [([],      [1],      []),
           ([1],     [2],      []),
           ([2],     [3],      []),
           ([2],     [5],      []),
           ([3],     [4],      []),
           ([5],     [6],      []),
           ([4],     [7],      []),
           ([6],     [7],      []),
           ([7],     [8],      []),
           ([8],     [9],      []),
           ([9],     [10],     []),
           ([10],    [11,13],  []),
           ([11],    [12],     []),
           ([13],    [14],     []),
           ([12,14], [15],     []),
           ([15],    [16],     []),
           ([16],    [],       [])] : PN;

val tRP = [([1],   [2],   [2]),
           ([2],   [3,4], [3,5]),
           ([3],   [5],   [4]),
           ([5],   [7],   [~1]),
           ([4],   [6],   [6]),
           ([6],   [8],   [~2]),
           ([7,8], [9],   [8]),
           ([9],   [10],  [9]),
           ([10],  [11],  [10]),
           ([11],  [12],  [11]),
           ([12],  [13],  [12]),
           ([13],  [15],  [15]),
           ([12],  [14],  [14]),
           ([14],  [15],  [15]),
           ([15],  [16],  [16]),
           ([16],  [17],  []),
           ([7],   [],    []),
           ([8],   [],    [])] : PN;
```

Lembre-se que, para simplificar a implementação do modelo no CPN Tools, o cancelamento do disparo incerto de uma transição t é realizado pelas transições de entrada dos lugares de saída de t. Desta forma, a terceira lista de cada produto pertencente à variá-

vel tRP representa, através dos identificadores, as transições responsáveis por cancelar o disparo incerto da transição em questão. Observe-se que, nas transições com identificadores 4 e 6, a terceira lista contém um valor negativo. Isto ocorre porque a transição de entrada dos lugares de saída, tanto da transição 4 quanto da 6, necessita tanto do lugar $W3$ quanto do lugar $W4$ marcados, o que nem sempre ocorre quando é necessário cancelar o disparo incerto realizado na transição 4 ou na 6. Assim sendo, duas transições com identificadores iguais à -1 e à -2 são adicionadas ao modelo com o único objetivo de cancelar, respectivamente, o disparo incerto da transição 4 e o da 6.

Antes de explicar a função $funAut$ e a parte que se modifica nas funções G e C devido à característica particular do modelo, o significado da variável $cteCaso$ deve ser descrito. Esta variável define uma lista com 16 constantes inteiras para cada Caso em execução. Esta lista determina os tempos máximos de espera usados nas funções de autorização associadas às transições do Caso em questão. Isto é, quando o valor de um destes inteiros é 0 (zero), a transição correspondente só poderá ser disparada de forma certa. Caso contrário, um disparo incerto poderá ser realizado quando o tempo corrente alcançar o valor definido para a transição em questão nesta lista.

A função $funAut$, descrita abaixo, é responsável por agrupar todas as funções de autorização associadas às transições. Ela faz uso da variável $ILPn$, a qual contém os tempos máximo de espera de todos os Casos em execução. Desta forma, através da função $LIST$, é possível obter a lista de tempos de espera máximo para o Caso com identificador idC e, posteriormente, através da função nth, obter o tempo máximo de espera relacionado à transição t. Assim sendo, as possíveis avaliações são:

- se todos os eventos necessários para disparar a transição t já chegaram, ou seja, $maxTC$ e $minTC$ não são negativos e o tempo corrente tm é igual ao tempo da ocorrência do último evento ($tm = maxTC$), então, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1;

- se o tempo máximo de espera é maior que zero (ou seja, é permitido um disparo incerto) e o tempo corrente alcançou este tempo máximo, então, a função de autorização associada a t é avaliada como incerta retornando o valor 2, exceto se a transição for identificada pelo identificador 7. Neste caso, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1;

- se nenhuma das duas possibilidades acima descritas é verificada, então, a função de autorização associada a t é avaliada como falsa, retornando o valor 3.

```
fun funAut(t, idC, ILPn, tm, minTC, maxTC) =
if (t > 0)
then if (tm = maxTC) andalso (minTC > (~1))
     then 1
```

```
else let val cteCASO = LIST(idC, ILPn);
         val maxT = nth(t, cteCASO);
     in if (MtT > 0) andalso (tm >= maxT)
        then if (t = 7) then 1 else 2
        else 3
     end
else 3
```

A parte que se modifica, tanto na função G quanto na função C, devido à característica particular do modelo, é dada abaixo:

```
val (minTC, maxTC) = MinMax((aEv, aEv), TL);
val av = funAut(t,idC,ILPn,tm,minTC,maxTC);
```

Note-se que foi necessário o uso de outra função para completar a lista de parâmetros usadas na função $funAut$. A função $MinMax$ é utilizada para definir o tempo de chegada do primeiro ($minTC$) e do último evento ($maxTC$) relacionados aos objetos que habilitam t.

C.2 WF-net Possibilística apresentada na Seção 6.1.2

Nesta Seção, os valores das três constantes $qtdeTr$, pRP e tRP, as funções de autorização e a parte que se modifica nas funções G e C, referente ao modelo implementado no CPN Tools do processo de encomenda definido por (WESKE, 2007) e apresentado na Figura 51, são dados a seguir. Além disto, as funções $aLPF$, $cLPF$ e $GFiring$ [2] descritas no Apêndice B são modificadas para suportar a troca do algoritmo de defuzzificação apresentado no Algoritmo 1 pelo apresentado no Algoritmo 3.

Primeiramente, os valores específicos para este modelo das constantes $qtdeTr$, pRP e tRP, declaradas no Apêndice B, são especificados abaixo considerando os identificadores dos lugares e das transições mostrados nos nomes dos lugares e das transições do modelo apresentado na Figura 50.

```
val qtdeTr = 14: INT;

val pRP = [([],  [1], []),
           ([1], [2], []),
           ([2], [3], []),
           ([3], [4], []),
           ([4], [5], []),
           ([5], [6], []),
```

[2] As funções $aLPF$ e $cLPF$ são responsáveis por definir a lista de transições a terem o disparo incerto cancelado e a função $GFiring$ por permitir ou não o disparo certo ou incerto de uma determinada transição

```
                ([6], [7], []),
                ([7], [8], []),
                ([8], [9], []),
                ([9], [10],[]),
                ([10],[11],[]),
                ([11],[12],[]),
                ([12],[13],[]),
                ([13],[14],[]),
                ([14],[], [])] : PN;

val tRP = [([1], [2], []),
           ([2], [3], []),
           ([3], [4], []),
           ([4], [5], []),
           ([5], [6], []),
           ([6], [7], [7]),
           ([7], [8], [8]),
           ([8], [9], [9]),
           ([9], [10],[10]),
           ([10],[11],[11]),
           ([11],[12],[12]),
           ([12],[13],[13]),
           ([13],[14],[]),
           ([14],[15],[])] : PN;
```

Observe na variável tRP, através da terceira lista de cada produto, que somente as transições com identificadores 6, 7, 8, 9, 10, 11 e 12 podem ser disparadas de forma incerta. Nas outras transições, esta lista se encontra vazia. Isto indica que nenhuma transição é responsável pelo cancelamento delas e, consequentemente, nenhum disparo incerto pode ser realizado nelas.

Levando-se em consideração que este modelo não faz uso do algoritmo de defuzzificação apresentado no Algoritmo 1 e sim do Algoritmo 3, algumas modificações são realizadas nas funções $aLPF$, $cLPF$ e $GFiring$ especificadas no Apêndice B para suportar tal troca. Assim sendo, estas funções são reescritas abaixo:

- **aLPF**: levando em consideração que nenhum disparo incerto pode ser finalizado, esta função retorna um produto composto por duas listas: uma vazia e outra composta pelas transições responsáveis por cancelar um disparo incerto. Para isto, é necessário obter: a lista $tPos$ das transições de saída do lugar p; a lista $tPosPF$ composta pelas transições pertencentes à $tPos$ que foram pseudo-disparadas; e a lista $pPos$ formada pelos lugares de saída das transições pertencentes à lista $tPosPF$.

Após isto, esta função é novamente chamada com os argumentos atualizados enquanto existir possíveis lugares posteriores marcados de forma imprecisa. Tais argumentos são atualizados da seguinte forma:

- a lista $pPos$ é concatenada aos elementos restantes da lista de lugares (TL) para verificar se outras transições posteriores a p foram pseudo-disparadas;
- remover da lista LPT os produtos que contenham como segundo elemento uma das transições pertencentes à lista $tPosPF$;
- concatenar a lista de transições obtidas a partir da função $posPre(3, tPosPF, tRP)$ à lista das transições responsáveis por cancelar um disparo incerto.

```
fun aLPF([], LPT, Tc) = ([],Tc) |
    aLPF(p::TL, LPT, Tc) =
        let val tPos = #2(nth(p, pRP));
            val tPosPF = LTPF(tPos,LPT);
            val pPos = posPre(2, tPosPF, tRP);
        in aLPF(tl^^pPos,
                cls(2,tPosPF, LPT),
                (posPre(3, tPosPF, tRP)^^Tc))
        end
```

- **cLPF**: esta função inicializa os argumentos necessários para executar a função $aLPF$ possibilitando, assim, a definição das listas de transições usadas durante a execução do procedimento de defuzzificação. Note-se que quando a transição t, responsável por iniciar o procedimento de defuzzificação, não foi pseudo-disparada, o retorno é um produto com duas listas vazias. Em contrapartida, se t foi pseudo-disparada, a função $aLPF$ é chamada considerando os seguintes argumentos: uma lista composta apenas pelos lugares de saída de t; a lista LPT com todos os produtos que contenham t como segundo elemento removidos; e a transição responsável por cancelar o disparo incerto de t na lista das transições responsáveis por cancelar os disparos incertos;

```
fun cLPF(t,LPT) =
  if tM(t, LPT)
  then let val pPos = #2(nth(t, tRP));
           val Tc = #3(nth(t, tRP));
       in aLPF(pPos, cls2(t, LPT), tC) end
  else ([],[])
```

- **GFiring**: esta função, diferentemente da especificada na seção B.6, permite o disparo de uma transição t com função de autorização avaliada como verdadeira

($av = 1$) somente se t foi anteriormente pseudo-disparada ($marked = true$) ou está habilitada por uma marcação precisa ($unMk = false$). O objetivo desta modificação é evitar executar um procedimento de defuzzificação com nenhum disparo incerto a ser cancelado. As condições para as outras situações, quando a função de autorização é avaliada como incerta ($av = 2$) ou como falsa ($av = 3$), se mantêm inalteradas;

```
fun GFiring(t, av, idC, Lm) =
  if (av=3)
  then false
  else let val LPT = LCaso(idC, Lm);
           val pPos = #2(nth(t, tRP));
           val marked = eLPT(1, pPos, LPT);
           val pPre::TL = #1(nth(t, tRP));
           val unMk = pUm(pPre, LPT);
       in if ((av = 1) andalso (marked orelse not unMk))
             orelse
             ((av = 2) andalso not marked)
          then true
          else false
       end
```

Antes de explicar a função $funAut$ e a parte que se modifica nas funções G e C devido à característica particular do modelo, o significado da variável $cteCaso$ deve ser descrito. Esta variável define uma lista com 21 contantes inteiras para cada Caso em execução. Esta lista determina nas primeiras 14 constantes os tempos máximos de espera usados nas funções de autorização associadas às transições do Caso em questão. Isto é, quando o valor de um destes inteiros é 0 (zero), a transição correspondente só poderá ser disparada de forma certa. Caso contrário, um disparo incerto poderá ser realizado quando o tempo corrente alcançar o valor definido para a transição em questão nesta lista. As outras 7 constantes restantes correspondem ao tempo de execução gasto em cada uma das sete atividades presentes neste modelo.

A função $funAut$, descrita abaixo, é responsável por agrupar todas as funções de autorização associadas às transições. Ela faz uso da variável $ILPn$ a qual contém todos os tempos máximo de espera dos Casos em execução além do tempo de execução de cada atividade presente no modelo. Desta forma, através da função $LIST$, é possível obter a lista de tempos para o Caso com identificador idC e, posteriormente, através da função nth, obter o tempo máximo de espera relacionado à transição t. Assim sendo, as possíveis avaliações são:

- se o evento necessário para disparar a transição t já chegou, ou seja, o tempo corrente

tm é igual ao tempo da ocorrência do evento ($tm = mTC$), então, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1;

- se o tempo máximo de espera é maior que zero (ou seja, é permitido um disparo incerto) e o tempo corrente alcançou este tempo máximo ($tm >= mtT$), então, a função de autorização associada a t é avaliada como incerta, retornando o valor 2;

- se nenhuma das duas possibilidades acima descritas é verificada, então, a função de autorização associada a t é avaliada como falsa, retornando o valor 3.

```
fun funAut(t, tm, mtC, idC, ILPn) =
  if (t > 0)
  then if (tm = mtC)
       then 1
       else let val cteCASO = LIST(idC, ILPn);
                val mtT = nth(t, cteCASO);
            in if (mtT > 0) andalso (tm >= mtT)
               then 2
               else 3
            end
  else 3
```

A lista de parâmetros da função $funAut$ presente tanto na função G quanto na função C é:

```
val av = funAut(t, tm, aEv, idC, ILPn);
```

As 7 constantes restantes na lista $ILPn$ são necessárias para definir o tempo de chegada do evento referente a conclusão das atividades $A1$, $A2$, $A3$, $A4$, $A5$, $A6$ e $A7$. Isto é necessário dado que uma atividade por ser iniciada através de um disparo incerto quando a atividade imediatamente anterior ainda não tenha sido concluída. Assim sendo, para considerar o tempo de execução gasto destas atividades, duas funções são definidas e a função C é alterada.

A definição das duas funções necessárias para armazenar este tempo gasto após a realização de um disparo incerto são dadas abaixo:

- **partList**: esta função é responsável por particionar uma lista de acordo com o índice dado por n. Ela retorna uma lista Z com todos os elementos pertencentes aos índices anteriores à n, o elemento z presente no índice n e uma lista $Z1$ com todos os elementos pertencentes aos índices posteriores à n.

```
fun partList(1, Z, z::Z1) = (Z, z, Z1) |
    partList(n, Z, z::Z1) = partList(n-1, Z^^[z], Z1);
```

- **attDR**: esta função é responsável por redefinir o tempo de chegada de um evento quando um disparo certo é realizado ou por atualizar os valores referentes ao tempo de execução das atividades na lista $ILPn$ para o Caso em questão. Para isto, o identificador usado na lista $ILPn$ para identificar a atividade é inicialmente obtido através das 3 primeiras linhas de código. Após isto, a lista pertencente ao Caso determinado por idC em $ILPn$ é obtida através da função pLM e, posteriormente, tal lista é particionada em três partes: nas listas Z e $Z1$ referentes, respectivamente, aos elementos anteriores e posteriores ao índice n e no inteiro drZ que indica o tempo de execução da atividade analisada. Por fim, a duração necessária para a chegada do evento, representada pela variável dur, ou o tempo de execução da atividade são modificados da seguinte forma:

  - se o disparo é incerto (($tyF = 2$) ou ($tyF = 3$)) e a transição em questão refere-se ao início da execução da atividade ($not\,tPar$), ou o disparo incerto realizado é cancelado ($tyF = 7$) e transição refere-se ao fim da execução da atividade ($tPar$), então a duração da atividade é atualizada subtraindo do tempo corrente tm a duração armazenada em drZ, a qual, inicialmente, tem como valor o inteiro 0;
  - se o disparo é certo ($tyF = 1$) deve-se verificar se a duração necessária para a chegada do evento dur é maior que a duração drZ armazenada. Se sim, este tempo é modificado para 0 ou, caso contrário, para a diferença entre dur e drZ.

```
fun attDR(t, idC, dur, (tm,tyF,Df,TDf,Lm,ILPn):CTRL) =
  let val tPar = (t mod 2) = 0;
      val realT = if (tPar)
                  then (Real.fromInt t)- 2.0
                  else (Real.fromInt t) - 1.0
      val n = floor(realT/2.0) + 15
      val (hd, (idC,iLPn), TL) = pLM([],(idC,[]),[],ILPn);
      val (Z, drZ, Z1) = partList(n, [], iLPn)
  in if (((tyF = 2) orelse (tyF = 3)) andalso (not tPar))
        orelse
        ((tyF = 7) andalso (tPar))
     then ((tm,tyF,Df,TDf,Lm,
            hd^^[(idC,Z^^[tm-drZ]^^Z1)]^^TL), dur)
     else if (tyF = 1) andalso ((dur - drZ) < 0)
          then ((tm,tyF,Df,TDf,Lm,ILPn), 0)
          else ((tm,tyF,Df,TDf,Lm,ILPn), dur-drZ)
  end
```

A fim de modificar a duração da chegada de um evento definida na função C através da variável dur ou atualizar o tempo de execução de uma atividade na variável $ILPn$, a chamada da função $attDR(t, idC, dur, r)$ substitui, neste modelo, a linha referente ao retorno (r, dur) da função C. Assim sendo, a duração e o objeto que se localiza no lugar de controle serão modificados através da função $attDR$.

C.3 WF-net Possibilística apresentada na Seção 6.1.3

Nesta Seção, os valores das três constantes $qtdeTr$, pRP e tRP, as funções de autorização e a parte que se modifica nas funções G e C, referente ao modelo implementado no CPN Tools da versão simplificada de um processo de aplicação de cartão de crédito definido por (WYNN et al., 2009) e apresentada na Figura 52 são dados a seguir.

Considerando os identificadores dos lugares e das transições, os valores atribuídos às constantes $qtdeTr$, pRP e tRP usadas neste modelo são:

```
val qtdeTr = 19: INT;

val pRP = [([],       [1],     []),
           ([1],      [2,11],  []),
           ([1,6],    [3],     []),
           ([4,7],    [5,6],   []),
           ([5],      [7],     []),
           ([3],      [4,8],   []),
           ([8],      [9,10],  []),
           ([9,10],   [2,11],  []),
           ([11],     [12,13],[]),
           ([13],     [14],    []),
           ([13],     [15],    []),
           ([14],     [19],    []),
           ([15],     [16,17],[]),
           ([17],     [18],    []),
           ([16,18],  [19],    []),
           ([2,12,19],[],      [])] : PN;

val tRP = [([1],    [2,3],   []),
           ([2,8],  [16],    []),
           ([3],    [6],     [8]),
           ([6],    [4],     [6]),
           ([4],    [5],     [7]),
           ([4],    [3],     [3]),
           ([5],    [4],     [6]),
```

```
([6],      [7],      [9]),
([7],      [8],      [11]),
([7],      [8],      [11]),
([2,8],    [9],      []),
([9],      [16],     []),
([9],      [10,11],  []),
([10],     [12],     []),
([11],     [13],     []),
([13],     [15],     []),
([13],     [14],     []),
([14],     [15],     []),
([12,15],  [16],     [])]   : PN;
```

Observe na variável tRP, através da terceira lista de cada produto, que somente as transições com identificadores 3, 4, 5, 6, 7, 8, 9 e 10 podem ser disparadas de forma incerta. Estas transições se referem à região de cancelamento, isto é, quando um pedido de cancelamento é solicitado, se o Caso referente a este pedido se localizar em algum destes lugares, uma sequência de disparos incertos será realizada enquanto a transição com identificador 2 não estiver habilitada. No momento em que ela estiver habilitada, o procedimento de defuzzificação será iniciado, permitindo assim cancelar a execução do Caso em questão. As outras transições em que a lista se encontra vazia, ou seja, nenhuma transição é responsável pelo cancelamento delas, nenhum disparo incerto pode ser realizado.

Antes de explicar a função $funAut$ e a parte que se modifica nas funções G e C devido à característica particular do modelo, o significado da variável $cteCaso$ deve ser descrito. Esta variável define uma lista com 2 constantes inteiras para cada Caso em execução. Esta lista determina se o pedido de cancelamento foi solicitado (1) ou não (0) e, caso sim, o tempo de chegada desta solicitação.

A função $funAut$, descrita abaixo, é responsável por agrupar todas as funções de autorização associadas às transições. Ela faz uso da variável $ILPn$, a qual contém, caso exista, as solicitações de cancelamento dos Casos em execução. Desta forma, através da função $LIST$, é possível obter a lista referente à provável solicitação de cancelamento da execução do Caso com identificador idC e, posteriormente, através da função hd, obter a veracidade da solicitação e o tempo de chegada desta solicitação. Assim sendo, as possíveis avaliações são:

- se algum disparo incerto foi realizado ($tyF = 2$ ou $tyF = 3$), ou seja, um cancelamento foi solicitado para o Caso com identificador igual à $idCF$, então, as funções de autorização associadas às transições com identificadores inferiores à 11 serão avaliadas como incerta, retornando o valor 2, enquanto a transição com identificador igual

à 2 não estiver habilitada. Quando isto ocorrer, a função de autorização associada a esta transição será avaliada como verdadeira, retornando o valor 1;

- se nenhum disparo incerto foi realizado mas o cancelamento foi solicitado ($cnl = 1$), o tempo corrente é maior ou igual ao tempo de chegada desta solicitação ($tm >= tmp$) e a transição a ser analisada se encontra dentro da área de cancelamento ($t >= 2$ e $t < 11$), então, a função de autorização associada a t é avaliada como incerta, retornando o valor 2, exceto se o identificador da transição for 2. Neste caso, a função de autorização é avaliada como verdadeira, retornando o valor 1;

- se a verificação acima não é válida, deve-se verificar se todos os eventos necessários para disparar a transição t já chegaram, ou seja, $maxTC$ e $minTC$ não são negativos e o tempo corrente tm é igual ao tempo da ocorrência do último evento ($tm >= maxTC$); caso sim, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1, exceto se o identificador da transição for 2. Neste caso, a função de autorização é avaliada como falsa, retornando o valor 3;

- se nenhuma das três possibilidades acima descritas é verificada, então, a função de autorização associada a t é avaliada como falsa, retornando o valor 3.

```
fun funAut(tyF,idC,idCF,t,tm,ILPn,minTC,maxTC) =
  if ((tyF = 2) orelse (tyF = 3))
  the if (idC = idCF)
      then if (t = 2)
           then 1
           else if (t >= 11)
                then 3
                else 2
      else 3
  else let val cteCASO = LIST(idC,ILPn);
           val cnl = hd(cteCASO)
           val tmp = hd(tl(cteCASO))
       in if ((cnl = 1) andalso (tm >= tmp)) andalso
             ((t >= 2) andalso (t < 11))
          then if (t = 2)
               then 1
               else 2
          else if (tm >= maxTC) andalso (minTC > (~1))
               then if(t = 2)
                    then 3
                    else 1
```

```
                else 3
        end
```

A parte que se modifica, tanto na função G quanto na função C, devido à característica particular do modelo, é dada abaixo:

```
val idCF = if (Lm = []) then 0 else #1(hd(Lm));
val (minTC, maxTC) = MinMax((aEv, aEv),TL);
val av = funAut(tyF,idC,idCF,t,tm,ILPn,minTC,maxTC);
```

Note-se que foi necessário o uso de duas outras funções para completar a lista de parâmetros usadas na função $funAut$. A primeira, hd, retorna o produto (idC, LPT) referente ao último Caso a ter um disparo realizado permitindo, assim, atribuir o identificador dele à variável $idCF$. Se nenhum disparo foi realizado, o valor de $idCF$ é zero. A outra função, $MinMax$, é utilizada para definir o tempo de chegada do primeiro ($minTC$) e do último evento ($maxTC$) relacionados aos objetos que habilitam t.

C.4 IOWF-net Possibilística apresentada na Seção 6.2.1

Nesta Seção, os valores das três constantes $qtdeTr$, pRP e tRP, as funções de autorização e a parte que se modifica nas funções G e C, referente ao modelo implementado no CPN Tools da versão simplificada de um processo que precede a apresentação de um artigo em uma conferência definido por (AALST, 1998b) e apresentado, respectivamente, nas Figuras 53 e 54 são dados a seguir.

Considerando os identificadores dos lugares e das transições, os valores atribuídos às constantes $qtdeTr$, pRP e tRP usadas neste modelo são:

```
val qtdeTr = 18: INT;

val pRP = [([],          [1],     []),
           ([1],         [2],     []),
           ([2],         [3,4],   []),
           ([3],         [5,6],   []),
           ([5],         [7,8],   []),
           ([7],         [9,10],  []),
           ([4,6,8,9,10],[],      []),

           ([],          [11],    []),
           ([11],        [12],    []),
           ([12],        [13],    []),
           ([13],        [14,15],[]),
```

```
                ([14],          [16,17],[]),
                ([16],          [18],   []),
                ([15,17,18],    [],     [])]  : PN;

val tRP = [([1], [2], []),
           ([2], [3], [3]),
           ([3], [4], [5]),
           ([3], [7], []),
           ([4], [5], [7]),
           ([4], [7], []),
           ([5], [6], []),
           ([5], [7], []),
           ([6], [7], []),
           ([6], [7], []),

           ([8], [9], []),
           ([9], [10],[]),
           ([10],[11],[]),
           ([11],[12],[]),
           ([11],[14],[]),
           ([12],[13],[]),
           ([12],[14],[]),
           ([13],[14],[])]  : PN;
```

Observe na variável tRP, através da terceira lista de cada produto, que somente as transições com identificadores 2, 3 e 5 podem ser disparadas de forma incerta. Nas outras transições, esta lista se encontra vazia. Isto indica que nenhuma transição é responsável pelo cancelamento delas e, consequentemente, nenhum disparo incerto pode ser realizado nelas. Além disto, pode-se notar que os lugares de comunicação não são inseridos nas listas dos lugares de entrada ou dos de saída da transição em questão dado que são considerados como variáveis booleanas associadas à função de autorização ou à ação de uma dada transição.

Antes de explicar a função $funAut$ e a parte que se modifica nas funções G e C devido à característica particular do modelo, o significado da variável $cteCaso$ deve ser descrito. Esta variável define uma lista para cada Caso em execução com 18 constantes inteiras usadas nas funções de autorização. Esta lista determina se uma transição pode ou não ser pseudo-disparada (permitido quando o valor da constante inteira determinada para a transição em questão é maior que 0 (zero)). Além disto, para a transição $receive\_reject$ esta constante determina o momento em que a mesma deve ser disparada de forma certa, caso ainda não tenha sido, e, para a transição too_late, o momento que o disparo da

mesma é permitido.

A função $funAut$, descrita abaixo, é responsável por agrupar todas as funções de autorização associadas às transições. Ela faz uso da variável $ILPn$, a qual contém os tempos definidos para todos os Casos em execução. Desta forma, através da função $LIST$, é possível obter a lista dos tempos definidos para o Caso com identificador idC e, posteriormente, através da função nth, obter o tempo relacionado à transição t. Assim sendo, as possíveis avaliações são:

- se a transição habilitada corresponde à transição *too_late* ($t = 17$), então, a função de autorização será avaliada como verdadeira, retornando 1, apenas se o tempo máximo de espera para enviar a versão final do rascunho é alcançado pelo tempo corrente ($tm >= maxT$). Caso não, a função de autorização sempre será avaliada como falsa, retornando 3;

- se a transição habilitada corresponde à transição *receive_reject* ($t = 4$), o evento associado a ela ainda é falso ($minTC = (-1)$) (ou seja, o evento referente à rejeição do resumo foi perdido) e o tempo corrente alcançou o tempo limite definido em $maxT$ ($tm = maxT$), então, a função de autorização será avaliada como verdadeira, retornando 1. Caso não, o próximo item será responsável por determinar a avaliação da função de autorização para esta transição;

- se todos os eventos necessários para disparar a transição t já chegaram, ou seja, $maxTC$ e $minTC$ não são negativos e o tempo corrente tm é igual ao tempo da ocorrência do último evento ($tm = maxTC$), então, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1;

- se o tempo máximo de espera é maior que zero (ou seja, é permitido um disparo incerto) e o tempo corrente alcançou este tempo máximo ($tm = maxT$), então, a função de autorização associada a t é avaliada como incerta, retornando o valor 2;

- se nenhuma das quatro possibilidades acima descritas é verificada, então, a função de autorização associada a t é avaliada como falsa, retornando o valor 3.

```
fun funAut(t, idC, ILPn, Lm, tm, minTC, maxTC) =
  if (t > 0)
  then let val cteCASO = LIST(idC, ILPn);
           val maxT = nth(t, cteCASO);
       in if (t = 17)
          then if (tm >= maxT)
               then 1
               else 3
          else if (t = 4) andalso (tm >= maxT)
```

```
                then 1
                else if (tm >= maxTC) andalso (minTC > (~1))
                     then let val lpt = LIST(idC,Lm)
                          in if (t = 5) andalso
                                eLPT(2, #1(nth(t,tRP)), lpt)
                             then 2
                             else if (t = 7) andalso uM(t, lpt)
                                  then 3
                                  else 1
                          end
                     else if (maxT > 0) andalso (tm >= maxT)
                          then 2
                          else 3
        end
   else 3
```

A parte que se modifica, tanto na função G quanto na função C, devido à característica particular do modelo, é dada abaixo:

```
val (minTC, maxTC) = MinMax((aEv, aEv), TL);
val av = funAut(t,idC,ILPn,Lm,tm,minTC,maxTC);
```

Note-se que foi necessário o uso de outra função para completar a lista de parâmetros usadas na função de autorização $funAut$. A função $MinMax$ é utilizada para definir o tempo de chegada do primeiro ($minTC$) e do último evento ($maxTC$) relacionados aos objetos que habilitam t.

C.5 IOWF-net Possibilística apresentada na Seção 6.2.2

Nesta Seção, os valores das três constantes $qtdeTr$, pRP e tRP, as funções de autorização e a parte que se modifica nas funções G e C, referente ao modelo implementado no CPN Tools da versão simplificada de um processo que precede a apresentação de um artigo em uma conferência definido por (AALST, 1998b) e apresentado, respectivamente, nas Figuras 56 e 57 são dados a seguir.

Considerando os identificadores dos lugares e das transições, os valores atribuídos às constantes $qtdeTr$, pRP e tRP usadas neste modelo são:

```
val qtdeTr = 17: INT;

val pRP = [([],      [1],    []),
           ([1],     [2],    []),
```

```
                ([2],          [3,4],    []),
                ([3],          [5,6],    []),
                ([5],          [7,8],    []),
                ([7],          [9],      []),
                ([4,6,8,9],    [],       []),

                ([],           [10],     []),
                ([10],         [11],     []),
                ([11],         [12],     []),
                ([12],         [13,14],  []),
                ([13],         [15,16],  []),
                ([15],         [17],     []),
                ([14,16,17],   [],       [])]  : PN;

val tRP = [([1],  [2],  []),
           ([2],  [3],  []),
           ([3],  [4],  []),
           ([3],  [7],  []),
           ([4],  [5],  []),
           ([4],  [7],  []),
           ([5],  [6],  [9]),
           ([5],  [7],  []),
           ([6],  [7],  []),

           ([8],  [9],  []),
           ([9],  [10], []),
           ([10], [11], []),
           ([11], [12], []),
           ([11], [14], []),
           ([12], [13], []),
           ([12], [14], []),
           ([13], [14], [])]  : PN;
```

Observe na variável *tRP*, através da terceira lista de cada produto, que somente a transição com identificador 7 pode ser disparada de forma incerta. Este disparo incerto permite evitar a ocorrência de um estado de *deadlock* dado que ele ou é finalizado quando o reconhecimento do recebimento da versão final por parte do modelo do processo *PC* é enviado ao modelo do processo *AU*, ou cancelado quando o tempo limite para o envio da versão final é alcançado. Desta forma, o possível erro de sincronização após o envio dos eventos *too_late* e *final_version* é contornado permitindo concluir a execução do Caso

em questão. Além disto, pode-se notar que os lugares de comunicação não são inseridos nas listas dos lugares de entrada ou dos de saída da transição em questão dado que são considerados como variáveis booleanas associadas à função de autorização ou à ação de uma dada transição.

Antes de explicar a função $funAut$ e a parte que se modifica nas funções G e C devido à característica particular do modelo, o significado da variável $cteCaso$ deve ser descrito. Esta variável define uma lista para cada Caso em execução composta por uma única constante inteira, a qual é usada nas funções de autorização. Esta constante determina o tempo limite para o envio da versão final.

A função $funAut$, descrita abaixo, é responsável por agrupar todas as funções de autorização associadas às transições. Ela faz uso da variável $ILPn$, a qual contém o tempo limite para o envio da versão final definido para todos os Casos em execução. Desta forma, através da função $LIST$ em conjunto com a função hd, é possível obter este tempo limite caso o identificador da transição habilitada seja igual à 16. Assim sendo, as possíveis avaliações são:

- se a transição habilitada corresponde à transição too_late ($t = 16$), então, a função de autorização será avaliada como verdadeira, retornando 1, apenas se o tempo máximo de espera para enviar a versão final do rascunho é alcançado pelo tempo corrente ($tm >= maxT$). Caso não, a função de autorização sempre será avaliada como falsa, retornando 3;

- se todos os eventos necessários para disparar a transição t já chegaram, ou seja, $maxTC$ e $minTC$ não são negativos e o tempo corrente tm é igual ao tempo da ocorrência do último evento ($tm = maxTC$), então, a função de autorização associada a t é avaliada como verdadeira, retornando o valor 1, exceto se a transição for identificada pelo identificador 7. Neste caso, a função de autorização associada a t é avaliada como incerta, retornando o valor 2;

- por fim, se nenhuma das duas possibilidades acima descritas é verificada, então, a função de autorização associada a t é avaliada como falsa, retornando o valor 3.

```
fun funAut(t, idC, ILPn, tm, minTC, maxTC) =
  if (t > 0)
  then if (t = 16)
       then let val cteCASO = LIST(idC,ILPn);
                        val maxT = hd(cteCASO);
            in if (tm >= maxT)
               then 1
               else 3
            end
```

```
        else if (tm >= maxTC) andalso (minTC > (~1))
             then if (t = 7)
                  then 2
                  else 1
             else 3
   else 3
```

A parte que se modifica, tanto na função G quanto na função C, devido à característica particular do modelo, é dada abaixo:

```
val (minTC, maxTC) = MinMax((aEv, aEv), TL);
val av = funAut(t,idC,ILPn,tm,minTC,maxTC);
```

Note-se que foi necessário o uso de outra função para completar a lista de parâmetros usadas na função de autorização $funAut$. A função $MinMax$ é utilizada para definir o tempo de chegada do primeiro ($minTC$) e do último evento ($maxTC$) relacionados aos objetos que habilitam t.