Verificação de Requisitos Funcionais e não Funcionais em Arquiteturas Orientadas a Serviços

Kênia Santos de Oliveira

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

Uberlândia
2020

Kênia Santos de Oliveira

Verificação de Requisitos Funcionais e não Funcionais em Arquiteturas Orientadas a Serviços

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 Doutora em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Stéphane Julia

Uberlândia
2020

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese de doutorado, 33/2020, PPGCO				
Data:	03 de dezembro de 2020	Hora de início:	14:04	Hora de encerramento:	17:30
Matrícula do Discente:	11613CCP002				
Nome do Discente:	Kênia Santos de Oliveira				
Título do Trabalho:	Verificação de Requisitos Funcionais e não Funcionais em Arquiteturas Orientadas a Serviços				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Engenharia de Software				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Pedro Frosi Rosa - FACOM/UFU, Carlos Roberto Lopes - FACOM/UFU, José Reinaldo Silva - USP, Ricardo Lüders - UTFPR e Stéphane Julia - FACOM/UFU orientador da candidata.

Os examinadores participaram desde as seguintes localidades: José Reinaldo Silva - São Paulo/SP; Ricardo Lüders - Curitiba/PR; Pedro Frosi Rosa, Carlos Roberto Lopes e Stéphane Julia - Uberlândia/MG. A discente participou da cidade de Catalão/GO.

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

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

Aprovada.

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

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

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

Documento assinado eletronicamente por **Pedro Frosi Rosa**, **Professor(a) do Magistério Superior**, em 04/12/2020, às 14:06, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Stéphane Julia**, **Professor(a) do Magistério Superior**, em 04/12/2020, às 14:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ricardo Luders**, **Usuário Externo**, em 04/12/2020, às 14:22, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Jose Reinaldo Silva**, **Usuário Externo**, em 04/12/2020, às 19:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Carlos Roberto Lopes**, **Professor(a) do Magistério Superior**, em 05/12/2020, às 09:49, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2426578** e o código CRC **1BF99372**.

Referência: Processo nº 23117.071957/2020-81 SEI nº 2426578

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

O48 Oliveira, Kênia Santos de, 1985-
2020 Verificação de Requisitos Funcionais e não Funcionais em Arquiteturas Orientadas a Serviços [recurso eletrônico] / Kênia Santos de Oliveira. - 2020.

Orientador: Stéphane Julia.
Tese (Doutorado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2020.786
Inclui bibliografia.
Inclui ilustrações.

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

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

Resumo

Este trabalho de pesquisa apresenta formalmente dois métodos para a verificação de cenários de requisitos funcionais e não funcionais em Arquiteturas Orientadas a Serviços (SOA) e também dois métodos para a detecção e remoção de requisitos negativos do tipo *deadlock* em SOA. A SOA representa processos de negócio e é modelada por uma *WorkFlow net* interorganizacional que não é necessariamente livre de *deadlock* (bloqueios).

O primeiro método proposto está relacionado com a verificação de cenários dos requisitos de serviços (comportamentais) em SOA. A verificação é baseada na construção das árvores de prova da Lógica Linear e dos grafos de precedência derivados a partir das árvores de prova corretamente finalizadas. Os grafos de precedência dos modelos de requisitos e de arquitetura são comparados por meio de um tipo de bissimulação definida neste trabalho a fim de verificar se todos os cenários que existem no modelo de requisitos também existem na arquitetura correspondente.

O segundo método está relacionado com a verificação do desempenho dos requisitos de serviços em SOA. As árvores de prova da Lógica Linear construídas para o primeiro método são reutilizadas com o acréscimo de datas simbólicas associadas a cada átomo das árvores geradas. No final da execução de cada cenário, intervalos de datas simbólicas são gerados para comparar se os cenários que são equivalentes em termos de comportamento também são equivalentes em termos de desempenho.

O terceiro método está relacionado com a detecção de requisitos negativos do tipo *deadlock* em SOA. A partir de uma marcação indesejada, que representa um estado parcial do modelo, são identificadas todas as sequências de ações, ou seja, todos os cenários que podem tornar um requisito de serviço em um requisito negativo do tipo *deadlock*. Para a identificação destas sequências de ações, é utilizado um raciocínio inverso na arquitetura aplicando a mesma ideia utilizada no método para verificação de requisitos funcionais.

O quarto método está relacionado com o controle do *deadlock*. Para prevenir as situações de *deadlock*, que são causadas pela troca de mensagens em um *workflow* interorganizacional, é utilizada a sincronização dos processos locais. A sincronização força os processos de *workflow* locais executarem simultaneamente certas atividades, removendo,

desse modo, a situação de *deadlock* do modelo.

Os métodos são validados através de um estudo de caso que também é simulado no simulador CPN Tools. A validação e a análise de complexidade realizada, mostram que os métodos podem ser efetivos para identificar se um sistema baseado em SOA satisfaz o comportamento e o desempenho das necessidades de negócio especificadas por um modelo de requisitos público e também para identificar e corrigir requisitos negativos do tipo *deadlock*.

Palavras-chave: *WorkFlow net* Interorganizacional. *t-Time WorkFlow net*. Redes de Petri. Lógica Linear. Arquitetura Orientada a Serviços. Verificação de Requisitos. Bissimulação.

Abstract

This research work formally presents two methods for scenarios verification of functional and non-functional requirements in Service-Oriented Architecture (SOA) models and also two methods for detecting and removing negative requirements of deadlock type in SOA. SOA represents business processes and it is modeled by a Interorganizational WorkFlow net that is not necessarily deadlock-free.

The first method is related to the verification of scenarios of service requirements (behavior) in SOA. The verification is based on the construction of Linear Logic proof trees and precedence graphs derived from proof trees correctly finalized. The precedence graphs of the requirement and architectural models are compared using a type of bisimulation defined in this work in order to verify if all existing scenarios of the requirement model also exist in the corresponding architecture.

The second method is related to the verification of performance of service requirements in SOA. The Linear Logic proof trees constructed for the first method are reused with the addition of symbolic dates associated with each atom of the produced trees. At the end of the execution of each scenario, symbolic date intervals are generated to compare if the scenarios that are equivalents in terms of behavior are also equivalents in terms of performance.

The third method is related to the detection of negative requirements of deadlock type in SOA. Starting from a feared marking, which represents a partial state of the model, all sequences of actions will be identified, that is, all scenarios that can turn a service requirement into a negative requirement of deadlock type. To identify these sequences of actions, an inverse reasoning will be used in the architecture model, applying the same idea used in the method to verify functional requirements.

The fourth method is related to deadlock control. To prevent deadlock situations that are caused by the exchange of messages in an interorganizational workflow, the synchronization of local processes will be used. Synchronization forces local workflow processes to perform certain activities simultaneously; thereby, removing the deadlock situation from the model.

The methods are validated through a case study that is also simulated in the CPN Tools simulator. The validation and the complexity analysis performed, show that the methods can be effective to identify whether an SOA-based system satisfies the behavior and performance of the business needs specified by a public requirements model and also to identify and correct negative requirements of deadlock type.

Keywords: Interorganizational WorkFlow net. t-Time WorkFlow net. Petri net. Linear Logic. Service Oriented Architecture. Requirements Verification. Bisimulation.

Lista de Ilustrações

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Colaboração mediada por computador é um assunto de interesse crescente entre uma variedade de diferentes organizações. No trabalho cooperativo, organizações colaboram entre si com os recursos que cada uma possui para alcançarem objetivos em comum. Esse ambiente colaborativo exige das organizações que seus processos de negócio sejam gerenciados de forma eficiente e eficaz. Nesse contexto, a área de Gerenciamento de Processos de Negócio, do inglês *Business Process Management* (BPM), tem recebido considerável atenção pelo seu potencial em aumentar significativamente a produtividade e economizar gastos (HOFSTEDE et al., 2009). BPM é considerado como um meio flexível e eficaz de analisar, modelar, controlar e otimizar operações de negócio utilizando métodos, técnicas e ferramentas tendo a tecnologia da informação como suporte.

Para representar os processos de colaboração que envolvem múltiplas organizações, processos de *workflow* interorganizacionais têm sido considerados. Um processo de *workflow* interorganizacional permite que organizações com habilidades complementares executem trabalhos que não estão dentro do alcance de uma única organização (CHEBBI; DUSTDAR; TATA, 2006). Essencialmente, um processo de *workflow* interorganizacional é um conjunto de processos de *workflow* locais (pertencentes as diferentes organizações parceiras do negócio) fracamente acoplados envolvidos em um mesmo processo de *workflow* global (AALST, 2000). Segundo o modelo de referência da *Workflow Management Coalition* (HOLLINGSWORTH, 1994), processos de *workflow* estão relacionados com a automatização de procedimentos em que documentos, informações ou tarefas são passados entre participantes de acordo com um conjunto definido de regras para alcançar ou contribuir com um objetivo geral do negócio.

Conforme apresentado por (AALST, 2000), um *workflow* interorganizacional pode ser representado por uma *WorkFlow net* sendo, nesse contexto, chamada de *WorkFlow net* interorganizacional. Uma *WorkFlow net* (AALST, 1996a) é um modelo formal [1] de

[1] Métodos formais manipulam uma descrição matemática precisa de um sistema de software com o propósito de estabelecer que o sistema exiba ou não algumas propriedades precisamente definidas (DILLON; SANKAR, 1997).

representação de processos de negócio baseado nas redes de Petri. Uma rede de Petri (MURATA, 1989) é um modelo formal e abstrato, com representação gráfica, que pode ser utilizado para a modelagem de diversos tipos de sistemas a eventos discretos. Além das redes de Petri, outras linguagens de modelagem podem ser utilizadas para representar processos de *workflow* como, por exemplo, BPMN (*Business Process Model and Notation*) (MPMN, 2013), Diagramas de Atividade da UML (*Unified Modeling Language*) (UML, 2015), YAWL (*Yet Another Workflow Language*) (AALST; HOFSTEDE, 2005), WDL (*Workflow Description Language*). No entanto, van der Aalst (AALST, 1998a) apresenta alguns motivos para utilizar as redes de Petri na modelagem de processos de *workflow* como: semântica formal, natureza gráfica, expressividade, variedade de propriedades já demonstradas, disponibilidade de várias técnicas de análise e o fato de ser uma ferramenta não proprietária. Por ser considerada uma ferramenta eficiente na modelagem e análise de processos de *workflow* por diferentes autores, as redes de Petri foram utilizadas em vários trabalhos como, por exemplo, em (LI; FAN; ZHOU, 2003), (ŢIPLEA; MARINESCU, 2005), (PASSOS; JULIA, 2009) e (AALST et al., 2011).

Um processo de *workflow* interorganizacional está inserido em um ambiente de computação distribuída. As diferentes organizações envolvidas no *workflow* interorganizacional distribuem seus serviços os quais devem operar através dos limites organizacionais; portanto, há a necessidade de se trabalhar com processos de *workflow* distribuídos por meio de complexos mecanismos de colaboração a fim de alcançar objetivos em comum. Esse fato levou as pesquisas em *workflow* a um novo patamar voltado para a definição de arquiteturas distribuídas de execução de processos. Nesse contexto, a Computação Orientada a Serviços, do inglês *Service Oriented Computer* (SOC), tem se destacado. SOC é um termo genérico que representa uma nova geração de plataforma de computação distribuída (ERL, 2009). Este paradigma de computação utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações/soluções (PAPAZOGLOU, 2003). Para construir um modelo de serviços, SOC baseia-se na Arquitetura Orientada a Serviços, do inglês *Service Oriented Architecture* (SOA), a qual é uma forma de reorganizar aplicações de software e infraestrutura em um conjunto de serviços que interagem (PAPAZOGLOU, 2003). SOA tem sido amplamente utilizada com o objetivo de integrar sistemas através de serviços que podem ser reusáveis por vários sistemas. O modelo arquitetural estabelecido por SOA objetiva melhorar a eficiência, agilidade e produtividade de um negócio posicionando serviços como meios primários (ERL, 2009).

Um serviço é uma capacidade de negócio da organização que é implementado e disponibilizado em um ambiente distribuído, na internet ou intranet, para que outras aplicações possam acessá-lo (PAPAZOGLOU, 2003). Cada serviço é composto por um conjunto de capacidades relacionadas com o contexto funcional que lhe é atribuído. Essas capacidades, adequadas para invocação por programas de consumo externos, são normalmente expressas através de uma publicação de contrato de serviço (ERL, 2009).

Segundo Zernadji et al. (ZERNADJI et al., 2016), a engenharia de aplicações orientadas a serviços ainda não é madura e levanta muitas questões desafiadoras: por exemplo, como satisfazer a qualidade de requisitos nesse tipo de processo de engenharia. Uma questão importante em Engenharia de Software é garantir que uma proposta de arquitetura de software reproduza o comportamento do modelo de especificação de requisitos. Ao longo do desenvolvimento de um software, a arquitetura pode se distanciar dos requisitos definidos no modelo de análise; por isso, verificar que o comportamento do modelo de requisitos existe na arquitetura correspondente minimiza riscos de falhas em projetos, aumenta a garantia da qualidade de software e evita custos com retrabalho (GOKNIL; KURTEV; BERG, 2014). A verificação dos requisitos na arquitetura correspondente a um modelo de análise garante que a abordagem apresentada produzirá um sistema que alcance os requisitos exigidos pelo cliente. Assim, é de grande interesse propor uma abordagem que verifique se os requisitos definidos em um modelo de análise, tanto funcionais quanto não funcionais, também estão presentes em uma SOA.

Diante das considerações expostas, a proposta apresentada nesse trabalho de pesquisa tem como objetivo principal utilizar *workflow* interorganizacionais para representar uma SOA no contexto de processos de negócio e mostrar que os cenários existentes no modelo de especificação de requisitos estarão presentes também na arquitetura correspondente em termos de comportamento e desempenho. Para mostrar a equivalência comportamental entre os dois modelos (requisitos e arquitetura), será então necessário propor uma definição de equivalência semântica entre modelos distintos, como já foi feito por exemplo no contexto das álgebras de processos com a noção de bissimulação (BASTEN, 1998). Também é objetivo deste trabalho de pesquisa identificar cenários que podem levar um requisito de serviço em um requisito negativo do tipo *deadlock*. Para isso, a partir de uma marcação indesejada (estado de *deadlock*), todos os cenários responsáveis por alcançar o estado de *deadlock* serão produzidos por meio das árvores de prova da Lógica Linear. Baseado nos cenários que correspondem aos comportamentos negativos, regras de sincronização serão propostas para remover as situações de *deadlock* causadas pelas trocas de mensagens entre os processos do modelo arquitetural.

1.1 Motivação

Em um ambiente de negócios, transformar requisitos de negócio em uma especificação de sistema é uma tarefa crucial de qualquer projeto de Engenharia de Software (WEIDLICH; MENDLING; WESKE, 2011a). Portanto, os modelos de processos desempenham um papel importante permitindo criar uma abstração de como funciona um negócio, ou seja, fornecendo o entendimento de como são realizadas as diversas atividades contidas em cada processo.

Alguns estudos, como (BOUKHEDOUMA et al., 2013), (AALST; WESKE, 2013),

(BOUKHEDOUMA et al., 2014) e (HUANG et al., 2014), mostram a relação de SOA com *workflow* interorganizacional. Desse modo, como forma de representação dos serviços, inclusive os mecanismos de comunicação entre processos distintos, *workflow* interorganizacional pode ser utilizado. Em particular, um *workflow* interorganizacional oferece a uma empresa a oportunidade de reconfiguração dos processos de negócio além dos limites de sua própria organização. As organizações envolvidas são essencialmente autônomas e têm a liberdade de criar ou modificar seus respectivos *workflows* a qualquer momento. Assim, em um processo de *workflow* interorganizacional há uma forte necessidade de coordenação para otimizar o fluxo de trabalho dentro e entre as diferentes organizações (AALST, 2003). Como um *workflow* interorganizacional pode ser representado por uma *WorkFlow net*, a verificação de processos de *workflow* pode ser realizada formalmente, visto que uma *WorkFlow net* é baseada numa rede de Petri. As redes de Petri tornaram-se referência na área de modelagem e análise formais de processos de *workflow* permitindo, entre outras características, a verificação formal de boas propriedades do modelo. Segundo van der Aalst (AALST, 1998a), é muito importante verificar os processos de *workflow* antes de colocá-los em produção, evitando assim uma definição de processos que contenha erros. Embora estudos como (HUANG et al., 2014), (DING et al., 2016) e (DU; LI; XIONG, 2012) considerem *WorkFlow nets* para modelar uma SOA, em geral, quando esses trabalhos apresentam algum tipo de análise no modelo, eles são baseados nos grafos de alcançabilidade. A desvantagem de considerar grafos de alcançabilidade é que pode ocorrer uma explosão do número de estados discretos, o que geralmente leva a uma complexidade elevada nos algoritmos de verificação utilizados.

Uma das propriedades que podem ser verificadas em um processo de *workflow* é a propriedade *Soundness*. Segundo van der Aalst (AALST, 1998a), a propriedade *Soundness* é considerada como o principal critério de correção para as *WorkFlow nets*, sendo que sua verificação garantirá que: se uma instância de processo começou a ser tratada, o tratamento desta será finalizado; após a finalização do tratamento, não haverá nenhuma pendência não tratada no processo para tal instância; e não existirá tarefas que não serão executadas (tarefas "mortas") em nenhuma instância do processo de *workflow*. Apesar da importância do critério de correção *Soundness*, na prática, na maioria das vezes, os processos de *workflow* não satisfazem tal critério (FAHLAND et al., 2009). Por isso, nos casos de *workflow* interorganizacionais não *sound*, podem ser consideradas as variantes do critério *Soundness*, como *Relaxed Soundness* e *Weak Soundness* (PASSOS, 2016).

Alguns trabalhos mostraram a relação entre a teoria das redes de Petri e a Lógica Linear como, por exemplo, em (GIRAULT; PRADIER-CHEZALVIEL; VALETTE, 1997) e (RIVIERE et al., 2001), uma vez que existe uma tradução quase direta entre a estrutura de uma rede de Petri e um sequente da Lógica Linear. Uma das vantagens de representar os modelos de redes de Petri com a Lógica Linear é que, conforme mostrado em (PASSOS; JULIA, 2014) e (PASSOS; JULIA, 2013), com a prova dos sequentes da Lógica Linear é

possível verificar o critério de correção *Soundness* para *WorkFlow net* interorganizacional sem considerar o grafo de alcançabilidade como acontece em abordagens mais clássicas no contexto da teoria das redes de Petri.

Em qualquer projeto de Engenharia de Software é fundamental que os requisitos definidos nos modelos de análise sejam verificados na arquitetura do software. A verificação dos requisitos na arquitetura permitirá confirmar se os comportamentos planejados inicialmente serão ou não contemplados pela arquitetura. Caso essa verificação inicial não seja realizada e erros sejam propagados para as fases de implementação e testes do ciclo de vida de desenvolvimento de software, a correção será mais difícil e mais cara quando detectada. Segundo Hoyos et al. (HOYOS; CASALLAS; JIMÉNEZ, 2012), a verificação de requisitos funcionais e não funcionais durante todo o processo do projeto de software é uma solução econômica quando comparado a um processo de validação de teste de um produto já compilado. Portanto, assim como para qualquer tipo de projeto de arquitetura de software, é crucial garantir que o projeto de Arquitetura Orientada a Serviços atenda adequadamente as especificações de requisitos, necessitando, desse modo, que estes sejam verificados já nas atividades de modelagem.

As abordagens apresentadas na literatura em relação à verificação de requisitos em modelos arquiteturais, lidam com variados tipos de arquiteturas sendo que, em sua maior parte, apresentam algum tipo de rastreabilidade de requisitos de maneira informal, como pode ser observado, por exemplo, em (MATÉ; TRUJILLO, 2014), (TRUBIANI; GHABI; EGYED, 2017) e (OLIVEIRA; SOARES, 2013). Em (MATÉ; TRUJILLO, 2014), foi proposta uma abordagem para rastreabilidade com o objetivo de registrar explicitamente o relacionamento entre elementos no nível conceitual em *data warehouses* utilizando a arquitetura orientada a modelos. Em (TRUBIANI; GHABI; EGYED, 2017), o objetivo foi automatizar a rastreabilidade entre modelos de arquitetura de software e requisitos não funcionais, como desempenho e segurança, para assim apoiar os arquitetos de software na identificação das causas que provavelmente mais contribuem para a violação de requisitos não funcionais. O trabalho de (OLIVEIRA; SOARES, 2013) utiliza a SysML (*Systems Modeling Language*) para modelar aspectos no nível de requisitos e utiliza um diagrama de requisitos estendido que fica acoplado a outros modelos com o objetivo de representar rastros entre a especificação de requisitos e o projeto de software. No contexto de SOA, por ser considerado um tópico de pesquisa ainda emergente (ZERNADJI et al., 2016), a verificação de requisitos em modelos de arquitetura ainda é uma questão desafiadora. No trabalho de (ZERNADJI et al., 2016), é apresentado um método que objetiva assistir arquitetos de software na orquestração de serviços *Web* em integrar requisitos de qualidade em seus artefatos. Essa abordagem não considera a verificação de requisitos definidos no modelo de análise, apenas define um método para integrar requisitos não funcionais (de qualidade) nos processos de negócio representados por serviços *Web*. Em (AALST, 2003), é apresentada uma abordagem para representar processos de *workflow* interorgani-

zacionais com o objetivo de garantir que a implementação local de um *workflow* não cria nenhum tipo de anomalia sobre os limites organizacionais. Devido as suas características, a abordagem pode ser aplicada em SOA; no entanto, é uma abordagem baseada em regras de construção definidas pelo autor. Na prática, organizações constroem seus processos de *workflow* sem se preocuparem muito com regras.

Considerando que em um projeto orientado a serviços os modelos de especificação de requisitos e de arquitetura possam ser representados por *workflow* interorganizacional, a verificação dos requisitos na arquitetura pode ser realizada por algum tipo de comparação entre os modelos, permitindo, assim, que suas correspondências sejam verificadas. Em (EUZENAT; SHVAIKO, 2013), é apresentada uma visão geral de técnicas básicas que mostram como correspondências podem ser identificadas entre entidades relacionadas semanticamente de diferentes ontologias [2]. Algumas das técnicas apresentadas em (EUZENAT; SHVAIKO, 2013) são: técnicas baseadas em nomes, baseadas em estrutura, baseadas em semântica e medidas de similaridade. Recentemente, várias publicações mostraram como técnicas de correspondência podem ser aplicadas para modelos de processos de negócios como (DIJKMAN et al., 2009), (EHRIG; KOSCHMIDER; OBERWEIS, 2007), (DONGEN; DIJKMAN; MENDLING, 2008), (WEIDLICH; DIJKMAN; MENDLING, 2010) e (NEJATI et al., 2007).

Perfis comportamentais são apresentados na literatura como uma conveniente forma para comparar pares de modelos de processos em relação ao seu comportamento. No entanto, segundo Polyvyanyy et al. (POLYVYANYY et al., 2016), o poder expressivo de perfis comportamentais é comprometido em algumas situações. Por exemplo, um perfil comportamental pode não capturar o fato de uma atividade 'b' sempre preceder uma atividade 'e' para algumas configurações em um sistema. Além disso, perfis comportamentais não podem ser usados para decidir equivalência de rastros [3] de autômatos finitos e, portanto, de redes de Petri, embora, segundo Aalst et al. (AALST; MEDEIROS; WEIJTERS, 2006), equivalência de rastros é a noção mais fraca de equivalência, pois não capturam os momentos de escolhas e podem ser infinitos. Adicionalmente, segundo Dijkman et al. (DIJKMAN et al., 2011), técnicas existentes para a verificação de equivalência comportamental são principalmente baseadas na análise do espaço de estado, a qual é computacionalmente cara. Mesmo as classes restritas de redes de Petri 1-*safe* [4] requer espaço exponencial para a maioria das noções de equivalência.

A bissimulaçao é uma das várias possibilidades para a formalização de uma relação de equivalência significativa nos processos. De acordo com Basten (BASTEN, 1998), há várias razões para considerar a bissimulação como uma relação de equivalência entre mo-

[2] Modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes (EUZENAT; SHVAIKO, 2013).

[3] Dois modelos são considerados equivalentes se o conjunto de rastros que eles podem executar são idênticos (AALST; MEDEIROS; WEIJTERS, 2006).

[4] Em uma rede de Petri 1-*safe* um lugar pode ser marcado por no máximo uma ficha (CHENG; ESPARZA; PALSBERG, 1993).

delos comportamentais, como: possui uma definição intuitiva e fácil de entender, preserva algumas propriedades importantes de sistemas concorrentes tais como *deadlock*. Em particular, processos que são bissimilares também são equivalentes quando são consideradas semânticas de sistemas concorrentes. A noção de bissimulação é rigorosa uma vez que considera os instantes no tempo que correspondem à tomadas de decisões. Mesmo a equivalência de bissimulação fraca (do inglês *weak bisimilarity*) é mais rigorosa do que a equivalência de rastros (DIJKMAN et al., 2011). Na equivalência de rastros, dois processos são considerados equivalentes se e somente se eles podem executar exatamente as mesmas sequências de ações, já na equivalência por bissimulação, dois processos são bissimilares se eles podem realizar as mesmas ações e atingir estados bissimilares (NICOLA, 2011). Uma bissimulação é uma noção mais fraca do que o isomorfismo (uma relação de bissimulação não precisa ser 1-1), mas é suficiente para garantir a equivalência no processamento. Grafos isomórficos, por exemplo, possuem o mesmo número de nós, as estruturas isomórficas devem ser essencialmente as mesmas, ou seja, algebricamente idênticas, o que não precisa ser o caso para grafos bissimilares (SANGIORGI, 2009). De acordo com (SANGIORGI, 2009), bissimulação é derivada da noção de isomorfismo, intuitivamente com o objetivo de obter relações mais grosseiras que o isomorfismo, mas ainda com a garantia de que os conjuntos relacionados tenham a "mesma" estrutura interna.

Na próxima seção, os objetivos desse trabalho de pesquisa são detalhados.

1.2 Objetivos e Desafios da Pesquisa

Considerando os aspectos discutidos na seção 1.1, a pesquisa descrita nesta tese de doutorado tem como objetivo geral utilizar *WorkFlow nets* interorganizacionais para representar uma proposta de SOA e verificar se os cenários obtidos no modelo de especificação de requisitos serão reproduzidos na arquitetura. Para que este objetivo seja alcançado, os modelos que representam os requisitos e a arquitetura precisam ser de alguma forma comparados. Segundo Aalst et al. (AALST; MEDEIROS; WEIJTERS, 2006), para comparar modelos de processos, é necessário assumir que esses modelos possuem algum tipo de semântica operacional e, portanto, é necessário também assumir a existência de alguma noção de equivalência. Assim, é necessário definir uma noção de equivalência de semântica operacional entre os modelos de processos baseados em *WorkFlow net* para que os modelos que representam os requisitos e a arquitetura sejam comparados. Para isso, será considerada neste trabalho a noção de equivalência do tipo bissimulação *branching* (GLABBEEK; WEIJLAND, 1989) que é uma variação de bissimulação (ou conhecida também como bissimulação *strong*). Quando um sistema pode executar etapas internas (silenciosas), das quais o impacto é considerado não observável, a bissimulação *strong* não é apropriada, pois não diferencia comportamentos internos (silenciosos) de comportamentos externos (observáveis). Já a bissimulação *branching* abstrai os passos internos de um

processo, mas exige que comportamentos visíveis (externos) sejam fortemente simulados. Com a definição de uma noção de equivalência de semântica operacional, é preciso definir também um algoritmo de verificação da equivalência em relação aos requisitos funcionais, no que se refere ao comportamento dos modelos.

Em uma arquitetura, provavelmente, existirão comportamentos adicionais não previstos no modelo de requisitos e que poderão levar eventualmente a situações não previstas durante a análise, como por exemplo, situações de *deadlock*. No entanto, mesmo que haja *deadlock* em dois *workflows* que se comunicam, podem existir vários cenários que permitem que esses *workflows* finalizem e que o conjunto desses cenários é o suficiente para cobrir os requisitos do sistema do modelo de análise. Assim, o método proposto deve permitir a verificação dos requisitos na arquitetura tanto em processos *sound* e não *sound*. Nos casos não *sound*, deverão ser consideradas as variantes do critério *Soundness*, que são *Relaxed Soundness* e *Weak Soundness* (PASSOS, 2016).

Além de requisitos funcionais, pretende-se também verificar requisitos não funcionais. Um dos requisitos não funcionais considerado é em relação ao desempenho. Nesse caso, serão verificados os tempos de resposta relacionados às atividades dos processos.

É possível considerar também que os processos sejam tolerantes a falhas pois, como já citado, mesmo que haja situações de *deadlock*, podem existir vários cenários que permitem que os processos finalizem corretamente e que atendem os requisitos do sistema do modelo de análise. Além disso, ao detectar as situações de *deadlock* que tornam os processos não *sound*, é possível considerar a possibilidade de remover os *deadlocks* e transformar os processos não *sound* em processos seguros e confiáveis. Neste caso, é importante que os requisitos funcionais sejam mantidos e que o desempenho não diminua consideravelmente. Portanto, o outro requisito não funcional que será tratado neste trabalho é em relação à confiabilidade dos processos.

Os objetivos podem ser resumidos como:

- Objetivo Geral
  Verificar que uma SOA reproduz os requisitos definidos em um modelo de especificação de requisitos.

- Objetivos específicos

  1. Representar uma SOA por meio de *WorkFlow nets* interorganizacionais.
  2. Representar os modelos de especificação de requisitos de uma SOA por meio de *WorkFlow nets*.
  3. Definir como se dará a equivalência de semântica operacional entre os modelos de processos baseados em *WorkFlow net* e Lógica Linear considerando a noção de equivalência do tipo bissimulação *branching*.

4. Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se os requisitos funcionais, em relação ao comportamento dos modelos, definidos num modelo de análise de requisitos estão presentes na arquitetura correspondente. Tal algoritmo deverá permitir analisar cenários tanto em modelos *sound* quanto não *sound*.
5. Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se os requisitos não funcionais, em relação essencialmente ao desempenho de uma SOA, definidos em um modelo temporal de análise de requisitos estão presentes na arquitetura correspondente. O algoritmo deverá permitir analisar cenários tanto em modelos *sound* quanto não *sound*.
6. Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para detectar possíveis situações que podem levar uma SOA a um estado de *deadlock*, não atendendo, dessa forma, o requisito não funcional de confiabilidade. Caso seja identificado processos não *sound*, um procedimento deverá permitir uma alteração localizada na arquitetura para atender este requisito não funcional, mantendo os requisitos funcionais e não alterando de forma significativa o desempenho da arquitetura.
7. Desenvolver estudos de casos ilustrativos, aplicando as abordagens definidas e verificadas nos itens 1, 2, 3, 4, 5 e 6.

1.3 Hipóteses

Diante dos objetivos apresentados na seção 1.2, a principal questão de pesquisa deste trabalho é:

Q1 - É possível verificar formalmente que modelos de SOA atendem requisitos especificados em modelos de análise?

Para essa questão, enuncia-se a seguinte hipótese:

H1 - Representando modelos de requisitos de sistemas de gerenciamento de processos de negócios e modelos de SOA por meio de redes de Petri (*WorkFlow nets*) e da Lógica Linear, é possível, baseando-se no cálculo dos sequentes da Lógica Linear, definir um algoritmo de verificação de requisitos funcionais e não funcionais em uma SOA.

Considerando a principal questão de pesquisa Q1, o primeiro subproblema a ser considerado nesse trabalho trata-se da representação das Arquiteturas Orientadas a Serviços por meio das *WorkFlow nets*. Portanto, pare esse subproblema, enuncia-se a seguinte questão de pesquisa:

Q2 - Como representar uma SOA utilizando *WorkFlow nets*?
Para essa questão, enuncia-se a seguinte hipótese:

H2 - Cada serviço de uma SOA pode ser representado por uma *WorkFlow net* única, sendo que a composição da totalidade dos serviços fornecidos pela arquitetura proposta (ou a composição das *WorkFlow nets* resultantes) pode ser representada por uma *WorkFlow net* interorganizacional.

Como a ideia principal do trabalho é garantir que um projeto de SOA atenda adequadamente às especificações de requisitos, além do modelo de SOA, é necessário representar o modelo de especificação de requisitos. Portanto, a seguinte questão de pesquisa é considerada:

Q3 - Como representar o modelo de especificação de requisitos que servirá como base para identificar os requisitos que deverão ser verificados no modelo de SOA?
Para essa questão, enuncia-se a seguinte hipótese:

H3 - Um modelo de requisitos de um sistema de gerenciamento de processos de negócios corresponde a um contrato de serviço. Um contrato de serviço descreve as tarefas que são de interesse público, ou seja, especifica os requisitos de sistema esperados que as partes envolvidas no processo terão de executar. Um modelo de contrato de serviço também pode ser representado por uma *WorkFlow net*.

Os requisitos funcionais serão verificados em relação ao comportamento dos modelos. Por isso, é preciso definir um algoritmo de verificação da equivalência entre os modelos de requisitos e de SOA em relação aos requisitos funcionais no que se refere ao comportamento dos modelos. Desse modo, para este problema, tem-se a seguinte questão de pesquisa:

Q4 - Considerando modelos de requisitos e de SOA, representados por *WorkFlow nets*, como definir um algoritmo para verificar a equivalência entre esses modelos em relação aos requisitos funcionais no que diz respeito ao comportamento dos modelos?
Para essa questão, enuncia-se a seguinte hipótese:

H4 - Como existe uma tradução quase direta entre a estrutura de uma rede de Petri e um sequente da Lógica Linear, o cálculo dos sequentes da Lógica Linear pode ser utilizado como base para a definição de um algoritmo que permite comparar os modelos de processos (requisitos e arquitetura) e verificar suas equivalências em relação aos requisitos funcionais no que diz respeito ao comportamento dos modelos. Como a Lógica Linear permite analisar separadamente subprocessos, é possível considerar cenários dos modelos de requisitos e verificar se estes cenários também estão presentes na arquitetura. A noção de equivalência do tipo bissimulação *branching* corresponde a uma equivalência que satisfaz o tipo de requisitos que devem ser verificados em uma SOA. Como o foco deste trabalho é de verificar processos que

possuem o mesmo comportamento observável (externo), mas eventualmente com diferentes comportamentos silenciosos (internos), a bissimulação *branching* atende essa característica.

Um dos requisitos não funcionais que será verificado será em relação ao desempenho dos modelos. Por isso, é preciso definir um algoritmo de verificação da equivalência entre os modelos de requisitos e de SOA em relação aos requisitos não funcionais no que se refere ao desempenho dos modelos. Desse modo, para este problema, enuncia-se a seguinte questão de pesquisa:

Q5 - Considerando modelos de requisitos e de SOA, representados por *WorkFlow nets*, como definir um algoritmo para verificar a equivalência entre esses modelos em relação aos requisitos não funcionais no que diz respeito ao desempenho dos modelos?
Para essa questão, enuncia-se a seguinte hipótese:

H5 - O cálculo dos sequentes da Lógica Linear juntamente com o cálculo de datas simbólicas baseado em operadores (*max*, *plus*), que considera dados extraídos de árvores de provas da Lógica Linear através da produção de intervalos de datas, pode ser utilizado como base para a definição do algoritmo que permite comparar os modelos de processos (requisitos e arquitetura) e verificar suas equivalências em relação aos requisitos não funcionais no que diz respeito ao desempenho dos modelos.

Outro requisito não funcional que será verificado está relacionado à confiabilidade. Por isso, é preciso definir um algoritmo para detectar possíveis situações que podem levar uma SOA a um estado de *deadlock*. Caso o *deadlock* ocorra, o algoritmo deve permitir a alteração da arquitetura para transformar os processos não *sound* em processos seguros e confiáveis, mantendo os requisitos funcionais e não alterando de forma significativa o desempenho.

Desse modo, para este problema, tem-se a seguinte questão de pesquisa:

Q6 - Considerando o modelo de SOA, representado por *WorkFlow nets*, como definir um algoritmo para detectar possíveis situações que podem levar uma SOA a um estado de *deadlock*?
Para essa questão, enuncia-se a seguinte hipótese:

H6 - A partir de uma marcação indesejada, que representa um estado parcial do modelo é possível identificar todas as sequências de ações (cenários) que levam uma SOA a um estado de *deadlock*. Para isso, um raciocínio inverso pode ser aplicado na arquitetura e o cálculo dos sequentes da Lógica Linear pode ser utilizado para verificar os cenários encontrados.

Relacionada com a questão Q6, ainda há outra questão de pesquisa:

Q7 - Caso uma SOA não seja livre de *deadlock*, como alterar a arquitetura para transformar os processos não *sound* em processos seguros e confiáveis, mantendo os requisitos funcionais e não alterando de forma significativa o desempenho?
Para essa questão, tem-se a seguinte hipótese:

H7 - Considerando que as *WorkFlow nets* que representam os processos individuais de cada organização são livres de *deadlock*, é possível trocar alguns lugares de comunicação assíncrona das *WorkFlow nets* compostas (*WorkFlow net* interorganizacional) por um tipo de mecanismo de comunicação síncrona forçando as *WorkFlow nets* individuais a iniciar tarefas específicas ao mesmo tempo (OLIVEIRA et al., 2017b). Desse modo, não será criada uma situação de espera circular por parte dos mecanismos de comunicação assíncronas que são os principais responsáveis por possíveis situações de *deadlock*. Após a alteração da arquitetura para corrigir possíveis *deadlock*, a verificação da equivalência entre os modelos de requisitos e de arquitetura deve ser realizada novamente. Para isso, será possível considerar somente os cenários modificados para a remoção do *deadlock* sem necessidade de refazer toda a análise para a verificação da equivalência entre os modelos.

1.4 Contribuições

Este trabalho de pesquisa contribuirá para a área de Engenharia de Software garantindo que uma proposta de SOA reproduza o comportamento do modelo de especificação de requisitos, permitindo assim que os requisitos exigidos pelo cliente tenham maior probabilidade de serem alcançados. Com a implementação da abordagem, também é maior a probabilidade de minimizar os riscos de falhas nos projetos que lidam com SOA e ainda aumentar a garantia da qualidade do projeto e evitar custos com retrabalho. Com essa verificação inicial será possível evitar que erros sejam propagados para as fases de implementação e testes sendo, portanto, essa uma solução mais econômica. As abordagens apresentadas nesse trabalho de pesquisa contribuirão não apenas para a verificação de requisitos funcionais, mas também para a verificação de requisitos não funcionais como desempenho e segurança. A verificação realizada será em relação a corretude dos requisitos definidos no modelo de análise. A corretude verifica até que ponto um sistema satisfaz as suas especificações e cumpre com os objetivos do usuário (VLIET, 2007).

Especificamente para a área de SOA, este trabalho de pesquisa contribuirá com a representação formal dos serviços que integram uma SOA. Portanto, a análise, a modelagem e a compreensão da arquitetura será realizada completamente de maneira formal. Métodos formais usados durante a fase de projeto permitem a detecção inicial de inconsistências arquiteturais e erros. Como os métodos formais são baseados em princípios matemáticos, os modelos serão precisos e expressivos. Com a utilização das *WorkFlow nets*, *WorkFlow nets* interorganizacionais e da Lógica Linear para representar os modelos

de processos será possível demonstrar seguramente a consistência dos modelos e ainda estabelecer se o sistema exibe ou não algumas propriedades precisamente definidas, como, por exemplo, livre de *deadlock*.

Outra importante contribuição será na área de processos de *workflow*, já que os modelos de requisitos e de SOA serão representados por processos de *workflow*. Diferentes de outras abordagens, os processos de *workflow* que representarão uma SOA serão construídos de forma independente e não será utilizado nenhum tipo de regra de construção, como acontece geralmente quando se usa especificação formal para a modelagem de sistemas complexos. Portanto, a contribuição se refere aos processos de *workflow* flexíveis que não sofrerão nenhum tipo de restrição na forma em que eles irão expressar os requisitos do cliente. Ainda na área de processos de *workflow*, outra contribuição será em relação à comparação de modelos de processos em que suas correspondências devem ser verificadas. As técnicas existentes na literatura para verificação de equivalência comportamental são principalmente baseadas na análise do espaço de estado, o que é computacionalmente caro. Como a abordagem apresentada nesse trabalho utilizará a Lógica Linear, juntamente com o conceito de bissimulação como base para definir o algoritmo de comparação entre os modelos de processos, não será necessário considerar, por exemplo, o grafo de alcançabilidade como acontece quando se trabalha exclusivamente com redes de Petri.

1.5 Revisão da Literatura Correlata

Algumas abordagens já foram propostas a fim de verificar requisitos em artefatos de um projeto de software. Por exemplo, os trabalhos de (HOYOS; CASALLAS; JIMÉNEZ, 2012), (ARIAS; HIRATA, 2011), (TSADIMAS; NIKOLAIDOU; ANAGNOSTOPOULOS, 2012), (TSADIMAS, 2015) e (POOLEY; ABDULLATIF, 2010) abordam a verificação de requisitos de tempo em diferentes contextos de software. Em (HOYOS; CASALLAS; JIMÉNEZ, 2012), é apresentada uma abordagem para verificações iniciais em projetos de sistemas embarcados. Os autores utilizam uma Linguagem de Descrição de Arquitetura, chamada de HiLes, para expressar o comportamento de sistemas embarcados através de uma rede de Petri. Para capturar os requisitos e para definir a solução lógica do sistema é utilizada a SysML (*System Modeling Language*). A solução lógica descreve os aspectos comportamentais e estruturais do sistema. O trabalho está mais preocupado com aspectos temporais; assim HiLeS é estendida para HiLeS-T baseado em redes de Petri temporais. Portanto, as informações comportamentais são extraídas a partir do modelo HiLes para um modelo de rede de Petri temporal que pode ser usado como uma entrada para a verificação do modelo. Para integrar capacidades de simulação em SysML, os autores de (TSADIMAS; NIKOLAIDOU; ANAGNOSTOPOULOS, 2012) e (TSADIMAS, 2015) propuseram o conceito de Visões de Avaliação, um diagrama discreto para especificar a arquitetura de sistemas de informação empresarial e as condições sob as quais os

requisitos de desempenho devem ser verificados. Um perfil da SysML correspondente, chamado de perfil EIS (*Enterprise Information System*), foi definido. A abordagem fornece a incorporação de resultados de simulação nos modelos EIS da SysML originais para permitir a verificação dos requisitos de desempenho correspondentes. Para verificar apenas o desempenho em arquiteturas de software, em (POOLEY; ABDULLATIF, 2010), foi introduzido o método CPASA usado juntamente com a ferramenta UML-JMT desenvolvida para implantar os testes de avaliação de desempenho exigidos neste método. Em (ARIAS; HIRATA, 2011), é descrito o mapeamento de um modelo de software para um modelo de simulação a fim de suportar a verificação de requisito de desempenho. O mapeamento é baseado em UML e Diagramas de Máquina de Estado, comentados com informações de desempenho, para um modelo de simulação que é especificado em Diagramas de Ciclo de Atividades. O modelo de simulação é traduzido para um programa de simulação para que a verificação dos requisitos de desempenho seja realizada. O mapeamento é parte de um *framework* baseado em perfis da UML para MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*) empregados para a verificação dos requisitos de desempenho para sistemas de computadores de tempo real. Como pode ser observado, estes trabalhos focam em diferentes contextos de software e abordam especificamente a verificação de requisitos de desempenho utilizando algum tipo de simulação. Além disso, em sua maioria são utilizadas abordagens semiformais não sendo possível realizar verificações formais nos modelos. Modelos semiformais podem permitir rápida modelagem, no entanto, podem encapsular ambiguidades pela falta de formalidade.

Para verificar requisitos que não são somente diretamente relacionados ao desempenho, os trabalhos de (AMATO; MOSCATO, 2014), (BUFFONI-ROGOVCHENKO et al., 2013) e (SENGUPTA; DASGUPTA, 2015) podem ser citados. No trabalho de (AMATO; MOSCATO, 2014), é descrito um perfil de modelagem usado em um *framework* chamado de MetaMORP(h)OSY (*Meta-modeling of Mas Object-based with Real-time specification in Project Of complex SYstems*), compatível com MARTE, para a descrição de serviços em nuvem e para a especificação de requisitos disponíveis (em termos de qualidade de software). A metodologia usa um perfil de modelagem capaz de descrever serviços como agentes em um ambiente multiagente e é baseado em Engenharia Dirigidas por Modelos. Quando um requisito tem de ser verificado, uma ou mais propriedades do RT-AML (linguagem baseada em UML para descrever diagramas referentes a abordagem proposta) devem ser analisadas. No trabalho de (BUFFONI-ROGOVCHENKO et al., 2013), os autores propuseram uma extensão para a linguagem Modelica, uma linguagem baseada em equação e orientada a objetos para modelagem de sistemas. O trabalho mostra como a verificação de requisitos pode ser usada juntamente com o processo de simulação para rastrear os componentes responsáveis pelas violações dos requisitos. Os autores criaram um modelo de monitoramento cujo objetivo é conectar um modelo físico e um conjunto de requisitos a serem verificados. A verificação é feita manualmente escrevendo o código

que conecta as entradas dos requisitos para o modelo físico. Em (SENGUPTA; DASGUPTA, 2015), foi proposto um modelo de verificação para descobrir se cada requisito em um determinado nível de abstração tem um objetivo correspondente em seu nível mais alto de abstração. Desse modo, os requisitos são especificados em níveis hierárquicos de abstração. Novamente, essas abordagens estão inseridas em contextos de software específicos sendo que a verificação é realizada por um tipo de conexão entre os artefatos. A maioria das abordagens utilizam métodos semiformais, não apresentando formalmente como requisitos podem ser verificados em outros artefatos de software.

Os trabalhos de (SEEDORF; NORDHEIMER; KRUG, 2009), (MANDAL; SARKAR, 2016), (RENAUX; VANWORMHOUDT; TOMBELLE, 2013), (SALOMIE et al., 2008), (ZHANG; MAO; ZHOU, 2009), (BENDRISS; BENABDELHAFID, 2011) e (PARDAL et al., 2012) abordam algum tipo de rastreabilidade em SOA. No trabalho de (SALOMIE et al., 2008), por exemplo, foi proposto um modelo de cadeia logística com características de rastreabilidade. Como uma solução para instanciar o modelo, foi apresentado um agente baseado em SOA que fornece os recursos necessários para a construção, execução e monitoramento de cadeias logísticas e realização de operações de rastreabilidade. O item rastreável percorre os segmentos da cadeia. Já no trabalho de (SEEDORF; NORDHEIMER; KRUG, 2009), foi proposto um modelo de ciclo de vida para descrever uma SOA e um *framework* de rastreabilidade chamado de STraS. Os *links* de rastreabilidade são incorporados como metadados adicionais em um artefato e extraídos por um *plug-in*, sendo capturados automaticamente. Em (RENAUX; VANWORMHOUDT; TOMBELLE, 2013), foi apresentado um *framework* de múltiplas visões para apoiar o projeto de serviços ao longo do processo de desenvolvimento, ou seja, a partir dos requisitos para a implementação. O *framework* é baseado na noção de Blocos de Serviços Lógicos que representa um serviço a partir de Casos de Uso e atua como um pivô entre as visões garantindo a consistência e a rastreabilidade. Em (MANDAL; SARKAR, 2016), foi proposta uma especificação de projeto de Sistema Orientado a Serviços, usando uma abordagem baseada em modelo específico de domínio. Na proposta foi definido um conjunto de conceitos e construções semânticas tanto para o domínio do processo empresarial como para o domínio do serviço. O modelo compreende um conjunto de conceitos semânticos e construções relacionadas para conceituar as facetas dos domínios de serviços e processos de negócio. A abordagem também propõe um mecanismo de rastreabilidade sistemática entre os conceitos de modelagem desses domínios. A rastreabilidade dessas abordagens apresentadas na literatura, consiste na definição e representação de *links* entre um artefato e outro, não sendo, portanto, realizada algum tipo de verificação, principalmente verificações formais uma vez que os modelos são informais ou semiformais como, por exemplo, UML, BPMN, XML (*eXtensible Markup Language*) e DAML-S (DARPA *Agent Markup Language for Services*).

No contexto de arquiteturas distribuídas, o trabalho de (HIERONS; NEZ, 2017) uti-

liza testes probabilísticos de software para validar a corretude do sistema. Para isso, os autores definiram diferentes relações de implementação que indicam se um sistema é uma implementação válida de uma especificação. As relações de implementação são definidas em termos de rastros. O comportamento do sistema baseado em estado é denotado por uma sequência de ações, sendo que o modelo usado é um sistema de transição rotulado probabilístico. No trabalho de (HIERONS; NEZ, 2017), são utilizados métodos formais para validar uma especificação, no entanto, diferentemente da abordagem apresentada nesse trabalho de pesquisa, a validação é realizada no nível de testes.

Em relação à verificação de similaridade de processos, várias publicações mostraram como as técnicas de correspondência podem ser aplicadas para modelos de processos de negócios (DIJKMAN et al., 2009), (EHRIG; KOSCHMIDER; OBERWEIS, 2007), (DONGEN; DIJKMAN; MENDLING, 2008), (WEIDLICH; DIJKMAN; MENDLING, 2010), (NEJATI et al., 2007). Por exemplo, em (NEJATI et al., 2007) é apresentada uma abordagem para encontrar correspondências no contexto de modelos *Statecharts* [5] e para combinar esses modelos em relação as correspondências conhecidas entre eles. Nesta abordagem, são utilizadas heurísticas para encontrar similaridades terminológicas, estruturais e semânticas entre modelos *Statecharts*. Em (DIJKMAN et al., 2009), os modelos de processos são vistos como um gráfico atribuído dirigido e são utilizadas técnicas de correspondência gráfica para identificar partes correspondentes de modelos de processos relacionados. Para verificar a relação dos modelos, os autores utilizaram correspondência léxica baseada na comparação dos rótulos que aparecem nos modelos de processos usando a similaridade sintática ou semântica ou a combinação de ambas, e correspondência estrutural baseada na topologia dos modelos de processos vistos como grafos. Em (DIJKMAN et al., 2011), são apresentadas métricas para a verificação de similaridade entre processos de negócio. A verificação de similaridade é aplicada em repositórios de modelos de processos de negócio que são construídos por organizações e podem conter centenas ou até mesmo milhares de modelos de processos de negócio; desse modo, antes de adicionar um novo modelo de processo em um repositório, é preciso verificar se há um modelo similar. Nesta abordagem, os modelos de processos são representados por *Event driven Process Chains* (EPCs). As métricas apresentadas no trabalho são estruturais, utilizando técnicas existentes para comparação de grafo baseada na distância de edição gráfica e comportamentais, considerando a semântica comportamental de modelos de processos. Como pode ser observado nas referências em relação a verificação de similaridade de processos, são considerados diferentes tipos de representação de processos e assim cada trabalho apresenta uma abordagem específica de comparação. Esses trabalhos diferem da abordagem proposta nesta tese em relação ao contexto de aplicação e aos tipos de linguagens de representação de processos consideradas.

[5] *Statecharts* é uma linguagem de projeto e implementação bastante usada para especificar comportamentos dinâmicos de sistemas de software (HAREL, 1987).

Na literatura muitos trabalhos têm considerado o conceito de perfil comportamental (WEIDLICH; MENDLING; WESKE, 2011a) para capturar as restrições comportamentais essenciais de um modelo de processo. Um perfil comportamental de um modelo de processo pode ser visto como um grafo completo sobre o conjunto de tarefas onde as arestas são rotuladas por tipos de relações comportamentais. Alternativamente, um perfil comportamental pode ser visto como uma matriz quadrada onde linhas e colunas representam rótulos de tarefas e cada célula é rotulada por uma relação comportamental entre um par de tarefas. No trabalho de (WEIDLICH; MENDLING; WESKE, 2011a), os autores buscaram verificar o alinhamento de modelos de processos justificando que os requisitos de negócio, representados pelos modelos de processo de negócios, diferem significativamente dos modelos de projeto de software. A verificação do alinhamento de modelos de processos requer a identificação de correspondências de modelos e, para isso, no trabalho de (WEIDLICH; MENDLING; WESKE, 2011a), são utilizadas verificações de perfis comportamentais. Perfis comportamentais têm sido aplicados no contexto de comparação comportamental, na busca de similaridade e na verificação de conformidade, como pode ser observado nos seguintes trabalhos (WEIDLICH; WESKE; MENDLING, 2009), (WEIDLICH et al., 2010), (KUNZE; WEIDLICH; WESKE, 2011), (WEIDLICH; MENDLING; WESKE, 2011a), (WEIDLICH; MENDLING; WESKE, 2011b), (WEIDLICH et al., 2010). Por exemplo, em (WEIDLICH; WESKE; MENDLING, 2009), perfis comportamentais são utilizados para gerenciar propagações de mudanças entre modelos de processos de negócio. Esses modelos residem em diferentes níveis de abstração e assumem diferentes perspectivas de modelagem. Dada uma mudança no modelo fonte, a abordagem isola uma potencial região de mudança no modelo de destino baseado no perfil comportamental das atividades correspondentes. Em (KUNZE; WEIDLICH; WESKE, 2011), foi proposto uma métrica comportamental, baseada em perfis comportamentais que quantifica a similaridade de modelos de processos. Usando as abstrações do comportamento de um modelo de processo foram propostas cinco medidas de similaridade e, baseado nessas medidas, foi construída uma métrica que quantifica a similaridade comportamental de modelos de processos.

Conforme apresentado por (ARMAS-CERVANTES et al., 2016) e por (POLYVYANYY et al., 2016), perfis comportamentais são considerados como uma forma conveniente para comparação de pares de modelos de processos com respeito ao seu comportamento ou computação de similaridade comportamental. Esses trabalhos mostraram que perfis comportamentais são eficientes para mostrar a equivalência de configuração entre sistemas capturados como redes não rotuladas acíclicas. No entanto, para a classe geral de redes acíclicas, os perfis comportamentais são exponencialmente imprecisos, o que significa que duas redes acíclicas com o mesmo perfil comportamental podem diferir em um número exponencial de configurações. Segundo (POLYVYANYY et al., 2016), o poder expressivo de perfis comportamentais é comprometido em algumas situações. Por exemplo, um

perfil nem sempre captura o fato de uma atividade 'b' sempre preceder uma atividade 'e' para algumas configurações em um sistema. Perfis comportamentais não podem ser usados para decidir equivalência de rastros de autômatos finitos e, portanto, de redes de Petri. Adicionalmente, segundo (DIJKMAN et al., 2011), técnicas existentes para a verificação de equivalência comportamental são principalmente baseadas na análise do espaço de estado, o qual é computacionalmente cara. Mesmo as classes restritas de redes de Petri 1-*safe* requerem espaço exponencial para a maioria das noções de equivalência. Diferentemente dos trabalhos citados, a abordagem apresentada nesse trabalho de pesquisa, utilizará a prova dos sequentes da Lógica Linear para verificar o comportamento dos processos evitando os problemas apresentados em relação aos perfis comportamentais.

Em relação à utilização de *workflow* no contexto distribuído, ou seja, *workflow* interorganizacional, alguns trabalhos podem ser citados. Em (AALST; WESKE, 2001) e (AALST, 2003), por exemplo, foi proposta uma abordagem para especificar *workflow* interorganizacional e também evitar que estes *workflow* possuam anomalias típicas como *deadlocks* e *livelocks*. A abordagem, chamada de P2P (*Public-To-Private*) é baseada na herança de projeção. Herança de projeção usa o encapsulamento como um mecanismo para estabelecer relacionamentos de superclasse e subclasse. O problema dessa abordagem é que são utilizadas regras definidas pelos autores para a construção do *workflow* interorganizacional. Esse fato, em particular, limita a flexibilidade de construção que processos interorganizacionais podem precisar. Já em (PASSOS; JULIA, 2015), é apresentado um método que identifica cenários livres de *deadlocks* no contexto da composição de serviços *Web* modelados por redes de Petri. Este método é baseado na análise das árvores de prova da Lógica Linear baseado nos cenários de serviços (módulos) que compõem o sistema composto. Essa abordagem somente detecta cenários seguros, isto é, os cenários livres de *deadlock*, os quais garante que situações sem *deadlock* poderão ser alcançadas durante a execução do sistema composto. Em (SAIDA; MAROUA; ZAIA, 2016), foi descrito uma abordagem e uma ferramenta para verificar a propriedade *Soundness* em modelos de processos interorganizacionais. Os modelos de processos são especificados com a linguagem BPEL (*Business Process Execution Language*) e são transformados para modelos de redes de Petri. Como a análise é realizada a partir dos modelos em redes de Petri é necessário construir o grafo de alcançabilidade, o que pode levar a uma explosão do espaço de estados. Baseado em visões de processos, os trabalhos de (EDER et al., 2011) e de (LIN; ISHIDA, 2008) apresentaram modelos para representar visões de processos interorganizacionais. No entanto, essas abordagens são baseadas em linguagens informais e semiformais como o Gráfico de Sequência de Mensagens, não sendo possível realizar a verificação formal dos modelos.

1.6 Método de Pesquisa

De acordo com (WAZLAWICK, 2009), o método de pesquisa consiste na sequência de passos necessários para demonstrar que o objetivo proposto foi atingido, ou seja, se os passos definidos no método forem executados os resultados obtidos deverão então ser convincentes. Lakatos e Marconi (MARCONI; LAKATOS, 2011) afirmam que a utilização de métodos científicos não é exclusiva da ciência, sendo possível usá-los para a resolução de problemas do cotidiano. No entanto, destacam que não há ciência sem o emprego de métodos científicos.

Segundo (PRODANOV; FREITAS, 2013), muitos foram os pensadores e filósofos do passado que tentaram definir um único método aplicável a todas as ciências e a todos os ramos do conhecimento. Essas tentativas culminaram no surgimento de diferentes correntes de pensamento. Assim, muitas vezes é necessário a combinação de métodos científicos diferentes, dependendo do objeto de investigação e do tipo de pesquisa.

O conhecimento científico difere dos outros tipos de conhecimento por ter fundamentações e metodologias a serem seguidas, além de se basear em informações classificadas, submetidas à verificação, que oferecem explicações plausíveis a respeito do objeto ou evento em questão (PRODANOV; FREITAS, 2013). Uma das classificações da ciência se dá em função dos objetos de estudo. Nesse nível, segundo (MARCONI; LAKATOS, 2011), a ciência é dividida em ciências formais e em ciências factuais, sendo que a primeira preocupa-se com o estudo das ideias e a segunda com o estudo dos fatos. A Lógica e a Matemática, por exemplo, são objetos de estudo das ciências formais. Como as ciências formais dedicam-se as ideias, contentam-se com a lógica para demonstrar rigorosamente seus teoremas cuja verdade depende unicamente do significado de seus termos ou sua estrutura lógica (MARCONI; LAKATOS, 2011). Desse modo, de acordo com as definições apresentadas, este trabalho de pesquisa insere-se no ramo das ciências formais.

Ao iniciar um trabalho de pesquisa, uma das primeiras atividades que deve ser realizada é a revisão da literatura. Uma revisão sistemática na literatura é um meio de avaliar e interpretar todas as pesquisas relevantes disponíveis para uma questão de pesquisa particular sintetizando os trabalhos existentes (KITCHENHAM; CHARTERS, 2007). A Figura 1 resume o processo de revisão da literatura seguido neste trabalho. Além do processo descrito na Figura 1, foi realizada também uma busca em profundidade, ou seja, as referências dos artigos selecionados foram analisadas na tentativa de encontrar trabalhos relevantes.

Após a revisão da literatura, foram definidos algoritmos baseados em teorias formais, como redes de Petri e Lógica Linear, para formalizar as abordagens propostas considerando os objetivos descritos na subseção 1.2.

Neste trabalho, os principais instrumentos de pesquisa serão os estudos de caso propostos para ilustração e para estudo comparativo com outras abordagens conhecidas, com o intuito de mostrar as vantagens e as limitações das abordagens propostas neste trabalho

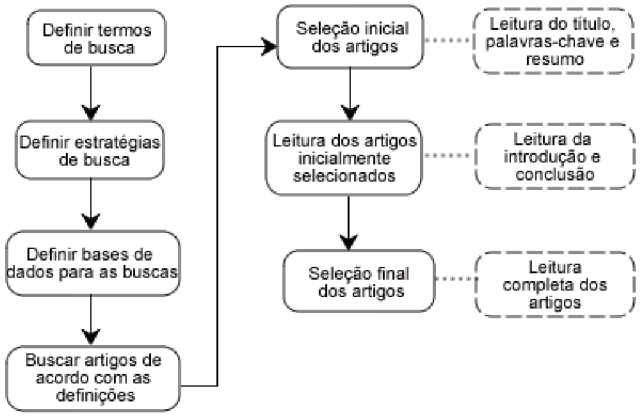


Figura 1 – Processo de Revisão da Literatura

de pesquisa. Estudo de caso é uma pesquisa empírica que investiga em profundidade um fenômeno contemporâneo dentro de seu contexto da vida real (YIN, 2009). Segundo (RUNESON; HöST, 2009), estudo de caso é um método de pesquisa adequado para muitos tipos de pesquisas em Engenharia de Software explorando e investigando fenômenos da vida real através da análise contextual detalhada de um número limitado de eventos ou condições e suas relações.

Para alcançar o principal objetivo desse trabalho de pesquisa (verificar que uma SOA reproduz os requisitos definidos num modelo de especificação de requisitos) é necessário subdividi-lo conforme os objetivos específicos 1, 2, 3, 4, 5 e 6 apresentados na subseção 1.2. Desse modo, as seguintes atividades serão consideradas:

Objetivo 1 - Representar uma SOA usando *WorkFlow nets* interorganizacionais.
Para alcançar esse objetivo, é necessário verificar os elementos de uma SOA que devem ser representados e, em seguida, definir como serão representados por *WorkFlow nets*. Como uma SOA trabalha com serviços distribuídos, é necessário considerar mecanismos de composição; por isso, deve-se também analisar e definir como as *WorkFlow nets* interorganizacionais poderão ser utilizadas para representar a composição de serviços. Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Objetivo 2 - Representar os modelos de especificação de requisitos de um projeto de SOA por meio de *WorkFlow nets*.
Baseado na hipótese **H3**, apresentada na subseção 1.3, é preciso analisar e definir como um modelo de requisitos, considerado como sendo um contrato de serviços, será representado por meio de *WorkFlow nets*. É preciso também verificar se o contrato de serviços será considerado na sua forma composta (composição de contrato de serviços) ou será considerado de forma independente (contrato de um único

serviço). Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Objetivo 3 - Definir como se dará a equivalência de semântica operacional entre modelos de processos baseados em *WorkFlow net* e Lógica Linear considerando a noção de equivalência do tipo bissimulação *branching*.
Primeiramente é necessário pesquisar sobre a aplicação de bissimulação, especificamente bissimulação *branching*, na verificação de equivalência de processos. É necessário também pesquisar por outras formas de equivalência de processos e comparar com bissimulação *branching*, analisando as vantagens e desvantagens das propostas existentes. Em seguida, deve-se definir, a partir das *WorkFlow nets* e da Lógica Linear, em quais condições os processos serão considerados equivalentes. Na sequência, teoremas e/ou proposições deverão ser demonstrados para que a abordagem possa ser formalmente verificada. Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Objetivo 4 - Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se os requisitos funcionais (em relação ao comportamento dos modelos) definidos num modelo de especificação de requisitos estão presentes na arquitetura correspondente. Tal algoritmo deverá permitir analisar cenários tanto em modelos *sound* quanto não *sound*.

Primeiramente, as *WorkFlows nets* que representam os modelos de requisitos e de arquitetura deverão ser convertidas para sequentes da Lógica Linear. Por isso, uma abordagem de conversão deverá ser definida. Em seguida, utilizando a noção de equivalência definida no objetivo 3, poderá ser realizada a comparação dos modelos em relação aos requisitos funcionais. Procedimentos mais específicos para a realização da comparação dos modelos deverão ser descritos. Depois de definido o algoritmo, este deverá ser validado através da prova de teoremas e/ou proposições. Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Objetivo 5 - Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se os requisitos não funcionais, em relação ao desempenho de SOA, definidos num modelo temporal de especificação de requisitos estão presentes na arquitetura correspondente. O algoritmo deverá permitir analisar cenários tanto em modelos *sound* quanto não *sound*.

Utilizando os sequentes da Lógica Linear obtidos por meio da abordagem do objetivo 4, poderão ser produzidas datas simbólicas de realização de atividades dos processos.

Em seguida, será preciso definir como a comparação dos modelos temporais, considerando as datas produzidas, deve ser realizada. Desse modo, serão estabelecidos passos mais específicos para a realização da comparação dos modelos para verificar a equivalência em relação aos requisitos não funcionais de desempenho. Depois de definido o procedimento de verificação, este deverá ser validado através da prova de teoremas e/ou proposições. Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Objetivo 6 - Definir um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para detectar possíveis situações que podem levar uma SOA a um estado de *deadlock*, não atendendo, dessa forma, o requisito não funcional de confiabiliade. Caso seja identificado processos não *sound*, um procedimento deverá permitir uma alteração localizada da arquitetura para atender este requisito não funcional, mantendo os requisitos funcionais e não alterando de forma significativa o desempenho da arquitetura.

Utilizando os sequentes da Lógica Linear obtidos por meio da abordagem do objetivo 4, deverá ser verificado quais os cenários que poderão levar o modelo a um estado de *deadlock*. Em seguida, é necessário definir uma abordagem para eliminar tais estados. Procedimentos mais específicos para a realização da comparação dos modelos deverão ser estabelecidos. Depois de definido tal procedimento, este deverá ser validado através da prova de teoremas e/ou proposições. Um estudo de caso ilustrativo deverá ser apresentado para mostrar a aplicação da abordagem proposta.

Os procedimentos propostos serão instrumentalizados e verificados através do uso da ferramenta CPN Tools (RATZER et al., 2003) que permite a modelagem, verificação e simulação de redes de Petri Coloridas hierárquicas, permitindo em particular a especificação de arquiteturas distribuídas por meio de vários modelos se comunicando.

1.7 Resultados Esperados

Com a realização deste trabalho, espera-se obter como resultado o desenvolvimento de um procedimento formal que consiga verificar a equivalência entre modelos de especificação de requisitos com modelos de especificação arquitetural no contexto de SOA. Para isso, espera-se obter também:

- Elaboração de modelos de SOA baseados em *WorkFlow nets* interorganizacionais.
- Elaboração de modelos de especificação de requisitos de um projeto de SOA por meio de *WorkFlow nets*.

- ❏ Definição de uma semântica operacional baseada no cálculo dos sequentes da Lógica Linear em que as provas de sequentes simularão formalmente os cenários dos modelos de especificação dos processos de negócios modelados por *WorkFlow nets*.
- ❏ Definição de um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se requisitos funcionais (em relação ao comportamento dos modelos) definidos num modelo de análise estão presentes no correspondente modelo de SOA.
- ❏ Definição de um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para verificar se requisitos não funcionais (em relação ao desempenho) definidos num modelo de análise estão presentes no correspondente modelo de SOA.
- ❏ Definição de um algoritmo, baseado no cálculo dos sequentes da Lógica Linear, para detectar se uma SOA atende o requisito não funcional em relação à confiabilidade, e a introdução de regras de alteração da arquitetura para atender tal requisito sempre que necessário. Desse modo, o que requisito não funcional em relação a manutenção também será verificado.
- ❏ Validação das abordagens propostas por meio da prova de teoremas e/ou proposições e também por meio de simulações no simulador CPN Tools.
- ❏ Estudo de caso aplicando os modelos e procedimentos desenvolvidos.

1.8 Organização da Tese

O restante da tese está estruturada como apresentado em seguida.

No Capítulo 2 é apresentada a fundamentação teórica referente a Redes de Petri (seção 2.1), *WorkFlow net* (seção 2.2), *WorkFlow net* Interorganizacional (seção 2.3), *t-Time WorkFlow net* (seção 2.4), Lógica Linear (seção 2.5), Propriedade *Soundness* (seção 2.6) e Bissimulação *Branching* (seção 2.7).

No Capítulo 3 são apresentados os métodos para a verificação do comportamento e do desempenho de cenários de requisitos em SOA. Na seção 3.1 são apresentadas as definições dos modelos de requisitos e de arquitetura. O método para a verificação do comportamento de cenários de requisitos em SOA é formalizado na seção 3.2 e o método para a verificação do desempenho é formalizado na seção 3.3. A seção 3.4, que finaliza o capítulo 3, apresenta uma análise da complexidade dos métodos propostos.

No Capítulo 4 são apresentados os métodos para detecção e remoção de requisitos negativos do tipo *deadlock* em SOA. Na seção 4.1 é apresentado o método para detecção de requisitos negativos, enquanto na seção 4.2 é apresentado o método para o controle de *deadlock*.

No Capítulo 5 é apresentado um estudo de caso que considera a aplicação dos métodos propostos em uma composição de serviços *Web*. Na seção 5.1 é apresentada a relação

entre a modelagem de um processo de *workflow* interorganizacional e a modelagem de uma composição de serviços *Web*. Na seção 5.2 é apresentada a aplicação do método de verificação de requisitos funcionais; na seção 5.3 é apresentada a aplicação do método para verificação de requisitos não funcionais de desempenho; na seção 5.4 é apresentada a aplicação do método para detecção e remoção de requisitos negativos do tipo *deadlock*; já na seção 5.5 é apresentado um modelo de simulação baseado na ferramenta CPN Tools aplicado ao estudo de caso.

No Capítulo 6 a conclusão desta pesquisa é apresentada. Na seção 6.1 as principais contribuições são apresentadas, na seção 6.2 são descritos os trabalhos futuros e na seção 6.3 são apresentadas as contribuições em relação à produção bibliográfica.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos necessários para compreender os métodos formalizados no Capítulo 3. A seção 2.1 apresenta as terminologias e notações básicas relacionadas à teoria das redes de Petri que são essenciais para o entendimento das *WorkFlow nets* e das *WorkFlow nets* interorganizacionais, apresentadas, respectivamente, nas seções 2.2 e 2.3. A seção 2.4 apresenta a definição de uma *WorkFlow net* com restrições de tempo. A seção 2.5 apresenta os conceitos básicos da relação entre as redes de Petri e a Lógica Linear, e a seção 2.6 apresenta os conceitos referentes a propriedade *Soundness*. Por fim, os conceitos referentes à bissimulação *branching* são apresentados na seção 2.7.

2.1 Redes de Petri

Rede de Petri é uma ferramenta matemática e gráfica utilizada para a modelagem de vários tipos de sistemas que teve sua origem na tese de doutorado *Kommunikation mit Automaten* (Comunicação entre autômatos), defendida por Carl Adam Petri na Faculdade de Matemática e Física da Universidade de Darmstadt na Alemanha em 1962 (PETRI, 1962). Petri tinha o objetivo de desenvolver um modelo em que as máquinas de estado fossem capazes de se comunicarem; assim, ele apresentou um tipo de grafo bipartido direcionado. O grafo é composto por dois tipos de nós denominados lugares (representado por círculos) e transições (representada por retângulos ou barras) conectados via arcos direcionados.

Em (MURATA, 1989) é apresentada a seguinte definição para uma rede de Petri clássica:

Definição 2.1.1 *Uma rede de Petri clássica é uma tripla* $PN = \{P, T, F\}$*, onde:*

1. $P = \{p_1, p_2, ..., p_m\}$ *é um conjunto finito de lugares;*

2. $T = \{t_1, t_2, ..., t_n\}$ *é um conjunto finito de transições;*

3. $F \subseteq (P \times T) \cup (T \times P)$ *é um conjunto de arcos que representa uma relação de fluxo (arcos que ligam os lugares às transições e as transições aos lugares).*

Uma rede de Petri com uma dada marcação inicial M_0 é representada por (PN, M_0). A marcação de um rede de Petri é indicada pela presença de fichas nos lugares e graficamente é representada por pontos pretos dentro dos lugares. Formalmente, umas das notações utilizadas para representar uma mudança de marcação em uma rede de Petri é $M_0 \xrightarrow{\sigma} M_n$: a sequência de disparo $\sigma = t_1, t_2, ..., t_{n-1}$ leva da marcação M_0 para a marcação M_n, isto é, $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} ... \xrightarrow{t_{n_1}} M_n$.

A Figura 2 (a) mostra um exemplo de uma rede de Petri, onde o conjunto de lugares é (P_1, P_2, P_3), o conjunto de transições é (T_1, T_2) e a marcação inicial M_0 é P_1.

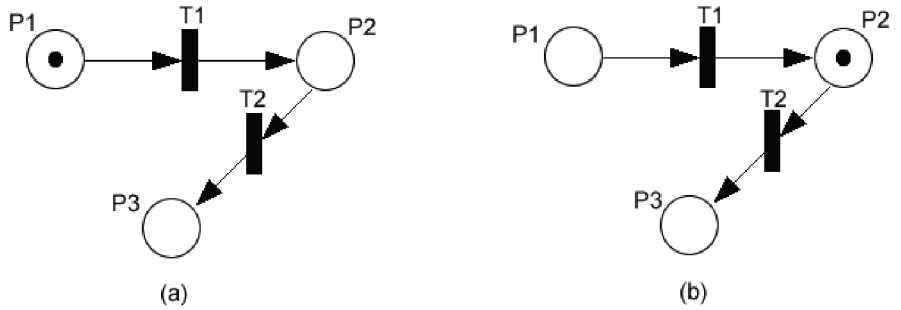


Figura 2 – Representação do Disparo de Transições.

Cada lugar representa em uma rede de Petri um estado parcial do sistema modelado; quando ocorre um evento o sistema passa do estado atual para o próximo estado. Cada evento está associado a uma transição; assim, a representação da ocorrência de um evento no sistema é dada pelo disparo da transição ao qual está associado. De acordo com (MURATA, 1989), as regras de disparo de uma transição são:

1. uma transição t é dita sensibilizada se, e somente se, cada lugar de entrada p de t contém pelo menos uma ficha. Por exemplo, a transição $T1$ da rede de Petri da Figura 2 (a) está sensibilizada, pois há uma ficha no lugar de entrada $P1$ de $T1$. Já a transição $T2$ da Figura 2 (a) não está sensibilizada, pois o lugar de entrada $P2$ de $T2$ não contém nenhuma ficha;

2. uma transição pode ou não ser disparada (dependendo se o evento realmente ocorre ou não);

3. se uma transição t disparar, então é removido uma ficha de cada lugar de entrada de t e produzido uma ficha em cada lugar de saída de t. Por exemplo, as redes de Petri das Figuras 2 (a) e 2 (b) mostram um exemplo de disparo de transição. Neste caso, $T1$ consome uma ficha do lugar de entrada $P1$ e produz uma ficha no lugar de saída $P2$.

Para obter algumas informações sobre o comportamento dinâmico de uma rede de Petri podem ser utilizados elementos estruturais juntamente com a informação sobre a marcação da rede, definindo, por exemplo, os componentes repetitivos e invariantes de transição. O invariante de transição (T-invariante) corresponde a uma sequência cíclica de eventos que pode ser repetida indefinidamente (MURATA, 1989). Observe, por exemplo, que a sequência $s = T_1T_2$ da rede de Petri apresentada na Figura 3 é um invariante de transição, pois o disparo de tal sequência não modifica a marcação da rede. O conjunto de transições T_1 e T_2 do invariante forma um componente repetitivo estacionário da rede. Os componentes repetitivos estacionários estabelecem uma condição necessária sobre o número de vezes que cada transição deverá ser disparada para que, eventualmente, um invariante de transição seja encontrado de acordo com a estrutura da rede de Petri.

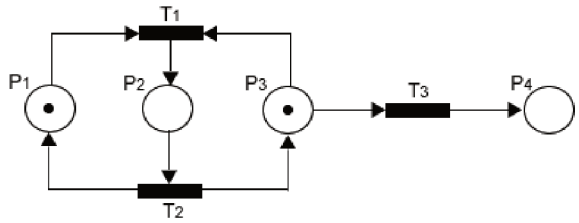


Figura 3 – Invariante de transição.

Para encontrar os T-invariantes de uma rede utiliza-se a equação fundamental da rede de Petri:

$$M' = M + C\mathbf{S} \text{ com } M \geq 0, \mathbf{S} \geq 0.$$

Na equação fundamental da rede de Petri, M' é a marcação obtida a partir da marcação inicial, M é a marcação inicial, C é a matriz de incidência (fornece o balanço das fichas na rede quando do disparo das transições) e $\mathbf{S}$ é um vetor característico da sequência s (cada componente do vetor representa o número de ocorrências de uma transição t numa sequência de disparo s). Para obter-se $M' = M$, a sequência s deve ser tal que o vetor $\mathbf{S}$ se verifique, ou seja, $C\mathbf{S} = 0$. Toda a solução do vetor $\mathbf{S}$ é chamada de componente repetitivo estacionário e a sequência s é dita invariante de transição.

Algumas ferramentas permitem o cálculo automático dos componentes repetitivos estacionários como, por exemplo, a ferramenta PIPE (DINGLE; KNOTTENBELT; SUTO, 2009) que será utilizada nesse trabalho de pesquisa. Aplicando a análise de invariantes da ferramenta PIPE na rede de Petri da Figura 3, por exemplo, o componente repetitivo estacionário apresentado na Figura 4 foi encontrado. O vetor mostrado na Figura 4 corresponde a uma lista não ordenada de transições que devem ser disparadas para que a rede volte a sua marcação inicial, sendo que o número 1 significa que cada transição foi disparada um única vez.

T1	T2	T3
1	1	0

Figura 4 – Componente repetitivo estacionário calculado pela ferramenta Pipe.

2.2 *WorkFlow net*

Grande parte dos trabalhos dentro das organizações não ocorrem em um contexto individual, mas em grupos de pessoas com tarefas divididas, o que torna o alcance de um objetivo mais rápido e eficiente. Desse modo, a coordenação, a comunicação e a cooperação são os principais fatores de uma atividade em grupo. Os sistemas de *workflow* surgiram justamente com o objetivo de facilitar e coordenar as atividades em grupo dentro de uma organização.

Segundo a WfMC (HOLLINGSWORTH, 1994), um *workflow* é a automação total ou parcial de um processo de negócio, durante a qual documentos, informações e tarefas são passados entre participantes do processo.

Um processo de *workflow* pode ser modelado em termos de uma rede de Petri denominada de *WorkFlow net* (WF-net). De acordo com (AALST, 1998a), a definição formal de uma *WorkFlow net* é apresentada como:

Definição 2.2.1 *Uma rede de Petri $PN = \{P, T, F\}$ é uma WorkFlow net se, e somente se:*

1. *PN possui dois lugares especiais: 'i' sendo o lugar inicial ($\bullet i = \varnothing$) e 'o' sendo o lugar final ($o\bullet = \varnothing$). A notação $\bullet i$ representa as transições de entrada do lugar 'i' (que neste caso é vazio) e $o\bullet$ representa as transições de saída do lugar 'o' (que neste caso também é vazio);*

2. *todo lugar da rede pertence em um caminho entre os lugares 'i' e 'o'.*

A Figura 5 mostra um exemplo de uma *WorkFlow net.*

De acordo com (AALST; HEE, 2004), para o roteamento de tarefas em uma WF-net, as seguintes construções básicas 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. Por exemplo, na WF-net da Figura 5, as transições $T1$ e $T2$ são executadas em sequência;

- ❏ paralela: mais de uma tarefa pode ser executada simultaneamente, ou em qualquer ordem. Por exemplo, na WF-net da Figura 5, as transições $T4$ e $T5$ podem ser executadas em paralelo;

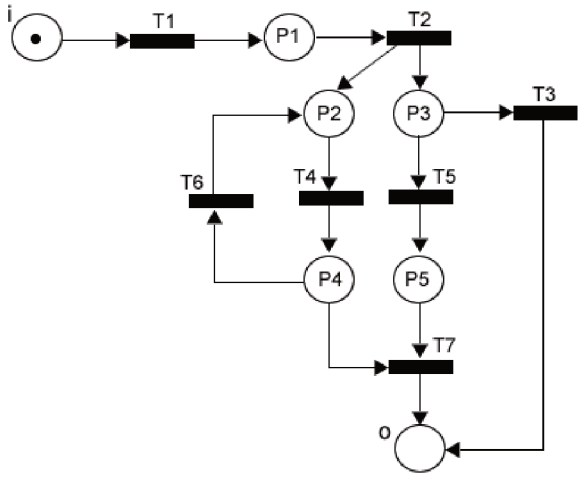


Figura 5 – Exemplo de uma WF-net.

- condicional (ou rota seletiva): quando há uma escolha entre duas ou mais tarefas. Por exemplo, na WF-net da Figura 5, as transições $T3$ e $T5$ estão em uma situação condicional, somente uma das duas tarefas será executada;
- iterativa: quando é necessário executar uma mesma tarefa (ou conjunto de tarefas) múltiplas vezes. Por exemplo, na WF-net da Figura 5, as transições $T4$ e $T6$ podem ser executadas múltiplas vezes.

2.3 *WorkFlow net* Interorganizacional

Em ambientes colaborativos, organizações cooperam umas com as outras para alcançarem objetivos em comum. Processos de colaboração que envolvem múltiplas organizações podem ser representados como processos de *workflow* interorganizacional. Um processo de *workflow* interorganizacional é essencialmente um conjunto de processos de *workflow* fracamente acoplados (AALST, 2000). No geral, n parceiros de negócio são envolvidos em um mesmo processo de *workflow* local; portanto, o processo de *workflow* global consiste de processos de *workflow* locais mais uma estrutura de interação. De acordo com (AALST, 2000), uma forma de interação entre vários processos é dada através do uso de mecanismos de comunicação assíncrona por meio da troca de mensagens simples entre os processos de *workflow* locais.

Para modelar um processo de *workflow* interorganizacional, uma *WorkFlow net* Interorganizacional (IOWF-net) pode ser utilizada. De acordo com (AALST, 1998b), a definição formal de uma *WorkFlow net* Interorganizacional é apresentada como:

Definição 2.3.1 *Uma WorkFlow net Interorganizacional é baseada em uma tupla IOWF-net* $= (PN_1, PN_2, ..., PN_n, P_{AC}, AC)$*, onde:*

1. $n \in \mathbb{N}$ *é o número de WorkFlow nets Locais (LWF-nets);*

2. *para cada* $k \in \{1, ..., n\} : PN_k$ *é uma WF-net com lugar inicial* i_k *e lugar final* o_k*;*

3. P_{AC} *é o conjunto de elementos de comunicação assíncrona (lugares de comunicação);*

4. *AC corresponde as relações de comunicação assíncrona. Essas relações especificam um conjunto de transições de entrada e um conjunto de transições de saída para cada elemento de comunicação assíncrona.*

Para esclarecer os conceitos sobre IOWF-nets, um exemplo simples apresentado na Figura 6 é considerado. A IOWF-net apresentada na Figura 6 possui duas LWF-nets: A e B. Cada LWF-net possui somente um lugar inicial (iA para a LWF-A e iB para a LWF-B) e um lugar final (oA para a LWF-A e oB para a LWF-B). Os lugares PC1, PC2 e PC3 são lugares de comunicação. No lugar PC1, por exemplo, a transição de entrada é T1 e a transição de saída é T4.

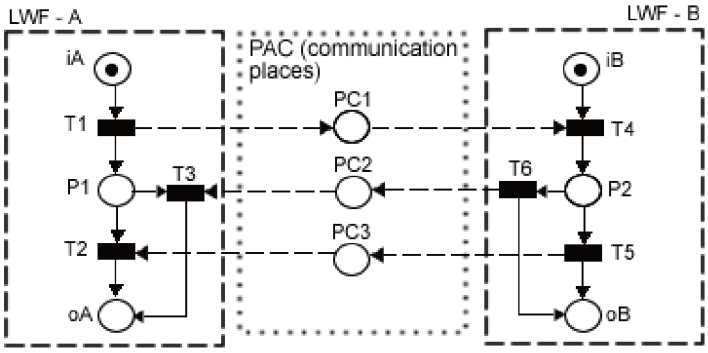


Figura 6 – Exemplo de uma IOWF-net.

Para transformar uma IOWF-net em uma WF-net simples, em (AALST, 1998b) foi definida a U(IOWF-net) (*Unfolded Interorganizational WorkFlow net*). Em uma U(IOWF-net), todas as LWF-nets são incluídas em um único processo de *workflow* considerando uma única transição inicial e uma única transição final. Um lugar inicial global 'i' e um lugar final global 'o' devem então ser adicionados a fim de respeitar a estrutura básica de uma WF-net simples, e os elementos de comunicação assíncrona existentes entre as diferentes LWF-nets são mapeados em lugares ordinários. A Figura 7 mostra a IOWF-net da Figura 6 transformada na U(IOWF-net) correspondente.

2.4 *t-time WorkFlow nets*

Para avaliar o desempenho de um processo modelado por uma rede de Petri é necessário considerar a modelagem explícita de restrições de tempo. Entre as várias extensões propostas para lidar com o tempo destacam-se os modelos de Ranchamdani's *Timed Petri nets* (RAMCHANDANI, 1973) e Merlin's *Time Petri nets* (MERLIN, 1974).

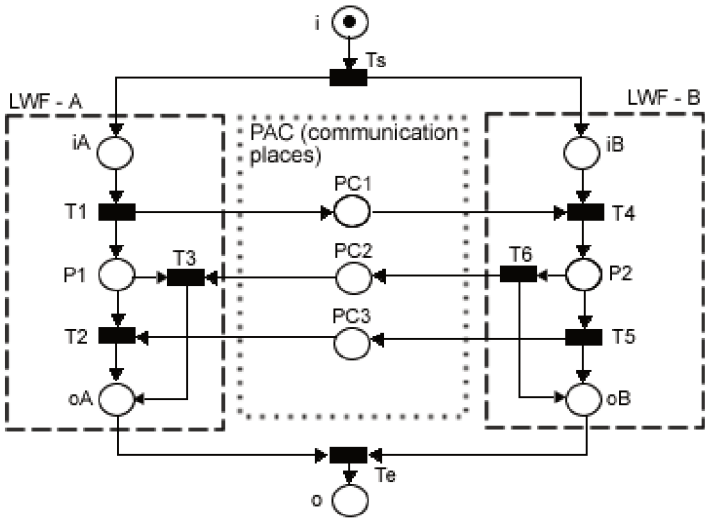


Figura 7 – Exemplo de uma U(IOWF-net).

No modelo de (MERLIN, 1974), as restrições de tempo são representadas por um intervalo $[\theta_{min}, \theta_{max}]$ associado a cada transição, onde este intervalo corresponde a uma duração de sensibilização imprecisa. Por exemplo, o intervalo [5, 11] associado a uma transição indica que esta será disparada em pelo menos cinco unidades de tempo depois de sua sensibilização e no máximo em onze unidades de tempo depois do instante de sensibilização da transição. Portanto, a tarefa relacionada com essa transição gastará entre 5 e 11 unidades de tempo para ser manipulada, isto é, o tempo mínimo gasto para manipular essa tarefa será de 5 unidades de tempo e no máximo de 11 unidades de tempo.

Nesse contexto, uma *t-time WorkFlow net* é uma *WorkFlow net* estendida com restrições de tempo associadas as transições, uma vez que nesse modelo as tarefas de execução serão associadas as transições.

De acordo com (LING; SCHMIDT, 2000), a definição formal de uma *t-time WorkFlow net* é apresentada como:

Definição 2.4.1 *Uma t-time WorkFlow net N é uma tupla (P, T, F, I), tal que:*

1. *(P, T, F) é uma WorkFlow net, onde P é um conjunto finito de lugares, T é um conjunto finito de transições e F é um conjunto finito de arcos;*

2. *I associa a cada transição $t \in T$ um intervalo de duração $I(t) = [\theta_{min(t)}, \theta_{max(t)}]$, onde $\theta_{min(t)}$ representa o tempo mínimo de disparo e $\theta_{max(t)}$ representa o tempo máximo de disparo da transição t após a sua sensibilização.*

A Figura 8 mostra um exemplo de uma *t-time WorkFlow net* em que um intervalo de tempo de duração é associado a cada transição $t \in T$. Por exemplo, a transição $T3$ possui o intervalo de tempo de duração [1, 4] associado a transição.

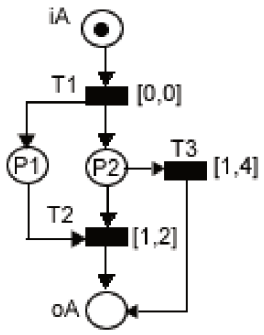


Figura 8 – Exemplo de uma *t-time WorkFlow net.*

2.5 Lógica Linear

A Lógica Linear é um refinamento da Lógica Clássica e Intuicionista, e foi introduzida por Jean-Yves Girard (GIRARD, 1987). Em vez de enfatizar a verdade, como na Lógica Clássica, ou provas, como na Lógica Intuicionista, a Lógica Linear enfatiza o papel das fórmulas como recursos.

A Lógica Linear introduz novos conectivos como: *par*($\parr$), *times* ($\otimes$), *with* (&), *plus* ($\oplus$) e *linear implication* ($\multimap$) (RIVIERE et al., 2001). No entanto, neste trabalho de pesquisa somente os conectivos *times* e *linear implication* serão utilizados.

O conectivo *times* ($\otimes$) representa a disponibilidade simultânea de recursos; por exemplo, '$A \otimes B$' representa a disponibilidade simultânea dos recursos 'A' e 'B'.

O conectivo *linear implication* ($\multimap$) representa um estado de mudança; por exemplo, '$A \multimap B$' denota que consumindo 'A', 'B' é produzido e depois da produção de 'B', 'A' não estará mais disponível.

Alguns estudos têm mostrado o relacionamento entre as teorias da rede de Petri e da Lógica Linear, como por exemplo em (GIRAULT; PRADIER-CHEZALVIEL; VALETTE, 1997) e (RIVIERE et al., 2001), uma vez que há uma tradução quase direta entre a estrutura de uma rede de Petri em fórmulas da Lógica Linear. Para traduzir um modelo de rede de Petri em fórmulas da Lógica Linear a seguinte definição foi apresentada em (RIVIERE et al., 2001).

Definição 2.5.1 *Tradução de uma rede de Petri em fórmulas da Lógica Linear:*

- *uma marcação M é um monômio em $\otimes$ sendo representada por $M = A_1 \otimes A_2 \otimes ... \otimes A_k$, onde A_i é um nome de lugar;*
- *uma transição é uma expressão da forma $M_1 \multimap M_2$, onde M_1 e M_2 são marcações;*
- *um sequente $M, s_i \vdash M'$ representa um cenário onde M e M' são respectivamente a marcação inicial e final, e s_i é a lista de transições não ordenadas.*

Para exemplificar a tradução de uma rede de Petri para a Lógica Linear, considere o exemplo da WF-net apresentada na Figura 8. A WF-net apresentada na Figura 8 possui inicialmente apenas um lugar marcado (iA) sendo, portanto, a marcação inicial $M = iA$. As transições são representadas pelas seguintes fórmulas da Lógica Linear: $T1 = iA \multimap P1 \otimes P2$, $T2 = P1 \otimes P2 \multimap oA$ e $T3 = P2 \multimap oA$. A WF-net apresentada na Figura 8 possui dois possíveis cenários: o primeiro cenário, considera o disparo das transições T1 e T2 sendo, portanto, representado pelo sequente $iA, T1, T2 \vdash oA$; o segundo cenário considera o disparo das transições T1 e T3 sendo, portanto, representado pelo sequente $iA, T1, T3 \vdash oA$.

Um sequente pode ser provado através de uma árvore de prova construída através da aplicação das regras de cálculo de sequente. Neste trabalho de pesquisa, somente três regras serão utilizadas. Considere que F, G e H são fórmulas e que Γ e Δ são blocos de fórmulas (RIVIERE et al., 2001):

- a regra $\multimap_L$, dada por $\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H} \multimap_L$, expressa o disparo de uma transição e gera dois sequentes, sendo que o sequente à direita representa o subsequente restante a ser provado e o sequente à esquerda representa as fichas consumidas pelo disparo da transição. Por exemplo, considerando o disparo da transição $T1 = iA \multimap P1 \otimes P2$ da WF-net mostrada na Figura 8, dois sequentes são gerados: $iA \vdash iA$ representando as fichas consumidas por esse disparo e a marcação $P1 \otimes P2$ que fará parte do subsequente remanescente a ser provado;

- a regra $\otimes_L$, dada por $\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L$, é usada para transformar uma marcação em uma lista de átomos. Por exemplo, o subsequente $P1 \otimes P2$ gerado pelo disparo da transição $T1 = iA \multimap P1 \otimes P2$ da WF-net mostrada na Figura 8, usará a regra $\otimes_L$ para ser transformada em uma lista de átomos $P1$ and $P2$;

- a regra $\otimes_R$, dada por $\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Delta, \Gamma \vdash F \otimes G} \otimes_R$, transforma um sequente do tipo $A, B \vdash A \otimes B$ em dois sequentes identidades $A \vdash A$ e $B \vdash B$. Por exemplo, considerando o disparo da transição $T2 = P1 \otimes P2 \multimap oA$ da WF-net mostrada na Figura 8, o sequente que representa as fichas consumidas por esse disparo $P1, P2 \vdash P1 \otimes P2$ também necessita ser tratado usando a regra $\otimes_R$, isto é, $\frac{P1 \vdash P1 \quad P2 \vdash P2}{P1, P2 \vdash P1 \otimes P2} \otimes_R$.

Para ilustrar a construção de uma árvore de prova da Lógica Linear, o sequente $iA, T1, T2 \vdash oA$ da WF-net mostrada na Figura 8 é considerado. A árvore de prova da Lógica Linear para este cenário é a seguinte:

$$\dfrac{\dfrac{\dfrac{P1 \vdash P1 \quad P2 \vdash P2}{P1, P2 \vdash P1 \otimes P2}\otimes_R \quad oA \vdash oA}{P1, P2, P1 \otimes P2 \multimap oA \vdash oA}\multimap_L}{iA \vdash iA \quad P1 \otimes P2, T2 \vdash oA}\otimes_L}{iA, iA \multimap P1 \otimes P2, T2 \vdash oA}\multimap_L$$

Uma árvore de prova da Lógica Linear é lida de baixo para cima. A prova é finalizada quando o sequente identidade correspondente ao lugar final é produzido, no caso do exemplo mostrado acima este sequente é $oA \vdash oA$; quando não há nenhuma regra que possa ser aplicada; ou quando todas as folhas da árvore de prova são sequentes identidade (sequentes do tipo $A \vdash A$).

As relações de causalidade, e assim de precedência, entre os eventos de um cenário obtidas pela rotulação de uma árvore de prova podem ser representadas na forma de um grafo, chamado de grafo de precedência (DIAZ, 2013). Um grafo de precedência representa uma ordem parcial que o disparo das transições deve seguir; isso significa que uma ação t_j não pode iniciar até que uma ação anterior t_i termine, ou seja, $t_i \longrightarrow t_j$ (t_i precede t_j).

Definição 2.5.2 *Um grafo de precedência é um par ordenado G = (V, E) onde V é um conjunto de vértices (ou nós) e E é um conjunto de arestas direcionadas (ou arcos), tal que:*

- *o conjunto V de vértices representa um conjunto de ações/tarefas ($t_1, t_2, ..., t_k$);*
- *o conjunto E de arestas direcionadas representa um conjunto de sequentes identidade (relação entre um evento que produziu um átomo e o evento que consumiu o mesmo átomo);*
- *os nós iniciais i_i representam as marcações iniciais;*
- *os nós finais f_i representam as marcações finais.*

Nesse trabalho de pesquisa, as redes de Petri, representadas pela Lógica Linear, são na realidade *WorkFlow nets*. Portanto, nos grafos de precedência somente um nó inicial (i_1) e um somente um nó final (f_1) serão representados, uma vez que em *WorkFlow nets* há somente uma marcação inicial e uma marcação final.

Para rotular uma árvore de prova, toda vez que a regra $\multimap_L$ é aplicada, a transição correspondente t_i rotula a aplicação da regra, bem como os átomos produzidos e consumidos. Além disso, o evento inicial deve ser rotulado por i_1 e o evento final deve ser rotulado por f_1. Uma vez que a rotulação foi realizada, cada sequente identidade representa a associação de duas visões do mesmo átomo: a parte da esquerda é rotulada pelo evento que o produziu e a parte da direita é rotulada pelo evento que o consumiu. Os rótulos

são mostrados acima dos átomos e abaixo das regras $\multimap_L$. Na sequência é apresentado um exemplo de uma árvore de prova rotulada referente ao sequente $iA, T1, T2 \vdash oA$ da WF-net mostrada na Figura 8.

$$\dfrac{\dfrac{\dfrac{\overset{T_1}{P1}\vdash\overset{T_2}{P1} \quad \overset{T_1}{P2}\vdash\overset{T_2}{P2}}{\overset{T_1}{P1},\overset{T_1}{P2}\vdash\overset{T_2}{P1\otimes P2}}\otimes_R \quad \overset{T_2}{oA}\vdash\overset{f_1}{oA}}{\overset{T_2}{P1},P2,P1\otimes P2\multimap oA\vdash\overset{f_1}{oA}}\multimap_L}{\overset{i_1}{iA}\vdash\overset{T_1}{iA} \quad \overset{T_1}{P1\otimes P2},\overset{T_1}{T2}\vdash\overset{f_1}{oA}}\otimes_L}{\overset{i_1}{iA},iA\multimap P1\otimes P2,T2\vdash\overset{f_1}{oA}}\multimap_L$$

A Figura 9 mostra o grafo de precedência referente ao sequente $iA, T1, T2 \vdash oA$ da WF-net mostrada na Figura 8 construído a partir da árvore de prova da Lógica Linear rotulada mostrada acima. Como pode ser observado no grafo, o nó inicial i_1 representa a marcação inicial, sendo que a partir desse nó é possível disparar a transição T_1, onde o átomo iA é produzido por i_1 e consumido por T_1. A partir do nó T_1 é possível disparar a transição T_2 produzindo e consumindo o átomo P_1 ou o átomo P_2. A partir do nó T_2 a marcação final f_1 é alcançada produzindo o átomo oA.

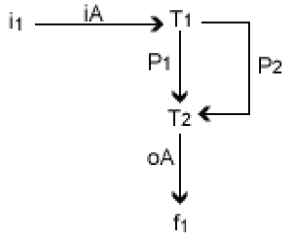


Figura 9 – Exemplo de um grafo de precedência.

Em uma árvore de prova da Lógica Linear, cada disparo de transição de uma *t-Time Petri net* pode gerar uma data simbólica associada a cada átomo (ficha) (RIVIERE et al., 2001). Para isso, nesse trabalho de pesquisa, D_i representará uma data e d_i uma duração associada a um disparo de uma transição (t_i). Um par (D_p, D_c), representando respectivamente as datas de produção e de consumo de um átomo, será associado a cada átomo. O cálculo de datas em árvores de prova canônica é dado pelos seguintes passos (RIVIERE et al., 2001):

- determinar uma data de produção D_i para todas as fichas da marcação inicial;
- para cada instância da regra $\multimap_L$, calcule a data de disparo desta transição: isto é igual ao maior valor de data de produção dos átomos consumidos por esta transição, acrescido pela duração de sensibilização d_i associada à transição considerada;
- atualizar as datas de todos os átomos que foram consumidos e produzidos.

A título de exemplo, a árvore de prova da Lógica Linear com datas simbólicas referente ao sequente $iA, T1, T2 \vdash oA$ da WF-net mostrada na Figura 8 é apresentada a seguir:

$$\dfrac{\dfrac{\dfrac{P1(D_1+d_1,D_1+d_1+d_2)\vdash P1 \quad P2(D_1+d_1,D_1+d_1+d_2)\vdash P2}{P1(D_1+d_1,.),P2(D_1+d_1,.)\vdash P1\otimes P2}\otimes_R \quad oA\vdash oA}{P1(D_1+d_1,.),P2(D_1+d_1,.),P1\otimes P2\multimap oA\vdash oA}\multimap_L}{iA(D_1,D_1+d_1)\vdash iA \quad P1(D_1+d_1,.)\otimes P2(D_1+d_1,.),T2\vdash oA}\otimes_L$$
$$\multimap_L$$
$$iA(D_1,.),iA\multimap P1\otimes P2,T2\vdash oA$$

Para melhor visualização, as datas simbólicas de produção e de consumo obtidas a partir das árvores de prova da Lógica Linear podem ser representadas na forma de tabelas. A Tabela 1, por exemplo, apresenta as datas simbólicas de produção e de consumo para a árvore de prova da Lógica Linear com datas simbólicas mostrada acima.

Átomos	Datas de Produção	Datas de Consumo
iA	D_1	$D_1 + d_1$
$P1$	$D_1 + d_1$	$D_1 + d_1 + d_2$
$P2$	$D_1 + d_1$	$D_1 + d_1 + d_2$
oA	$D_1 + d_1 + d_2$	*desconhecido*

Tabela 1 – Exemplo de datas simbólicas de produção e de consumo.

Assim como no modelo de uma *t-time Petri net*, uma duração de sensibilização d_i no cálculo de datas simbólicas nas árvores de prova da Lógica Linear tem um valor que pertence a um intervalo de tempo $\Delta_i = [\delta_i min, \delta_i max]$ (RIVIERE et al., 2001) que também podem ser representados na forma de tabelas. Continuando com o exemplo do sequente $iA, T1, T2 \vdash oA$ da WF-net mostrada na Figura 8, a Tabela 2 mostra os intervalos de datas simbólicas para este sequente.

Átomos	Intervalos de Datas Simbólicas de Produção	Intervalos de Datas Simbólicas de Consumo
iA	D_1	$[D_1 + d_{1min}, D_1 + d_{1max}]$
$P1$	$[D_1 + d_{1min}, D_1 + d_{1max}]$	$[D_1 + d_{1min} + d_{2min}, D_1 + d_{1max} + d_{2max}]$
$P2$	$[D_1 + d_{1min}, D_1 + d_{1max}]$	$[D_1 + d_{1min} + d_{2min}, D_1 + d_{1max} + d_{2max}]$
oA	$[D_1 + d_{1min} + d_{2min}, D_1 + d_{1max} + d_{2max}]$	*desconhecido*

Tabela 2 – Exemplo de intervalos de datas simbólicas de produção e de consumo.

2.6 Propriedade *Soundness*

A corretude de um processo de *workflow* é associada à verificação da propriedade *Soundness* (AALST, 1996b). Essa propriedade garante que todas as instâncias do processo de uma WF-net serão corretamente tratadas e finalizadas, e que todas as atividades definidas no processo serão executadas. Segundo (AALST, 1996b), uma *WorkFlow net* é *sound* se, e somente se, satisfaz os seguintes requisitos:

- ❏ para cada ficha colocada no lugar de início 'i', apenas uma ficha aparece no lugar de término 'o';
- ❏ quando uma ficha aparece no lugar 'o', todos os outros lugares estão vazios;
- ❏ considerando uma tarefa associada a uma transição, é possível evoluir da marcação inicial até uma marcação que sensibiliza tal transição, ou seja, não deve haver nenhuma transição "morta" na *WorkFlow net*.

Muitos estudos têm considerado a análise da propriedade *Soundness* em processos de *workflow* interorganizacional como em (AALST, 1998b) e (PASSOS; JULIA, 2013). É interessante para as organizações, quando o critério de correção *Soundness* clássico não é satisfeito para uma dada IOWF-net, identificar os cenários que ainda assim satisfazem os requisitos do negócio. Nesse contexto, variantes do critério *Soundness* foram propostos. Em (PASSOS, 2016), por exemplo, foi definida uma abordagem para identificar se uma IOWF-net é *Sound*, *Relaxed Sound* ou *Weak Sound*. Uma IOWF-net é *Relaxed Sound* se cada tarefa do processo é considerada em pelo menos um dos cenários que podem ser corretamente finalizados. Uma IOWF-net é *Weak Sound* se não há *deadlock* no modelo e a correta conclusão do processo é garantida, mesmo em casos onde o processo possui tarefas que não são executadas, ou seja, tarefas "mortas".

Em uma WF-net, uma situação de *deadlock* acontece quando para uma dada marcação M' nenhuma transição pode ser disparada (BANASZAK; KROGH, 1990), como mostra, por exemplo, a WF-net apresentada na Figura 10. Como pode ser observado na Figura 10, caso a transição T2 seja disparada, não será possível sensibilizar mais nenhuma outra transição; portanto, a rede se encontrará em um estado de *deadlock*.

De acordo com (BARKAOUI; ABDALLAH, 1995), 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.

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 do que entradas, o conjunto de lugares P' pode ficar livre de fichas podendo provocar uma situação de *deadlock*. Em particular, um sifão que não contém nenhuma ficha em uma determinada marcação M permanecerá sem fichas para qualquer marcação subsequente à M.

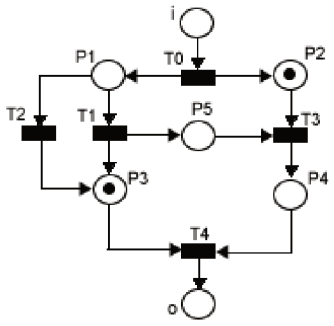


Figura 10 – Exemplo de uma WF-net em estado de *deadlock*.

No trabalho de (PASSOS; JULIA, 2013), as árvores de prova da Lógica Linear são usadas para analisar o critério *Soundness* em uma IOWF-net. Na análise, é necessário verificar a propriedade *Soundness* para cada LWF-net que compõe a IOWF-net e também para a U(IOWF-net). De acordo com (PASSOS; JULIA, 2013), uma IOWF-net representada pela Lógica Linear é *Sound* se respeitar as seguintes restrições:

1. para cada cenário de cada LWF-net analisada e também para cada cenário da U(IOWF-net), no final da árvore de prova deve-se verificar:

   a) se não há nenhum átomo disponível para consumo (significa que todos os lugares da WF-net estão vazios);

   b) se não há nenhuma fórmula de transição que não foi disparada;

   c) se apenas um sequente identidade $o \vdash o$ foi produzido (apenas uma ficha aparece no lugar final);

2. considerando todos os cenários da LWF-net analisada e também todos os cenários da U(IOWF-net), cada transição t_i deve, no mínimo, aparecer em um dos cenários.

Uma abordagem similar pode ser utilizada para verificar variantes do critério *Soundness* (*Relaxed Sound* e *Weak Sound*). De acordo com (PASSOS, 2016), considerando as restrições apresentadas anteriormente para a verificação da propriedade *Soundness*, uma IOWF-net é *Relaxed Sound* se as árvores de prova dos cenários da U(IOWF-net) respeitam as restrições 1-a e 1-c, e para cada cenário que satisfaz essa restrição, a restrição 2 também é respeitada. Uma IOWF-net é *Weak Sound* se todas as árvores de prova dos cenários da U(IOWF-net) respeitam a restrição 1-a e 1-c.

2.7 Bissimulação *Branching*

A ideia de bissimilaridade foi primeiro introduzida em (PARK, 1981) e pode ser interpretada como: dois processos são equivalentes se e somente se eles podem sempre copiar ou simular as ações uns dos outros. Bissimilaridade não faz distinção entre ações externas

(observáveis) e ações internas (silenciosas). Portanto, bissimilaridade não é um conceito de equivalência adequado para processos com comportamento interno.

Bissimilaridade *branching* foi primeiro introduzida em (GLABBEEK; WEIJLAND, 1989) e é uma variante de bissimilaridade distinguindo comportamento externo de comportamento interno. A distinção entre comportamento externo e interno captura a ideia de que um ambiente observando dois processos não visualiza nenhuma diferença em seus comportamentos, enquanto que internamente os dois processos realizam computações diferentes (BASTEN, 1998). Para distinguir entre comportamento externo e interno, ações silenciosas podem ser introduzidas. Ações silenciosas são ações que não podem ser observadas e usualmente são denotadas pelo símbolo τ. A Figura 11, apresentada em (BASTEN, 1998), mostra a essência do conceito de bissimulação *branching*.

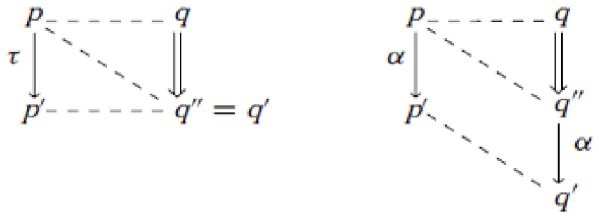


Figura 11 – A essência do conceito de bissimulação *branching* (BASTEN, 1998).

Na Figura 11 as linhas tracejadas representam uma bissimulação *branching*; τ representa uma ação silenciosa; α representa uma ação observável; p, q, p', q', q'' representam processos; e a relação $\Longrightarrow$ consiste de um número arbitrário (≥ 0) de τ-etapas. No lado esquerdo da Figura 11, o processo p pode evoluir em outro processo p' executando uma ação silenciosa (τ) e o processo q pode evoluir em outro processo q'' executando uma sequência de zero ou mais ações silenciosas. Assim, é possível observar no lado esquerdo da figura que o processo p possui uma relação de equivalência com o processo q e q'', e o processo p' possui uma relação de equivalência com o processo q''. Esse fato claramente afirma que dois processos, que são equivalentes, continuarão equivalentes depois da introdução adicional de alguma ação silenciosa (τ) em um dos processos ou mesmo em ambos.

No lado direito da Figura 11, o processo p pode evoluir em outro processo p' executando uma ação observável (α), o processo q pode evoluir em outro processo q'' executando uma sequência de zero ou mais ações silenciosas (τ) e o processo q'' pode evoluir em outro processo q' executando uma ação observável (α). Assim, é possível observar no lado direito da figura, que o processo p possui uma relação de equivalência com o processo q e q'', e o processo p' possui uma relação de equivalência com o processo q'. Esses fatos claramente afirmam que dois processos que são equivalentes, continuarão equivalentes, depois da introdução adicional de alguma ação observável em um dos processos, somente se a mesma ação observável também existir no outro processo e a mesma sequência de restrições é respeitada.

No próximo capítulo é definido como a noção de bissimulação *branching* é usada nesse

trabalho de pesquisa.

Capítulo 3

Verificação de Requisitos Funcionais e não Funcionais em SOA

Neste capítulo, os métodos para a verificação dos requisitos de serviços, que representam o comportamento do modelo (requisitos funcionais), e dos requisitos de desempenho (requisitos não funcionais) em SOA são formalizados. Para isso, as definições dos modelos de requisitos e de arquitetura são apresentadas na seção 3.1. O método para a verificação dos requisitos de serviços em SOA é definido na seção 3.2. Já na seção 3.3 é apresentado o método para a verificação de desempenho dos requisitos de serviços em SOA. Na seção 3.4 é apresentada uma análise da complexidade dos métodos propostos.

3.1 Definição dos modelos de requisitos e de arquitetura

O modelo de requisitos, no contexto de SOA, considerado neste trabalho de pesquisa é representado por uma WF-net que especifica as funcionalidades, ou seja, os serviços que são de interesse de todas as partes envolvidas no processo (pode ser visto também como um contrato entre essas partes envolvidas); por isso, é considerado como um modelo público ou uma *public* WF-net. Não é definido nesta pesquisa como o modelo de requisitos é obtido, ou seja, é considerado um modelo de requisitos já com os relacionamentos estabelecidos entre as partes públicas envolvidas.

Definição 3.1.1 *Uma public WF-net* N^{publ} *é uma tupla* $(P^{publ}, T^{publ}, F^{publ})$*, tal que:*

1. $(P^{publ}, T^{publ}, F^{publ})$ *é uma WorkFlow net de interesse público;*
2. N^{publ} *é uma WF-net sound.*

Neste trabalho de pesquisa, um modelo de SOA é considerado como um conjunto de processos privados que interagem entre si através de mecanismos de comunicação assín-

crona a fim de produzir os serviços especificados no modelo de requisitos correspondente. Portanto, os processos privados são representados por *private* WF-nets e a arquitetura, que é composto por um conjunto de *private* WF-nets, é representado por uma IOWF-net. Como a arquitetura contém ações adicionais as quais são somente de interesse local, em alguns momentos, a arquitetura é referenciada apenas como modelo privado.

Definição 3.1.2 *Uma private WF-net* N^{priv} *é uma tupla* $(P^{priv}, T^{priv}, F^{priv})$*, tal que* $(P^{priv}, T^{priv}, F^{priv})$ *é uma WorkFlow net de interesse local.*

Definição 3.1.3 *Uma IOWF-net é uma tupla* $(N_1^{priv}, N_2^{priv}, ..., N_n^{priv}, P_{AC}, AC)$*, tal que:*

1. $n \in \mathbb{N}$ *é o número de private WF-nets;*
2. *para cada* $k \in \{1, ..., n\}$: N_k^{priv} *é uma private WF-net com lugar inicial* i_k *e lugar final* o_k*;*
3. P_{AC} *é o conjunto de elementos de comunicação assíncrona (lugares de comunicação);*
4. AC *corresponde as relações de comunicação assíncrona em que é especificado um conjunto de transições de entrada e um conjunto de transições de saída para cada elemento de comunicação assíncrona.*

Para analisar uma arquitetura representada por uma IOWF-net, um lugar inicial global 'i' e um lugar final global 'o' será adicionado com o propósito de respeitar a estrutura básica de uma simples WF-net como explicado na seção 2.3. Portanto, a arquitetura final será uma U(IOWF-net).

Neste trabalho de pesquisa, serviços são considerados como cenários. Os serviços, que correspondem aos requisitos comportamentais especificados no modelo de análise de requisitos, são representados pelos cenários da *public* WF-net, e os serviços do modelo de SOA são representados pelos cenários da U(IOWF-net). Um cenário no contexto de um processo de *workflow* corresponde a uma rota bem definida mapeada na *public* WF-net correspondente, e um cenário no contexto de um processo de *workflow* interorganizacional corresponde a uma rota bem definida na U(IOWF-net) correspondente. Se a *public* WF-net ou a U(IOWF-net) tem mais de uma rota (lugares com dois ou mais arcos de saída), mais de um cenário deve então ser considerado.

No contexto da Lógica Linear, um cenário de uma *public* WF-net ou de uma U(IOWF-net) é dado por um sequente linear que deve se provado através da construção da árvore de prova da Lógica Linear. Conforme explicado na seção 2.5, a partir da árvore de prova da Lógica Linear é possível construir o grafo de precedência correspondente ao sequente que foi provado; portanto, cada cenário de uma *public* WF-net ou de uma U(IOWF-net) também pode ser representado por um grafo de precedência.

Para encontrar os cenários que devem ser provados através da árvore de prova da Lógica Linear, componentes repetitivos estacionários são calculados. Conforme apresentado na seção 2.1, os componentes repetitivos estacionários estabelecem uma condição necessária sobre o número de vezes que cada transição deverá ser disparada para que, eventualmente, um invariante de transição seja encontrado de acordo com a estrutura da rede de Petri. Para o cálculo dos componentes repetitivos estacionários, uma transição ligando o lugar final ao lugar inicial é adicionada na WF-net para transformá-la em uma rede de Petri cíclica. Quando a rede de Petri cíclica volta a sua marcação inicial após uma execução, significa que um cenário livre de bloqueio foi executado e este cenário é dado pelo componente repetitivo estacionário encontrado. Portanto, o conjunto de transições pertencentes a um componente repetitivo estacionário produzirá um candidato a um cenário comportamental que poderá representar um possível requisito de serviço. É importante ressaltar que após a obtenção dos componentes repetitivos estacionários, a transição ligando o lugar final ao lugar inicial é desconsiderada e a rede de Petri cíclica volta a ser uma WF-net.

Considere, por exemplo, a WF-net da Figura 5. Adicionando uma transição (T8) que liga o lugar final 'o' ao lugar inicial 'i' a rede de Petri cíclica da Figura 12 é obtida.

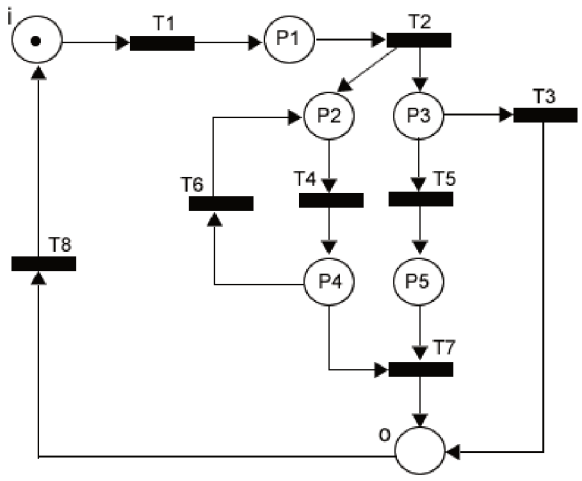


Figura 12 – Exemplo de uma WF-net transformada em uma rede de Petri cíclica.

Aplicando a análise de invariantes da ferramenta PIPE na rede de Petri da Figura 12, por exemplo, os componentes repetitivos estacionários apresentados na Figura 13 são encontrados. O primeiro vetor da Figura 13 corresponde a uma lista não ordenada de transições que devem ser disparadas para que a rede volte a sua marcação inicial, sendo que o disparo de uma transição é representado pelo número 1 (cada transição foi disparada uma única vez). Desse modo, esse componente repetitivo estacionário será considerado como candidato à um cenário que será representado por um sequente da Lógica Linear e provado através da construção da árvore de prova. O segundo vetor da Figura 13 mostra o disparo apenas das transições T4 e T6 que, neste caso, estão representando a existência

de um ciclo iterativo que pode acontecer com o disparo dessas duas transições.

T1	T2	T3	T4	T5	T6	T8	T7
1	1	0	1	1	0	1	1
0	0	0	1	0	1	0	0

Figura 13 – Exemplo do cálculo dos componentes repetitivos estacionários para a rede de Petri da Figura 12.

Para a *public* WF-net, todos os componentes repetitivos estacionários encontrados que fazem a rede voltar a sua marcação inicial serão considerados como cenários que deverão ser representados por sequentes da Lógica Linear e provados pela construção da árvore de prova da Lógica Linear. Para os sequentes corretamente finalizados, os grafos de precedência deverão ser gerados. Na *public* WF-net todos os cenários encontrados serão considerados, pois representam todos os requisitos de serviço especificados pelo modelo de análise de requisitos.

A transição que foi adicionada ligando o lugar final ao lugar inicial para transformar a WF-net em uma rede de Petri cíclica não é considerada na representação do sequente. No caso do componente repetitivo estacionário encontrado na Figura 13, a transição $T8$ não será considerada no sequente. Assim, para o exemplo da rede de Petri mostrada na Figura 12, o componente repetitivo estacionário mostrado na Figura 13 é representado pelo seguinte sequente da Lógica Linear:

$$i, T1, T2, T4, T5, T7 \vdash o$$

A árvore de prova da Lógica Linear rotulada com as transições de disparo para o sequente mostrado acima é a seguinte:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{i^{1}\ i^{T1}\vdash i}\quad \dfrac{}{P4^{T4}\vdash P4^{T7}}\quad \dfrac{}{P5^{T5}\vdash P5^{T7}}}{P4,P5\vdash P4\otimes P5}\otimes_R \quad \dfrac{}{o^{f1}\vdash o^{T7}}}{i,i\multimap P1,P4,P5\vdash o^{f1}}\multimap_{L}^{T1} \quad \dfrac{}{P3^{T2}\vdash P3^{T5}}}{i,i\multimap P1,P3,P4,P5,P4\otimes P5\multimap o\vdash o^{f1}}\multimap_{L}^{T5} \quad \dfrac{}{P2^{T2}\vdash P2^{T4}}}{i,i\multimap P1,P2,P3,P4,P3\multimap P5,T7\vdash o^{f1}}\multimap_{L}^{T4} \quad \dfrac{}{P3^{T2}\vdash P3^{T2}}}{i,i\multimap P1,P2,P3,P2\multimap P4,T5,T7\vdash o^{f1}}\otimes_L \quad \dfrac{}{P1^{T1}\vdash P1^{T2}}}{i,i\multimap P1,P2\otimes P3,T4,T5,T7\vdash o^{f1}}\multimap_{L}^{T2} \quad \dfrac{}{P2^{T2}\vdash P2^{T2}}}{i,i\multimap P1,P1\multimap P2\otimes P3,T4,T5,T7\vdash o^{f1}}\multimap_{L}^{T1}$$

$$i^{1}\ i,i\multimap P1,T2,T4,T5,T7\vdash o$$

O grafo de precedência gerado a partir da árvore de prova rotulada para o sequente $i, T1, T2, T4, T5, T7 \vdash o$ é mostrado na Figura 14. Conforme explicado na seção 2.5, um grafo de precedência é construído a partir da árvore de prova rotulada, ou seja, toda vez que a regra $\multimap_L$ é aplicada, a transição t_i correspondente rotula a aplicação da regra, bem como os átomos produzidos e consumidos.

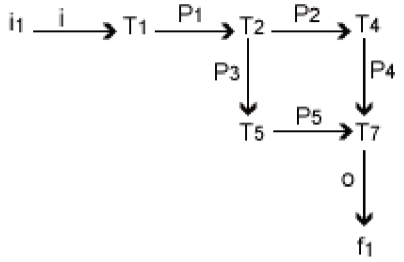


Figura 14 – Grafo de precedência referente ao sequente $i, T1, T2, T4, T5, T7 \vdash o$.

Para a U(IOWF-net), todos os componentes repetitivos estacionários encontrados que fazem a rede voltar a sua marcação inicial também serão considerados como cenários. No entanto, somente os cenários que potencialmente possuam equivalência com algum dos cenários encontrados para a *public* WF-net deverão ser representados por sequentes da Lógica Linear e provados pela construção da árvore de prova da Lógica Linear. Consequentemente, somente para estes potenciais cenários, os grafos de precedência serão gerados. Para encontrar estes potenciais cenários, deve-se analisar cada componente repetitivo estacionário da U(IOWF-net) verificando se contém todas as transições de um dos componentes repetitivos estacionários da *public* WF-net. Se o resultado da comparação for positivo, significa que o cenário correspondente ao componente repetitivo estacionário da U(IOWF-net) analisado é um candidato à equivalência por bissimulação de um dos cenários da *public* WF-net.

3.2 Verificação dos requisitos de serviços em SOA

Conforme definido na seção 3.1, os requisitos de serviços, que representam o comportamento dos modelos, são primeiramente identificados como cenários na *public* WF-net e na U(IOWF-net) através do cálculo dos componentes repetitivos estacionários. Posteriormente esses cenários são representados por sequentes da Lógica Linear que devem ser provados pela construção das árvores de prova. A partir das árvores de prova corretamente finalizadas, os grafos de precedência, que representam uma ordem parcial das ações que um serviço realiza, são construídos. Portanto, ao final desse processo, os cenários da *public* WF-net e da U(IOWF-net) serão representados pelos respectivos grafos de precedência.

Os grafos de precedência, que representam os cenários da arquitetura provavelmente, conterão ações que são de interesse somente da respectiva *private* WF-net. Essas ações,

consideradas como ações silenciosas, e os arcos ligados a elas serão representados por linhas tracejadas nos grafos de precedência referentes ao modelo arquitetural.

Os grafos de precedência também podem ser vistos como um tipo de semântica operacional associada a um processo de *workflow*, uma vez que mostram formalmente um tipo de simulação simbólica de um conjunto de ações desempenhadas por uma WF-net. A equivalência entre duas WF-nets (*public* WF-net e U(IOWF-net)) podem ser consequentemente verificadas usando o conceito de bissimulação *branching*, a qual tem o propósito de comparar semânticas operacionais de modelos comportamentais formais distintos. Desse modo, pode ser verificado se o comportamento de uma *public* WF-net é simulado pela U(IOWF-net) correspondente.

Considerando a noção de bissimulação *branching* apresentada na seção 2.7, a definição de bissimulação *branching* entre dois grafos apresentada em (GLABBEEK; WEIJLAND, 1989) e a definição de grafo de precedência apresentada na seção 2.5, uma definição de semântica operacional é dada, a qual será utilizada para verificar a equivalência de comportamento entre os modelos de arquitetura e de requisitos definidos neste trabalho.

Antes de definir a semântica operacional usada neste trabalho de pesquisa, duas definições auxiliares são necessárias. Os seguintes conjuntos e elementos devem ser considerados:

- G_r é um conjunto de grafos de precedência de um modelo de requisitos e G_a é um conjunto de grafos de precedência de uma arquitetura;
- A_r é um conjunto de ações observáveis de um modelo de requisitos e A_a é definido como $A \cup A_\tau$, onde A é o conjunto de ações observáveis e A_τ é o conjunto de ações silenciosas de uma arquitetura;
- $a_r, a'_r \in A_r$ e $a_a, a'_a \in A_a$.

Definição 3.2.1 *A relação '$\longrightarrow$' define um caminho direto entre dois nós (ações) de um grafo de precedência. Por exemplo, $a_r \longrightarrow a'_r$ indica que o nó a_r está diretamente relacionado com o nó a'_r, ou seja, não há subcaminhos entre a_r e a'_r.*

Definição 3.2.2 *A relação '$\Longrightarrow$' define um caminho entre dois nós (ações) de um grafo de precedência com uma ou mais ações silenciosas (τ) entre esses dois nós. Por exemplo, $a_a \Longrightarrow a'_a$ indica que há uma ou mais ações τ entre a_a e a'_a, ou seja, $a_a \longrightarrow \tau \longrightarrow ... \ a'_a$.*

Considerando que os cenários comportamentais dos modelos de requisitos e de arquitetura são representados por grafos de precedência, a verificação da equivalência comportamental entre os modelos é baseada na seguinte definição.

Definição 3.2.3 *Dois grafos de precedência $g_r \in G_r$ e $g_a \in G_a$ são bissimilares branching (notação $g_r \sim_b g_a$) se existe uma relação binária R (chamada bissimulação branching) entre os nós de g_r e g_a, tal que:*

- *as raízes são relacionadas por* $R \subseteq nós(g_r) \times nós(g_a)$ *considerando que as ações iniciais de dois grafos de precedência devem ser as mesmas (as ações estão relacionadas aos nós dos grafos);*
- *se* $a_r \ R \ a_a$ *e* $a_r \longrightarrow a'_r$*, então* $a_a \longrightarrow a'_a$ *e* $a'_r \ R \ a'_a$ *ou existe um caminho entre* a_a *e* a'_a *com uma ou mais ações silenciosas* $(a_a \Longrightarrow \tau \longrightarrow a'_a)$ *de modo que* $a_r \ R \ \tau$ *e* $a'_r \ R \ a'_a$*.*

A Figura 15 mostra a essência da noção de bissimulação *branching* aplicada neste trabalho de pesquisa (definição 3.2.3).

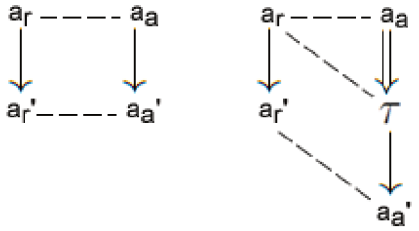


Figura 15 – A essência da noção de bissimulação *branching* com grafos de precedência.

De acordo com a definição 3.2.3, as ações silenciosas não comprometem a equivalência entre dois grafos de precedência, pois uma ação a'_r (gerada após a execução da ação a_r, $a_r \longrightarrow a'_r$) terá uma relação R com uma ação a'_a após a execução de uma ou mais ações silenciosas $(a_a \Longrightarrow \tau \longrightarrow a'_a)$. Desse modo, para facilitar a verificação de equivalência entre os modelos de requisitos e de arquitetura, os grafos de precedência referentes a U(IOWF-net) podem ser simplificados removendo as ações silenciosas. Para isso, considere t_j como uma ação silenciosa, t_{j-1} a ação que precede t_j, t_{j+1} a ação que procede t_j, x e y os rótulos dos arcos entre as ações $(t_{j-1} \xrightarrow{x} t_j \xrightarrow{y} t_{j+1})$. As regras de redução são as seguintes:

- a remoção de uma ação silenciosa t_j resulta na criação de um novo arco entre as ações t_{j-1} e t_{j+1} $(t_{j-1} \longrightarrow t_{j+1})$;
- o novo arco é rotulado pela junção dos rótulos dos arcos entre t_{j-1} e t_j e entre t_j e t_{j+1} $(t_{j-1} \xrightarrow{xy} t_{j+1})$.

Quando as ações silenciosas são removidas dos grafos de precedência referentes a uma U(IOWF-net), os grafos simplificados obtidos devem ser os mesmos da *public* WF-net correspondente, quando o comportamento da arquitetura reproduz o comportamento do modelo de requisitos. Após a redução de um grafo de precedência referente a uma U(IOWF-net), arcos adicionais ainda podem aparecer no grafo. No entanto, tais arcos não comprometem a equivalência com o grafo de precedência da *public* WF-net, caso sejam arcos redundantes. Arcos redundantes especificam uma restrição de precedência que já existe no grafo de precedência. Portanto, um arco redundante não muda o comportamento do grafo de precedência podendo, então, ser desconsiderado.

Definição 3.2.4 *Arcos redundantes: Seja $G = (V, E)$ uma grafo de precedência, onde V é um conjunto de nós (ações) e E é um conjunto de arestas direcionadas (arcos).*

Um arco $e \in E$ conectando dois nós ($t_j, t_k \in V$), que possui uma relação de precedência ($t_j \xrightarrow{e} t_k$), é chamado redundante em G se existe outro caminho entre os mesmos nós ($t_j \longrightarrow ... \longrightarrow t_k$).

O grafo de precedência da Figura 16 (b), por exemplo, produz a mesma sequência de ações do grafo de precedência da Figura 16 (a); no entanto, o grafo de precedência da Figura 16 (b) possui um arco adicional (P_5). O arco adicional P_5 é simplesmente uma restrição redundante indicando que a ação T_1 deve ocorrer antes da ação T_3. A arco P_5 é considerado redundante, pois existe outra sequência de ações ($T_1 \longrightarrow T_2 \longrightarrow T_3$) indicando que a ação T_1 deve ocorrer antes da ação T_3. Portanto, o arco P_5 pode ser desconsiderado sem modificar a equivalência entre os grafos.

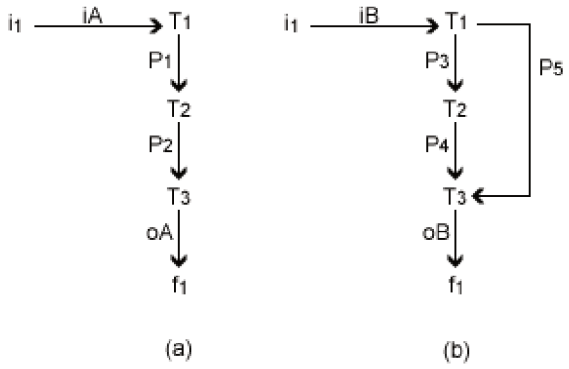


Figura 16 – Exemplo de arco redundante.

As etapas do método para formalmente verificar se arquitetura representada por uma U(IOWF-net), no contexto de SOA, possui os cenários que corretamente satisfazem as necessidades do negócio definidas em um modelo público de análise representado por uma *public* WF-net, são resumidamente apresentadas:

1. Identificar todos os cenários da *public* WF-net por meio do cálculo dos componentes repetitivos estacionários. Para isso, a *public* WF-net deve ser transformada em uma rede de Petri cíclica adicionando uma transição que liga o lugar final ao lugar inicial da *public* WF-net.

2. Representar os cenários da *public* WF-net, identificados na etapa 1, em sequentes da Lógica Linear e prová-los por meio da construção das árvores de prova.

3. Para as árvores de prova corretamente finalizadas na etapa 2, correspondentes aos cenários da *public* WF-net, construir os respectivos grafos de precedência rotulando as árvores de prova com as transições disparadas.

4. Identificar todos os cenários da U(IOWF-net), por meio do cálculo dos componentes repetitivos estacionários. Para isso, a U(IOWF-net) deve ser transformada em uma rede de Petri cíclica adicionando uma transição que liga o lugar final ao lugar inicial da U(IOWF-net).

5. Analisar cada componente repetitivo estacionário da U(IOWF-net), identificados na etapa 4, verificando se estes contém todas as transições de um dos componentes repetitivos estacionários da *public* WF-net que foram identificados na etapa 1 (lembrando que cada componente repetitivo estacionário representa um cenário). Caso isso aconteça, significa que o cenário correspondente ao componente repetitivo estacionário analisado é um candidato à equivalência por bissimulação de um dos cenários do modelo público. Caso contrário, o cenário analisado pode ser desconsiderado e, consequentemente, não será necessária a construção da correspondente árvore de prova e do grafo de precedência.

6. Representar os cenários candidatos da U(IOWF-net), identificados na etapa 5, em sequentes da Lógica Linear e prová-los por meio da construção das árvores de prova.

7. Para as árvores de prova corretamente finalizadas na etapa 6, correspondentes aos cenários da U(IOWF-net), construir os respectivos grafos de precedência rotulando as árvores de prova com as transições disparadas.

8. Reduzir os grafos de precedência correspondentes a U(IOWF-net), identificados na etapa 7, retirando dos grafos as ações silenciosas.

9. Verificar a equivalência entre os grafos de precedência referentes a *public* WF-net (encontrados na etapa 3) e os grafos de precedência referentes a U(IOWF-net) (encontrados na etapa 8) utilizando noção de bissimulação *branching* definida para este trabalho.

Para ilustrar o método proposto, os exemplos apresentados nas Figuras 17, 18 e 19 serão utilizados. A Figura 17 representa uma *public* WF-net (modelo de requisitos), a Figura 18 representa as *private* WF-nets que compõem a arquitetura, e a Figura 19 representa uma U(IOWF-net) que é a arquitetura final.

Esses exemplos envolvem dois parceiros de negócio: um contratante ($N^{priv}_{contractor}$) e um contratado ($N^{priv}_{subcontractor}$). No processo mostrado nos exemplos, primeiramente o contratante envia um pedido para o contratado e então este envia um orçamento para o contratante. Em seguida, o contratante confirma ou cancela o pedido, e baseada na resposta recebida, o contratado envia o pedido para o contratante ou encerra o processo.

O modelo público necessariamente será *sound*, mas o modelo privado poderá ser *relaxed sound*, ou seja, o modelo poderá conter *deadlock*, mas cada tarefa do processo deve ser considerada em pelo menos um dos cenários que podem ser corretamente finalizados.

Portanto, para a análise qualitativa dos modelos, é necessário identificar todos os cenários livres de *deadlock* na arquitetura proposta que satisfazem os cenários especificados no modelo de requisitos.

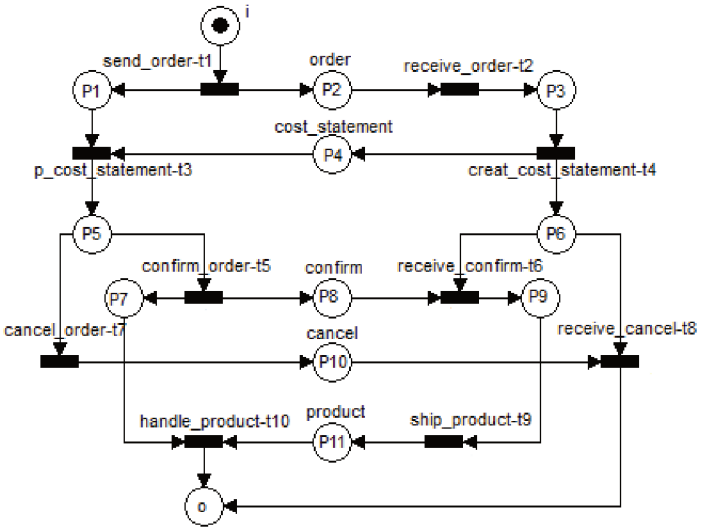


Figura 17 – *Public* WF-net (modelo de requisitos).

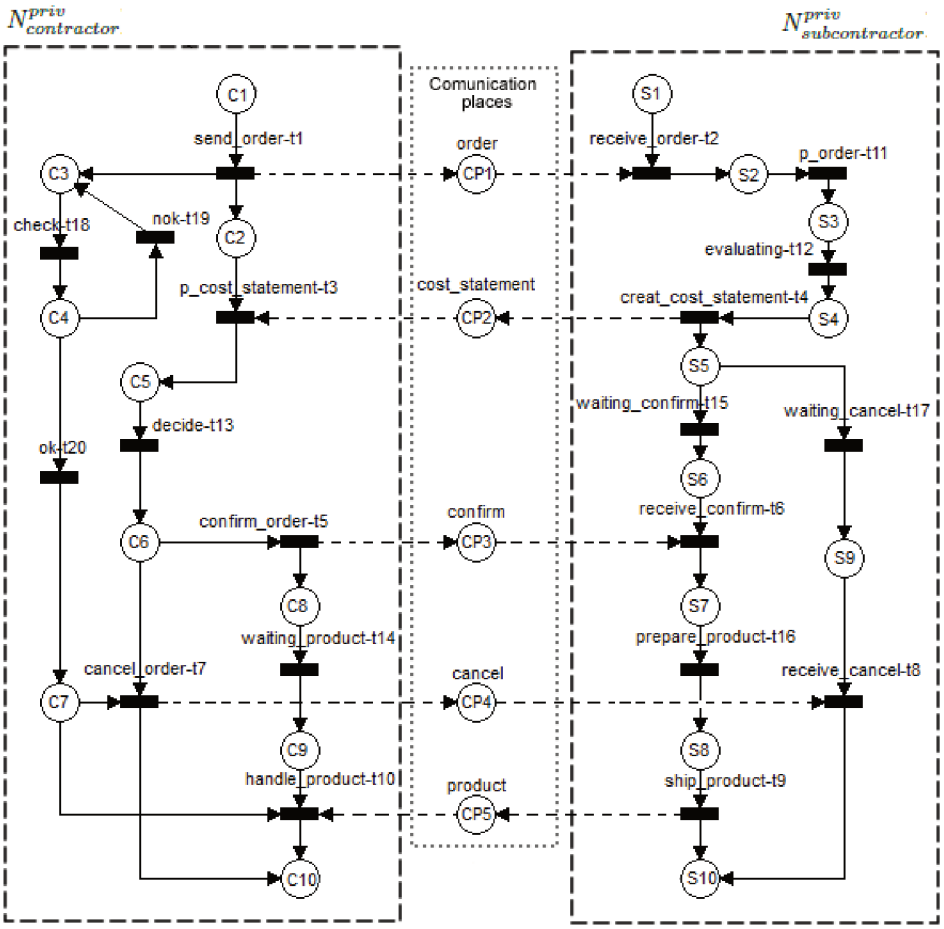


Figura 18 – *Private* WF-nets.

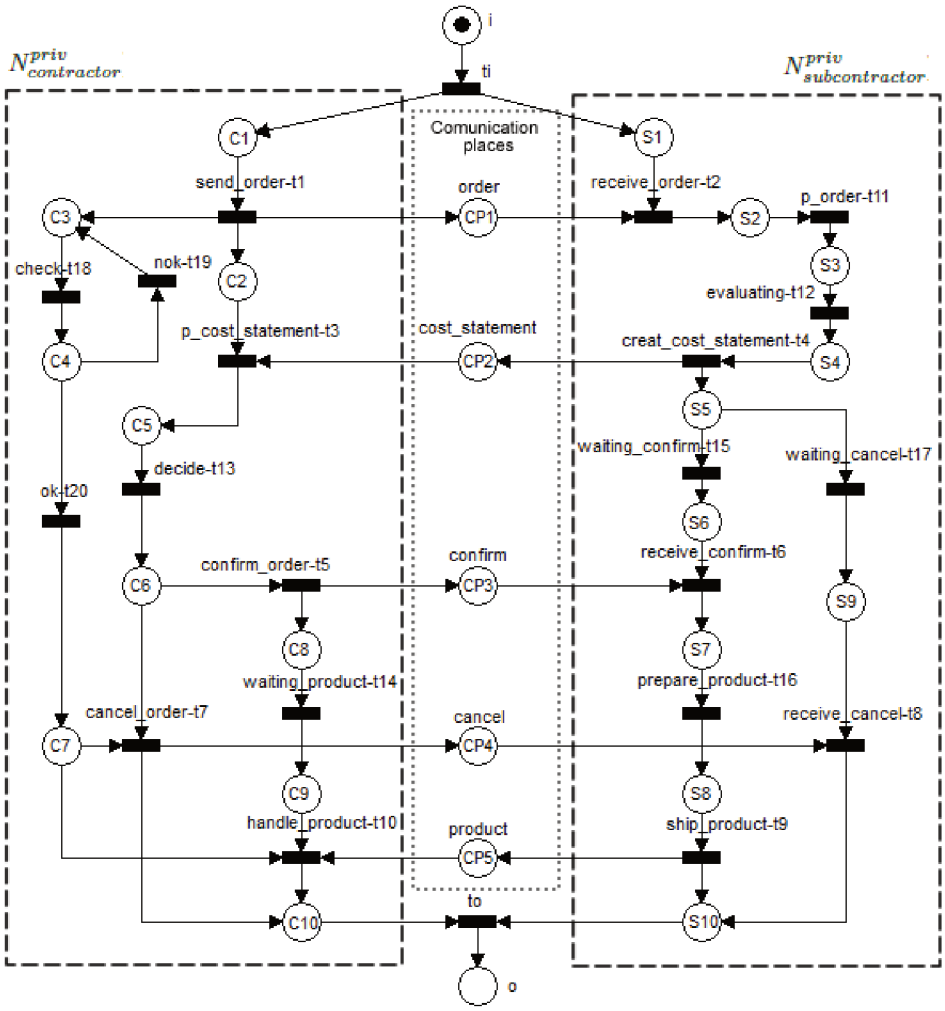


Figura 19 – U(IOWF-net) (arquitetura).

Nos modelos mostrados nas Figuras 17, 18 e 19, as transições correspondem as ações e os lugares correspondem as relações causais entre as ações. Os lugares "*order*", "*confirm*", "*cost_statement*", "*cancel*" e "*product*" são usados para a troca de mensagens entre as partes envolvidas no processo.

Neste trabalho de pesquisa, as rotas iterativas são substituídas por uma simples tarefa global, como é geralmente o caso de abordagens hierárquicas baseadas nas noções de blocos bem formados (VALETTE, 1979). A Figura 20 mostra um exemplo de restrições de rotas iterativas que existe na $N^{priv}_{conctractor}$ da Figura 19 e a correspondente transformação em uma única tarefa.

Como os modelos de requisitos e de arquitetura são analisados baseados na construção das árvores de prova da Lógica Linear, é necessário representar a *public* WF-net e U(IOWF-net) através das fórmulas da Lógica Linear. Desse modo, as transições (ações) da *public* WF-net apresentada na Figura 17 são representadas pelas seguintes fórmulas da Lógica Linear:

$send\text{-}order\text{-}t_1 = t_1 = i \multimap P_1 \otimes P_2$,

$receive\text{-}order\text{-}t_2 = t_2 = P_2 \multimap P_3$,

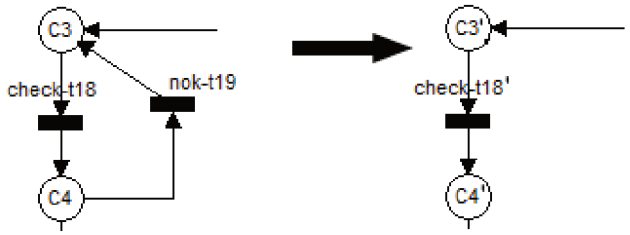


Figura 20 – Transformação de restrições de rotas iterativas em uma única tarefa.

$p\text{-}cost\text{-}statement\text{-}t_3 = t_3 = P_1 \otimes P_4 \multimap P_5$,
$creat\text{-}cost\text{-}statement\text{-}t_4 = t_4 = P_3 \multimap P_4 \otimes P_6$,
$confirm\text{-}order\text{-}t_5 = t_5 = P_5 \multimap P_7 \otimes P_8$,
$receive\text{-}confirm\text{-}t_6 = t_6 = P_6 \otimes P_8 \multimap P_9$,
$cancel\text{-}order\text{-}t_7 = t_7 = P_5 \multimap P_{10}$,
$receive\text{-}cancel\text{-}t_8 = t_8 = P_6 \otimes P_{10} \multimap o$,
$handle\text{-}product\text{-}t_9 = t_9 = P_9 \multimap P_{11}$,
$handle\text{-}product\text{-}t_{10} = t_{10} = P_7 \otimes P_{11} \multimap o$.

As transições da U(IOWF-net) apresentada na Figura 19 são representadas pelas seguintes fórmulas da Lógica Linear:
$t_i = i \multimap C_1 \otimes S_2$,
$send\text{-}order\text{-}t_1 = t_1 = C_1 \multimap C'_3 \otimes C_2 \otimes CP_1$,
$receive\text{-}order\text{-}t_2 = t_2 = S_1 \otimes CP_1 \multimap S_2$,
$p\text{-}cost\text{-}statement\text{-}t_3 = t_3 = C_2 \otimes CP_2 \multimap C_5$,
$p\text{-}order\text{-}t_{11} = t_{11} = S_2 \multimap S_3$,
$check\text{-}t'_{18} = t'_{18} = C'_3 \multimap C'_4$,
$evaluating\text{-}t_{12} = t_{12} = S_3 \multimap S_4$,
$creat\text{-}cost\text{-}statement\text{-}t_4 = t_4 = S_4 \multimap S_5 \otimes CP_2$,
$decide\text{-}t_{13} = t_{13}C_5 \multimap C_6$,
$ok\text{-}t_{20} = t_{20} = C'_4 \multimap C_7$,
$cancel\text{-}order\text{-}t_7 = t_7 = C_6 \otimes C_7 \multimap C_{10} \otimes CP_4$,
$receive\text{-}cancel\text{-}t_8 = t_8 = S_9 \otimes CP_4 \multimap S_{10}$,
$confirm\text{-}order\text{-}t_5 = t_5 = C_6 \multimap C_8 \otimes CP_3$,
$view\text{-}status\text{-}t_{15} = t_{15} = S_5 \multimap S_6$,
$receive\text{-}confirm\text{-}t_6 = t_6 = S_6 \otimes CP_3 \multimap S_7$,
$waiting\text{-}product\text{-}t_{14} = t_{14} = C_8 \multimap C_9$,
$prepare\text{-}product\text{-}t_{16} = t_{16} = S_7 \multimap S_8$,
$ship\text{-}product\text{-}t_9 = t_9 = S_8 \multimap CP_5 \otimes S_{10}$,
$handle\text{-}product\text{-}t_{10} = t_{10} = C_7 \otimes C_9 \otimes CP_5 \multimap C_{10}$,
$waiting\text{-}cancel\text{-}t_{17} = t_{17} = S_5 \multimap S_9$,
$t_o = C_{10} \otimes S_{10} \multimap o$.

Na etapa 1 do método, é necessário identificar todos os cenários da *public* WF-net (modelo de requisitos) por meio do cálculo dos componentes repetitivos estacionários. Aplicando a análise de invariantes da ferramenta PIPE, lembrando que para essa análise a *public* WF-net foi transformada em uma rede de Petri cíclica, os componentes repetitivos estacionários apresentados na Figura 21 foram encontrados.

T1	T10	T11	T2	T3	T4	T5	T6	T7	T8	T9
1	1	1	1	1	1	1	1	0	0	1
1	0	1	1	1	1	0	0	1	1	0

Figura 21 – Componentes repetitivos estacionários da *public* WF-net.

Os vetores apresentados na Figura 21 mostram dois componentes repetitivos estacionários que correspondem a uma lista não ordenada de transições que devem ser disparadas para voltar a rede a sua marcação inicial. Cada componente repetitivo estacionário encontrado para o modelo público será considerado como um cenário. Portanto, seguindo a etapa 2 do método, os cenários da *public* WF-net da Figura 17, chamados respectivamente de Sr1 e Sr2, são representados pelos seguintes sequentes da Lógica Linear:

- Sr1: $i, t_1, t_2, t_3, t_4, t_7, t_8 \vdash o$
- Sr2: $i, t_1, t_2, t_3, t_4, t_5, t_6, t_9, t_{10} \vdash o$

Após a representação dos cenários da *public* WF-net em sequentes da Lógica Linear, os mesmos devem ser provados utilizando as árvores de prova da Lógica Linear. A árvore de prova do cenário Sr1 é mostrada em sequência:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_6 \vdash P_6}\quad \dfrac{}{P_{10} \vdash P_{10}}}{P_6, P_{10} \vdash P_6 \otimes P_{10}}\otimes_R \quad \dfrac{}{o \vdash o}}{P_5 \vdash P_5 \quad P_6, P_{10}, P_6 \otimes P_{10} \multimap o \vdash o}\multimap_L}{\dfrac{P_1 \vdash P_1 \quad P_4 \vdash P_4}{P_1, P_4 \vdash P_1 \otimes P_4}\otimes_R \quad P_6, P_5, P_5 \multimap P_{10}, t_8 \vdash o}\multimap_L}{P_1, P_4, P_6, P_1 \otimes P_4, \multimap P_5, t_7, t_8 \vdash o}\otimes_L}{P_3 \vdash P_3 \quad P_1, P_4 \otimes P_6, P_1 \otimes P_4 \multimap P_5, t_7, t_8 \vdash o}\otimes_L}{P_1, P_3, P_1 \otimes P_4 \multimap P_5, P_3 \multimap P_4 \otimes P_6, t_7, t_8 \vdash o}}{P_2 \vdash P_2 \quad P_1, P_3, P_1 \otimes P_4 \multimap P_5, t_4, t_7, t_8 \vdash o}\multimap_L}{P_1, P_2, P_2 \multimap P_3, t_3, t_4, t_7, t_8 \vdash o}\otimes_L}{i \vdash i \quad P_1 \otimes P_2, t_2, t_3, t_4, t_7, t_8 \vdash o}\multimap_L}{i, i \multimap P_1 \otimes P_2, t_2, t_3, t_4, t_7, t_8 \vdash o}$$

Como pode ser observado na última linha (lembrando que a árvore de prova da Lógica Linear é lida de baixo para cima), a árvore de prova para o cenário Sr1 é corretamente

finalizada, pois o sequente identidade $o \vdash o$ ('o' corresponde ao lugar final) é produzido, não há transições para serem provadas e não há átomos para serem consumidos. Consequentemente, o cenário Sr1 pode ser corretamente executado.

As próximas árvores de provas serão resumidamente apresentadas (apenas o primeiro e o último sequente). As árvores de prova completas para todos os outros cenários que devem ser provados são apresentadas no Apêndice A.1.

A árvore de prova resumida do cenário Sr2 da *public* WF-net é mostrada em sequência:

$$\dfrac{\dfrac{\dfrac{\dfrac{P_7 \vdash P_7 \quad P_{11} \vdash P_{11}}{P_7, P_{11} \vdash P_7 \otimes P_{11}} \otimes_R \quad o \vdash o}{} \multimap_L}{\vdots}}{i, i \multimap P_1 \otimes P_2, t_2, t_3, t_4, t_5, t_6, t_9, t_{10} \vdash o}$$

Como pode ser observado na última linha, a árvore de prova do cenário Sr2 é corretamente finalizada. Consequentemente, o cenário Sr2 também pode ser corretamente executado.

Na etapa 3 do método, deve-se construir os grafos de precedência para as árvores de prova correspondentes aos cenários da *public* WF-net corretamente finalizadas. Para gerar o grafo de precedência, as árvores de prova dos respectivos cenários devem ser rotuladas com o disparo das transições como explicado na seção 2.5. Para o cenário Sr1, a árvore de prova rotulada com as transições de disparo é mostrada em sequência:

$$\dfrac{\overset{i_1}{i}\vdash\overset{t_1}{i} \quad \dfrac{\dfrac{\dfrac{\overset{t_1}{P_2}\vdash\overset{t_2}{P_2} \quad \dfrac{\dfrac{\overset{t_2}{P_3}\vdash\overset{t_4}{P_3} \quad \dfrac{\dfrac{\dfrac{\dfrac{\overset{t_1}{P_1}\vdash\overset{t_3}{P_1} \quad \overset{t_4}{P_4}\vdash\overset{t_3}{P_4}}{\overset{t_1}{P_1},\overset{t_4}{P_4}\vdash\overset{t_3}{P_1}\otimes\overset{t_3}{P_4}} \otimes_R \quad \dfrac{\overset{t_3}{P_5}\vdash\overset{t_7}{P_5} \quad \dfrac{\dfrac{\dfrac{\overset{t_4}{P_6}\vdash\overset{t_8}{P_6} \quad \overset{t_7}{P_{10}}\vdash\overset{t_8}{P_{10}}}{\overset{t_4}{P_6},\overset{t_7}{P_{10}}\vdash\overset{t_8}{P_6}\otimes\overset{t_8}{P_{10}}} \otimes_R \quad \overset{t_8}{o}\vdash\overset{f_1}{o}}{\overset{t_4}{P_6},\overset{t_7}{P_{10}},P_6\otimes P_{10}\multimap o\vdash\overset{f_1}{o}} \multimap_{L\,t_8}}{}}{\overset{t_4}{P_6},\overset{t_3}{P_5},P_5\multimap P_{10},t_8\vdash\overset{f_1}{o}} \multimap_{L\,t_7}}{\overset{t_1}{P_1},\overset{t_4}{P_4},\overset{t_4}{P_6},P_1\otimes P_4,\multimap P_5,t_7,t_8\vdash\overset{f_1}{o}} \multimap_{L\,t_3}}{\overset{t_1}{P_1},\overset{t_4}{P_4}\otimes\overset{t_4}{P_6},P_1\otimes P_4\multimap P_5,t_7,t_8\vdash\overset{f_1}{o}} \otimes_L}{}}{\overset{t_1}{P_1},\overset{t_2}{P_3},P_1\otimes P_4\multimap P_5,P_3\multimap P_4\otimes P_6,t_7,t_8\vdash\overset{f_1}{o}} \multimap_{L\,t_4}}{\overset{t_1}{P_1},\overset{t_2}{P_3},P_1\otimes P_4\multimap P_5,t_4,t_7,t_8\vdash\overset{f_1}{o}}}{\overset{t_1}{P_1},\overset{t_1}{P_2},P_2\multimap P_3,t_3,t_4,t_7,t_8\vdash\overset{f_1}{o}} \multimap_{L\,t_2}}{\overset{t_1}{P_1}\otimes\overset{t_1}{P_2},t_2,t_3,t_4,t_7,t_8\vdash\overset{f_1}{o}} \otimes_L}{\overset{i_1}{i},i\multimap P_1\otimes P_2,t_2,t_3,t_4,t_7,t_8\vdash\overset{f_1}{o}} \multimap_{L\,t_1}$$

O grafo de precedência correspondente ao cenário Sr1 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 22.

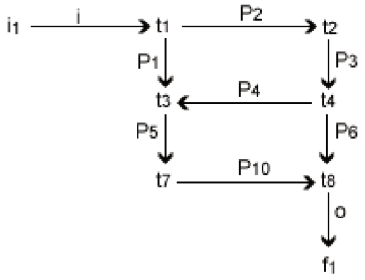


Figura 22 – Grafo de precedência do cenário Sr1.

Para o cenário Sr2, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência:

$$\dfrac{\dfrac{}{P_7 \vdash P_7}^{t_5\ t_{10}} \quad \dfrac{}{P_{11} \vdash P_{11}}^{t_9\ t_{10}}}{P_7, P_{11} \vdash P_7 \otimes P_{11}}^{t_5\ t_9\ t_{10}\ t_{10}} \otimes R \qquad \dfrac{}{o \vdash o}^{t_{10}\ f_1} \multimap L$$

$$\vdots$$

$$i \vdash i \multimap P_1 \otimes P_2, t_2, t_3, t_4, t_5, t_6, t_9, t_{10} \vdash o$$

O grafo de precedência correspondente ao cenário Sr2 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 23.

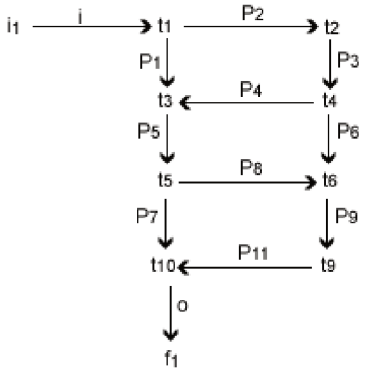


Figura 23 – Grafo de precedência do cenário Sr2.

Na etapa 4 do método, todos os cenários da U(IOWF-net) (arquitetura) são identificados por meio do cálculo dos componentes repetitivos estacionários. Aplicando a análise de invariantes da ferramenta PIPE para a U(IOWF-net), lembrando que para essa análise a U(IOWF-net) também foi transformada em uma rede de Petri cíclica, os componentes repetitivos estacionários apresentados na Figura 24 foram encontrados.

T1	T10	T11	T12	T13	T14	T16	T17	T18'	T2	T20	T21	T3	T4	T5	T6	T7	T8	T9	Ti	To	T15
1	0	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	0	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1

Figura 24 – Componentes repetitivos estacionários da U(IOWF-net).

Na etapa 5 do método, cada componente repetitivo estacionário encontrado para a U(IOWF-net) deve ser analisado verificando se contém todas as transições de um dos componentes repetitivos estacionários do modelo de requisitos que foram identificados na etapa 1. Para os dois componentes repetitivos estacionários da U(IOWF-net) essa verificação se confirma, ou seja, o primeiro vetor da Figura 24 possui todas as transições do segundo vetor da Figura 21 e o segundo vetor da Figura 24 possui todas as transições do primeiro vetor da Figura 21. Portanto, os cenário correspondentes aos componentes repetitivos estacionários analisados são potenciais candidatos à equivalência por bissimulação de um dos cenários do modelo público.

Os potenciais cenários da U(IOWF-net), encontrados na etapa 5, são chamados respectivamente de Sa1 e Sa2, e, seguindo a etapa 6 do método, são representados pelos seguintes sequentes da Lógica Linear:

- cenário Sa1: $i, t_1, t_{11}, t_{12}, t_{13}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_7, t_8, t_i, t_o \vdash o$
- cenário Sa2: $i, t_1, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{16}, t'_{18}, t_2, t_{20}, t_3, t_4, t_5, t_6, t_9, t_i, t_o, t_{15} \vdash o$

A árvore de prova resumida do cenário Sa1 é a seguinte:

$$\dfrac{\dfrac{\dfrac{C_{10}\vdash C_{10} \quad S_{10}\vdash S_{10}}{C_{10},S_{10}\vdash C_{10}\otimes S_{10}}\otimes_R \quad o\vdash o}{\quad}\multimap_L}{\vdots}$$
$$i,i\multimap C_1\otimes S_1,t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o$$

A árvore de prova resumida do cenário Sa2 é a seguinte:

$$\dfrac{\dfrac{\dfrac{S_{10}\vdash S_{10} \quad C_{10}\vdash C_{10}}{S_{10},C_{10}\vdash C_{10}\otimes S_{10}}\otimes_R \quad o\vdash o}{\quad}\multimap_L}{\vdots}$$
$$i,i\multimap C_1\otimes S_1,t_1,t_{10},t_{11},t12,t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o$$

Como pode ser observado na última linha, as árvores de prova dos cenários Sa1 e Sa2 são corretamente finalizadas e, consequentemente, os cenários Sa1 e Sa2 também podem ser corretamente executados.

Na etapa 7 do método, deve-se construir os grafos de precedência para as árvores de prova correspondentes aos cenários da U(IOWF-net) corretamente finalizadas. Para o cenário Sa1, a árvore de prova resumida e rotulada com as transições de disparo é a seguinte:

$$\dfrac{\dfrac{\dfrac{}{C_{10}\vdash C_{10}}{\scriptstyle t_7,t_o}\quad\dfrac{}{S_{10}\vdash S_{10}}{\scriptstyle t_8,t_o}}{C_{10},S_{10}\vdash C_{10}\otimes S_{10}}\otimes_R\quad\dfrac{}{o\vdash o}{\scriptstyle t_o,f_1}}{\vdots}\multimap_L{\scriptstyle t_{17}}$$

$$i,i\multimap C_1\otimes S_1,t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o \quad {\scriptstyle i_1,f_1}$$

O grafo de precedência correspondente ao cenário Sa1 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 25.

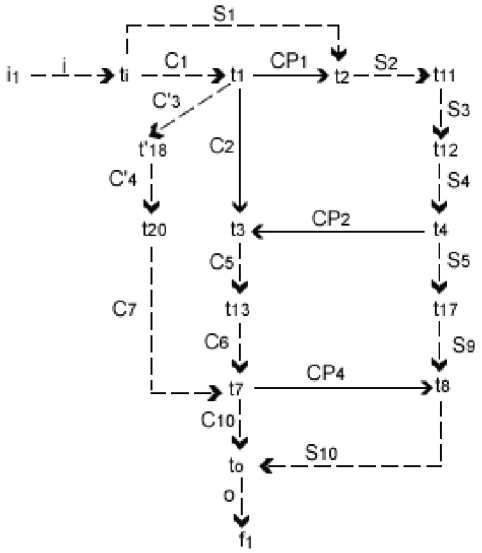


Figura 25 – Grafo de precedência do cenário Sa1.

A árvore de prova resumida e rotulada com as transições de disparo do cenário Sa2 é a seguinte:

$$\dfrac{\dfrac{\dfrac{}{S_{10}\vdash S_{10}}{\scriptstyle t_9,t_o}\quad\dfrac{}{C_{10}\vdash C_{10}}{\scriptstyle t_{10},t_o}}{S_{10},C_{10}\vdash C_{10}\otimes S_{10}}\otimes_R\quad\dfrac{}{o\vdash o}{\scriptstyle t_o,f_1}}{\vdots}\multimap_L{\scriptstyle t_o}$$

$$i,i\multimap C_1\otimes S_1,t_1,t_{10},t_{11},t12,t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o \quad {\scriptstyle i_1,f_1}$$

O grafo de precedência correspondente ao cenário Sa2 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 26.

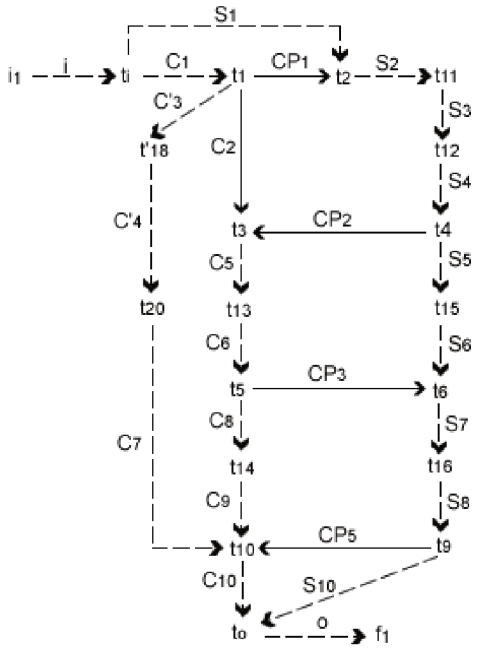


Figura 26 – Grafo de precedência do cenário Sa2.

Na etapa 8 do método, os grafos de precedência correspondentes a U(IOWF-net), identificados na etapa 7, são simplificados retirando do grafo as ações silenciosas. Ao remover todas as ações silenciosas dos grafos de precedência dos cenários Sa1 e Sa2 mostrados respectivamente nas Figuras 25 e 26 (lembrando que os arcos que estão ligados as ações silenciosas são representados no grafo de precedência por linhas tracejadas), os grafos de precedência simplificados das Figuras 27 e 28 são obtidos.

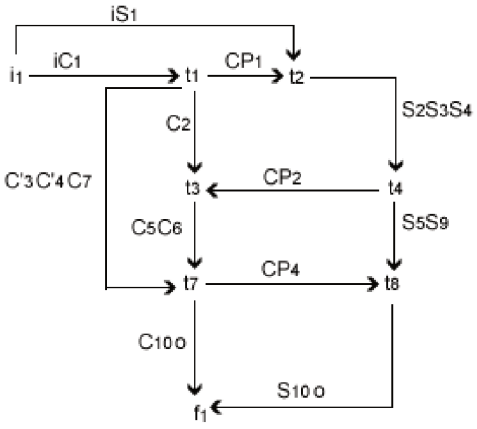


Figura 27 – Grafo de precedência simplificado do cenário Sa1.

Na etapa 9 do método, deve-se comparar os grafos de precedência do modelo público com os grafos de precedência associados ao modelo privado. A razão para isso é verificar se os requisitos de serviço especificados na *public* WF-net estão também presentes na proposta de arquitetura especificada na U(IOWF-net). Portanto, os grafos de precedência obtidos a partir dos cenários Sr1 e Sr2 devem então ser comparados com os grafos

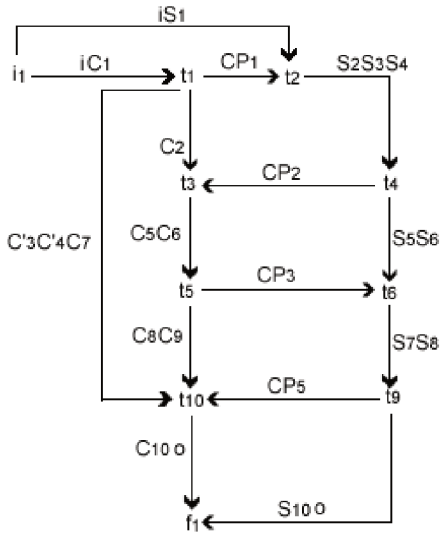


Figura 28 – Grafo de precedência simplificado do cenário Sa2.

de precedência obtidos a partir dos cenários Sa1 e Sa2 usando a semântica operacional baseada na noção de bissimulação *branching*.

A partir das comparações entre os grafos de precedência, foi encontrado que o grafo de precedência do cenário Sr1 (Figura 22) e o grafo de precedência simplificado do cenário Sa1 (Figura 27) executam a mesma sequência de ações respeitando a mesma sequência de restrições. Para facilitar a visualização, a Figura 29 mostra esses dois grafos lado a lado. Os arcos adicionais iS_1, $C'_3C'_4C_7$ e $C_{10}o$ que existem no grafo de precedência simplificado do cenário Sa1 (Figura 29 (b)), e que não existem no grafo de precedência do cenário Sr1 (Figura 29 (a)) são simplesmente restrições redundantes, as quais podem ser desconsideradas sem modificar as especificações dos requisitos. Por exemplo, o arco $C'_3C'_4C_7$ da Figura 29 (b) simplesmente afirma que a ação t_1 deve acontecer antes da ação t_7. No entanto, essa afirmação já existe através da sequência de arcos C_2, C_5C_6, por exemplo. Desconsiderando os arcos redundantes do grafo de precedência simplificado do cenário Sa1, os grafos de precedência das Figuras 29 (a) e 29 (b) são equivalentes respeitando a noção de bissimulação *branching* definida neste estudo.

O grafo de precedência do cenário Sr2 (Figura 23) e o grafo de precedência simplificado do cenário Sa2 (Figura 28) executam a mesma sequência de ações respeitando a mesma sequência de restrições. A Figura 30 mostra esses dois grafos lado a lado. Desconsiderando os arcos redundantes (iS_1, $C'_3C'_4C_7$ e $S_{10}o$) do grafo de precedência simplificado do cenário Sa2, os grafos de precedência da Figura 30 (a) e 30 (b) são equivalente respeitando a noção de bissimulação *branching* definida neste estudo.

Na U(IOWF-net), existem outros dois cenários, chamados de Sa3 e Sa4, que não foram identificado pelos componentes repetitivos estacionários na etapa 4 do método proposto, pois estes cenários levam o modelo a um estado de *deadlock*. Para mostrar formalmente que estes cenários não são finalizados corretamente, as árvores de prova correspondentes

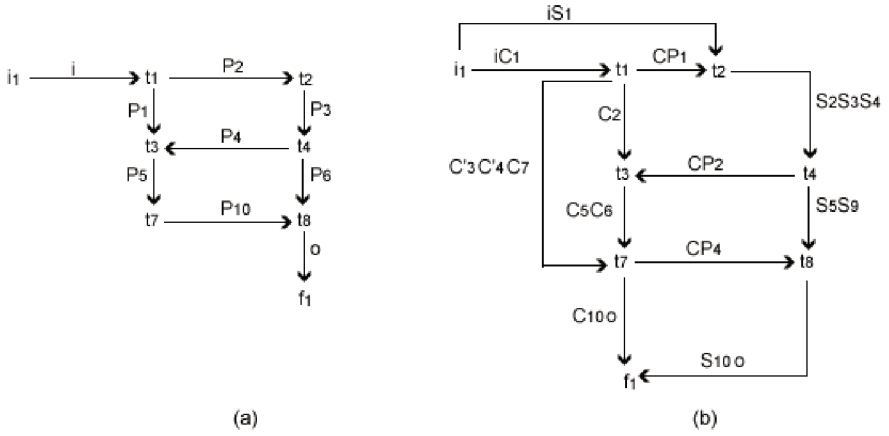


Figura 29 – (a) Grafo de precedência do cenário Sr1 e (b) Grafo de precedência do cenário Sa1.

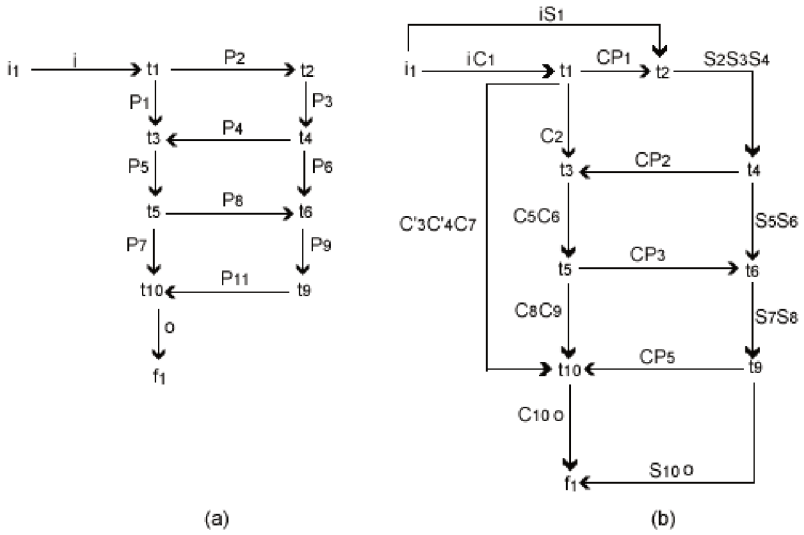


Figura 30 – (a) Grafo de precedência do cenário Sr2 e (b) Grafo de precedência simplificado do cenário Sa2.

foram construídas. Para isso, considere que o cenário Sa3 e Sa4 são representados pelos seguintes sequentes da Lógica Linear:

- Sa3: $i, t_i, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o$
- Sa4: $i, t_i, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_7, t_{15}, t_6, t_{16}, t_9, t_o \vdash o$

Em seguida é apresentada a árvore de prova resumida para o cenário Sa3, a árvore de prova completa é mostrada no Apêndice A.1.

$$\dfrac{\dfrac{}{C_8 \vdash C_8} \quad \dfrac{}{C_7, CP_3, S_9, C_9, t_{10}, t_8, t_o \vdash o}}{\vdots}$$

$$i, i \multimap C_1 \otimes S_1, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o$$

Em seguida é apresentada a árvore de prova resumida para o cenário Sa4, a árvore de prova completa é mostrada no Apêndice A.1.

$$\dfrac{\dfrac{}{S_5 \vdash S_5} \quad \dfrac{}{CP_4, C_{10}, S_6, t_6, t_{16}, t_9, t_o \vdash o}}{\vdots}$$
$$\overline{i, i \multimap C_1 \otimes S_1, t_i, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_7, t_{15}, t_6, t_{16}, t_9, t_o \vdash o}$$

Como pode ser observado na última linha das árvores de prova, os cenários Sa3 e Sa4 não são corretamente executados. No final da prova do cenário Sa3 observa-se que há quatro transições que não foram disparadas e que existem fichas presentes em C_7, C_9, CP_3 e S_9. Já na última linha da árvore de prova do cenário Sa4 observa-se que há três transições que não foram disparadas e que existem fichas presentes em C_{10}, CP_4 e S_6. Isso significa que os cenários Sa3 e Sa4 não correspondem a um comportamento *sound*, por isso, não foram identificados pelos componentes repetitivos estacionários. Consequentemente, estes cenários não serão considerados como parte dos cenários necessários para cobrir os requisitos de negócio do modelo público.

Embora o modelo de SOA, representado por uma U(IOWF-net) (Figura 19), não seja livre de situações de *deadlock* (cenários Sa3 e Sa4), os cenários Sa1 e Sa2 são corretamente executados e verificam os requisitos especificados no modelo público representado por uma *public* WF-net (Figura 17). Portanto, mesmo que um modelo de SOA não seja livre de *deadlock* ou possua requisitos de serviços adicionais mas que não são do interesse do modelo público de contrato, é possível verificar se este modelo possui os cenários que corretamente satisfazem as necessidades do negócio definidas no modelo de análise.

Para o exemplo mostrado nessa seção, pode-se concluir que os requisitos definidos no modelo de análise estão bem definidos na arquitetura. Em particular, os requisitos presentes nos cenários Sr1 e Sr2 do modelo de análise estão presentes, respectivamente, nos cenários Sa1 e Sa2 da arquitetura.

3.3 Verificação do desempenho dos requisitos de serviços em SOA

Os requisitos não funcionais de desempenho especificam características para avaliar a velocidade ou a eficácia operacional de um recurso que deve ser entregue por uma arquitetura de sistema. Existem várias classes de requisitos de desempenho sendo que o tempo de resposta é um dos mais conhecidos. Neste trabalho, o requisito não funcional de desempenho é verificado em relação ao intervalo de data que finaliza a execução de um cenário de requisito de serviço.

As datas de execução das tarefas tanto do modelo de requisitos quanto de arquitetura serão dadas através de intervalos de datas simbólicas as quais são obtidas durante a construção das árvores de prova da Lógica Linear. Posteriormente, os intervalos de datas simbólicas produzidos no final da execução de cada cenário são substituídos por valores numéricos. Desse modo, os cenários da arquitetura que satisfazem o comportamento dos cenários especificados no modelo de requisitos serão equivalentes em termos de desempenho se o intervalo de data numérica que finaliza um cenário da arquitetura pertence ao intervalo de data numérica que finaliza o cenário equivalente do modelo de requisitos. A principal vantagem de empregar datas simbólicas é que, quando já calculadas, podem ser utilizadas diretamente para qualquer instância dos modelos de requisitos e de arquitetura.

Durante a construção das árvores de prova da Lógica Linear, as datas simbólicas de produção e de consumo são obtidas para todos os átomos. Desse modo, para qualquer atividade de um cenário ou partes intermediárias de um processo, é possível realizar análises em relação aos tempos de produção e de consumo. No entanto, neste trabalho, apenas a data de produção do átomo que representa a finalização de um cenário será analisada, pois a verificação de equivalência será do cenário como um todo.

Definição 3.3.1 *Um cenário da U(IOWF-net) que possui comportamento equivalente a um cenário da public WF-net de acordo com a noção de bissimulação* branching *definida para teste estudo, será também equivalente no desempenho se* $[D_{PminCAo}, D_{PmaxCAo}] \in [D_{PminCRo}, D_{PmaxCRo}]$*, tal que:*

- $[D_{PminCAo}, D_{PmaxCAo}]$ *representa o intervalo de data simbólica de produção do átomo que finaliza o cenário da U(IOWF-net);*
- $[D_{PminCRo}, D_{PmaxCRo}]$ *representa o intervalo de data simbólica de produção do átomo que finaliza o cenário correspondente da public WF-net.*

Para a verificação do desempenho dos requisitos de serviços em SOA, devem ser considerados todos os cenários da arquitetura que satisfazem o comportamento dos cenários especificados no modelo de requisitos. Tais cenários são obtidos através da aplicação do método para a verificação dos requisitos de serviços em SOA apresentado na seção 3.2. Portanto, a primeira etapa do método proposto para a verificação do desempenho dos requisitos de serviços em SOA é identificar os cenários da U(IOWF-net) que satisfazem o comportamento dos cenários especificados na *public* WF-net.

Na segunda etapa do método, para cada cenário identificado na primeira etapa, deve-se calcular as datas simbólicas de produção e de consumo através da correspondente árvore de prova da Lógica Linear, conforme explicado na seção 2.5. Como as árvores de prova já foram construídas para a identificação dos cenários que são equivalentes no comportamento (primeira etapa), elas podem ser reutilizadas para o cálculo das datas simbólicas. A

partir das datas simbólicas, os respectivos intervalos de datas simbólicas também são obtidos. É importante ressaltar que, apesar do cálculo de dadas simbólicas ser realizado para todos os átomos dos cenários, somente as datas de produção dos átomos que finalizam os cenários são consideradas para a verificação do desempenho. Consequentemente, somente os intervalos de datas simbólicas de produção dos átomos que finalizam os cenários são considerados.

Na terceira etapa do método, deve-se calcular o intervalo de data numérica do átomo que finaliza a execução de cada cenário. Para isso, os valores numéricos definidos na *public* WF-net e na U(IOWF-net) são substituídos nos intervalos de datas simbólicas de produção dos átomos que finalizam a execução dos cenários, calculados na segunda etapa.

Na quarta etapa do método, deve-se comparar os intervalos de datas numéricas calculados na etapa anterior. Se o intervalo de data numérica que finaliza um cenário da U(IOWF-net) pertencer ao intervalo de data numérica que finaliza o cenário correspondente da *public* WF-net, então estes cenários serão também equivalentes no desempenho.

As etapas do método para formalmente verificar o desempenho dos requisitos de serviços em SOA, representadas por U(IOWF-net), resumidamente são:

1. identificar os cenários da U(IOWF-net) que satisfazem o comportamento dos cenários especificados na *public* WF-net;

2. para cada cenário identificado na etapa 1, calcular as datas simbólicas de produção e de consumo através da correspondente árvore de prova da Lógica Linear e também os respectivos intervalos de datas simbólicas;

3. calcular o intervalo de data numérica do átomo que finaliza a execução de cada cenário;

4. comparar os intervalos de datas numéricas, calculados na etapa 3, dos cenários da *public* WF-net com os intervalos de datas numéricas da U(IOWF-net).

Para a análise de desempenho, serão consideradas explícitas restrições de tempo nos modelos de requisitos e de arquitetura; por isso, será associada a cada transição $t \in T$ da *public* WF-net e da U(IOWF-net) um intervalo $[\theta_{min(t)}, \theta_{max(t)}]$ que corresponde a uma duração de sensibilização da transição (MERLIN, 1974). Assim, o intervalo [2, 4], por exemplo, indica que a transição irá disparar pelo menos duas unidades de tempo após a transição ter sido sensibilizada e no máximo quatro unidades de tempo após a sensibilização desta transição. Portanto, para a verificação do requisito não funcional de desempenho, a *public* WF-net e a U(IOWF-net) serão modelos temporizados.

O método proposto para a verificação do desempenho dos requisitos de serviços em SOA é também ilustrado usando os exemplos apresentados na seção 3.2. No entanto, como pode ser observado nas Figuras 31, 32 e 33, os modelos são temporizados.

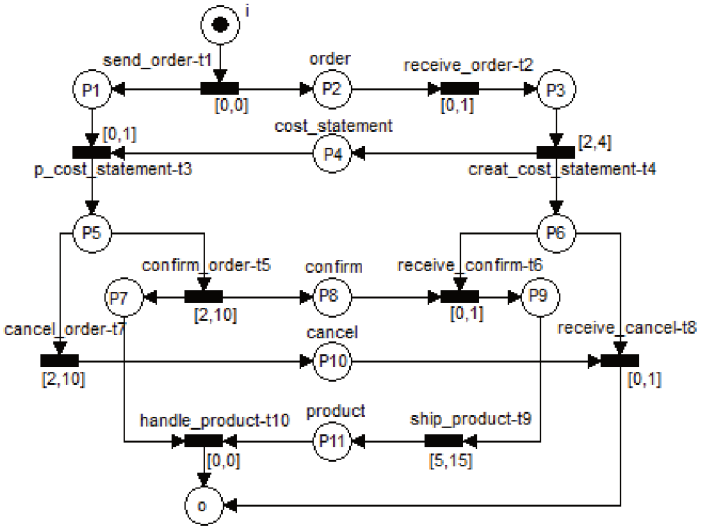


Figura 31 – *Public* WF-net (modelo de requisitos) temporizada.

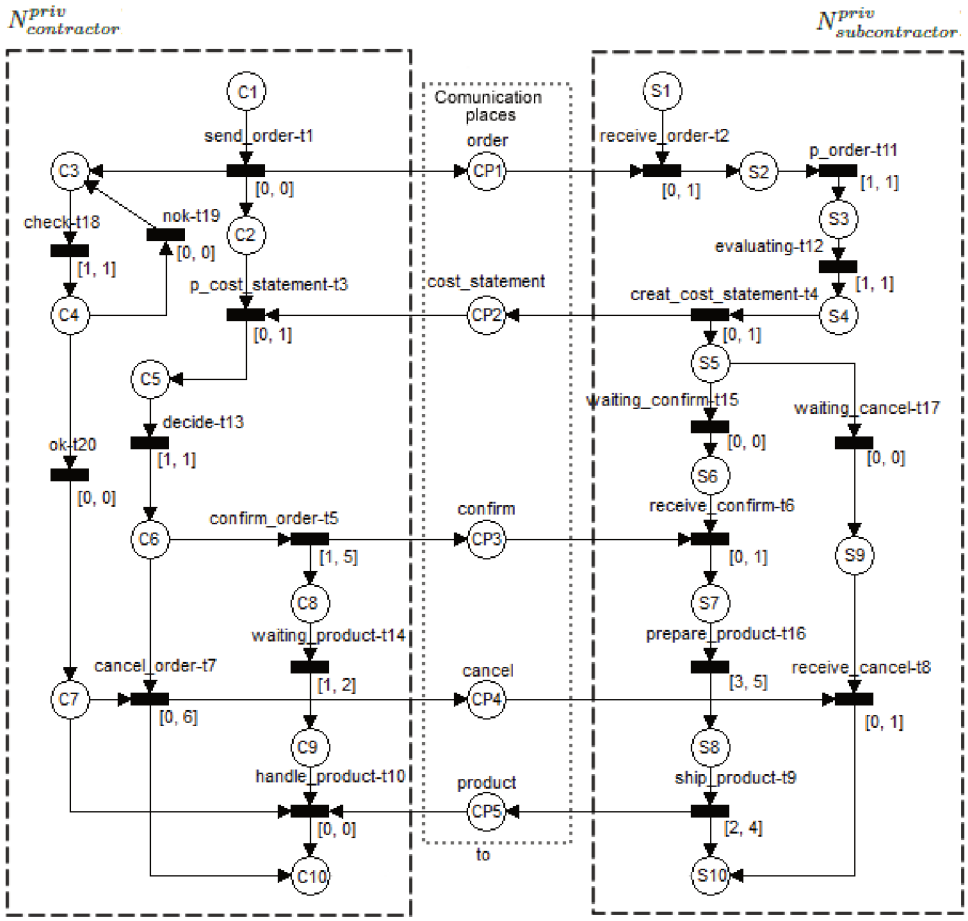


Figura 32 – *Private* WF-nets temporizadas.

Na etapa 1 do método, é necessário identificar os cenários da arquitetura (U(IOWF-net)) que satisfazem o comportamento dos cenários especificados no modelo de requisitos

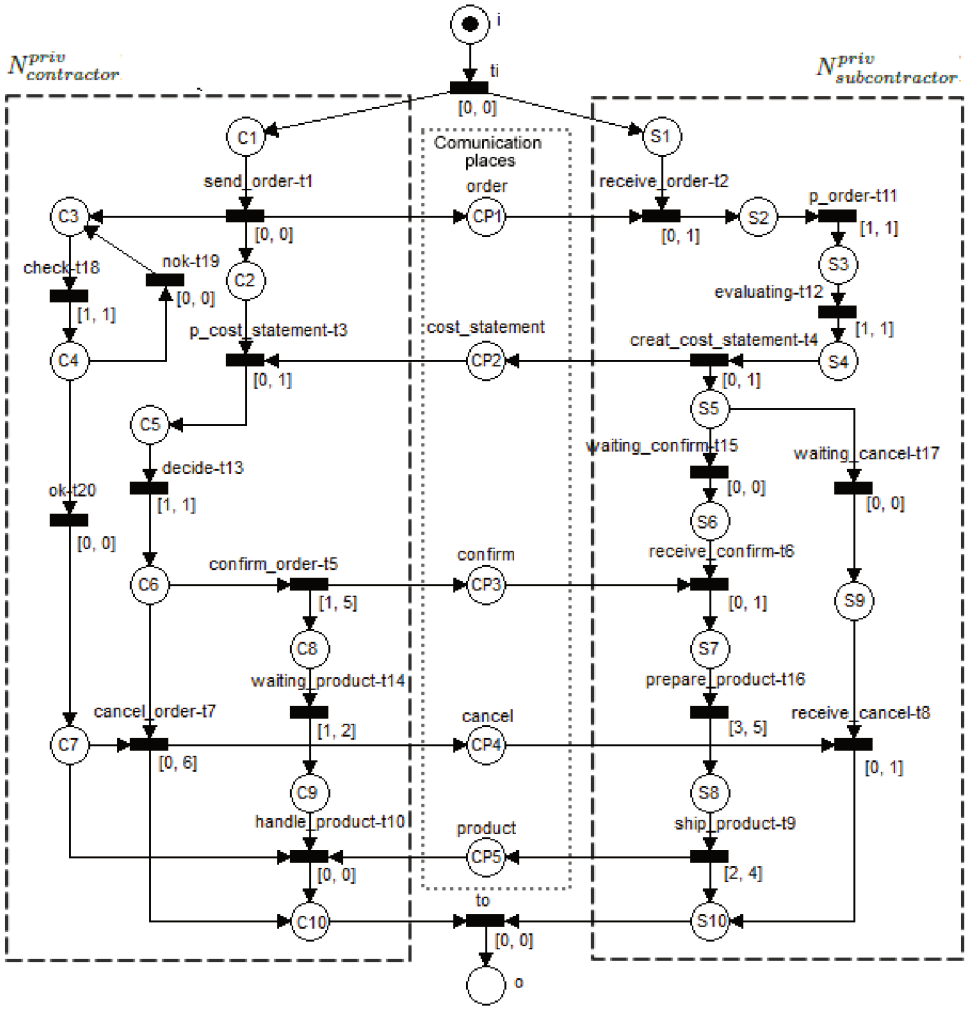


Figura 33 – U(IOWF-net) (arquitetura) temporizada.

(*public* WF-net). Seguindo o método apresentado na seção 3.2, conclui-se que o cenário Sa1 da arquitetura produz um comportamento equivalente, respeitando a noção de bissimulação *branching* definida para este estudo, com o cenário Sr1 do modelo de requisitos, e o cenário Sa2 da arquitetura produz um comportamento equivalente com o cenário Sr2 do modelo de requisitos. O processo completo para identificar tais cenários equivalentes é também mostrado na seção 3.2, uma vez que os exemplos usados para demonstrar os métodos propostos são os mesmos.

Na etapa 2 do método, é necessário calcular para cada cenário identificado na etapa 1 (Sa1, Sr1, Sa2 e Sr2) as datas simbólicas de produção e de consumo. Em seguida, obter os intervalos de datas simbólicas de produção dos átomos que finalizam a execução dos cenários. Para simplificar a demonstração do método proposto, o cálculo das datas simbólicas na árvore de prova é demonstrado apenas para o cenário Sr1, para os cenários Sr2, Sa1 e Sa2 as árvores de prova com datas simbólicas são mostradas no apêndice A.2.

Considerando $Seq_1 = D_i + d_1 + d_2 + d_4$, a árvore de prova correspondente com datas simbólicas para o cenário Sr1 é a seguinte:

Para melhor visualização, as datas simbólicas de produção e de consumo do cenário

$$\dfrac{\dfrac{P_6(Seq_1,Seq_1+d_3+d_7+d_8)\vdash P_6 \quad P_{10}(Seq_1+d_3+d_7,Seq_1+d_3+d_7+d_8)\vdash P_{10}}{P_6(Seq_1,Seq_1+d_3+d_7+d_8),P_{10}(Seq_1+d_3+d_7,Seq_1+d_3+d_7+d_8)\vdash P_6\otimes P_{10}}\otimes_R \quad o(Seq_1+d_3+d_7+d_8,.)\vdash o}{\dfrac{P_5(Seq_1+d_3,Seq_1+d_3+d_7)\vdash P_5 \quad P_6(Seq_1,.),P_{10}(Seq_1+d_3+d_7,.),P_6\otimes P_{10}\multimap o\vdash o}{\dfrac{\dfrac{P_1(D_i+d_1,Seq_1+d_3)\vdash P_1 \quad P_4(Seq_1,Seq_1+d_3)\vdash P_4}{P_1(D_i+d_1,Seq_1+d_3),P_4(Seq_1,Seq_1+d_3)\vdash P_1\otimes P_4}\otimes_R \quad P_6(Seq1,.),P_5(Seq_1+d_3,.),P_5\multimap P_{10},t_8\vdash o}{\dfrac{P_1(D_i+d_1,.),P_4(Seq_1,.),P_6(Seq_1,.),P_1\otimes P_4\multimap P_5,t_7,t_8\vdash o}{\dfrac{P_3(D_i+d_1+d_2,Seq_1)\vdash P_3 \quad P_1(D_i+d_1,.),P_4(Seq_1,.)\otimes P_6(Seq_1,.),P_1\otimes P_4\multimap P_5,t_7,t_8\vdash o}{\dfrac{P_1(D_i+d_1,.),P_3(D_i+d_1+d_2,.),P_1\otimes P_4\multimap P_5,P_3\multimap P_4\otimes P_6,t_7,t_8\vdash o}{\dfrac{P_2(D_i+d_1,D_i+d_1+d_2)\vdash P_2 \quad P_1(D_i+d_1,.),P_3(D_i+d_1+d_2,.),P_1\otimes P_4\multimap P_5,t_4,t_7,t_8\vdash o}{\dfrac{P_1(D_i+d_1,.),P_2(D_i+d_1,.),P_2\multimap P_3,t_3,t_4,t_7,t_8\vdash o}{\dfrac{i(D_i,D_i+d_1)\vdash i \quad P_1(D_i+d_1,.)\otimes P_2(D_i+d_1,.),t_2,t_3,t_4,t_7,t_8\vdash o}{i(D_i,.),i\multimap P_1\otimes P_2,t_2,t_3,t_4,t_7,t_8\vdash o}\multimap_L}\otimes_L}\multimap_L}\otimes_L}\multimap_L}\otimes_L}\multimap_L}\otimes_L}\multimap_L}\multimap_L$$

Sr1 são mostradas na Tabela 3. Para os cenários Sr2, Sa1 e Sa2 as tabelas com as datas simbólicas de produção e de consumo são mostradas no apêndice A.3.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_1$
P_1	$D_i + d_1$	$D_i + d_1 + d_2 + d_4 + d_3$
P_2	$D_i + d_1$	$D_i + d_1 + d_2$
P_3	$D_i + d_1 + d_2$	$D_i + d_1 + d_2 + d_4$
P_4	$D_i + d_1 + d_2 + d_4$	$D_i + d_1 + d_2 + d_4 + d_3$
P_5	$D_i + d_1 + d_2 + d_4 + d_3$	$D_i + d_1 + d_2 + d_4 + d_3 + d_7$
P_6	$D_i + d_1 + d_2 + d_4$	$D_i + d_1 + d_2 + d_4 + d_3 + d_7 + d_8$
P_{10}	$D_i + d_1 + d_2 + d_4 + d_3 + d_7$	$D_i + d_1 + d_2 + d_4 + d_3 +$ $d_7 + d_8$
o	$D_i + d_1 + d_2 + d_4 + d_3 +$ $d_7 + d_8$	desconhecido

Tabela 3 – Datas Simbólicas de produção e de consumo para o cenário Sr1.

Os intervalos de datas simbólicas do átomo 'o' (que corresponde a conclusão dos cenários) são mostradas na Tabela 4. Lembrando que tais datas serão usadas para comparar o desempenho entre os cenários do modelos de requisitos e os cenários da arquitetura.

Na etapa 3 do método, é necessário calcular os intervalos de datas numéricas a partir das informações de intervalos de datas simbólicas apresentadas na Tabela 4. Considerando os intervalos de tempo da *public* WF-net mostrada na Figura 31 e da U(IOWF-net) mostrada na Figura 33 e o fato que o processo inicia na data 0 ($D_i = 0$), os intervalos de datas numéricas do átomo que finaliza os cenários Sr1, Sr2, Sa1 e Sa2 são apresentados na Tabela 5.

Na etapa 4 do método, deve-se comparar os intervalos de datas numéricas dos cenários da *public* WF-net e da U(IOWF-net) que são equivalentes em termos de comportamento. Como mostrado na Tabela 6, o intervalo de data numérica [3, 13] do cenário Sa1 não

Átomo 'o'	Intervalo de Datas Simbólicas de Produção
Sr1	$[D_i + d_{1min} + d_{2min} + d_{3min} + d_{4min} + d_{7min} + d_{8min},$ $D_i + d_{1max} + d_{2max} + d_{3max} + d_{4max} + d_{7max} + d_{8max}]$
Sr2	$[D_i + d_{1min} + d_{2min} + d_{4min} + d_{3min} + d_{5min} + d_{6min} + d_{9min} + d_{10min},$ $D_i + d_{1min} + d_{2max} + d_{4max}) + d_{3max} + d_{5max} + d_{6max} + d_{9max} + d_{10max}]$
Sa1	$[D_i + d_{imin} + d_{1min} + max(d_{2min} + d_{11min} + d_{12min} + d_{4min} + d_{17min},$ $max(d_{2min} + d_{11min}) + d_{12min} + d_{4min} + d_{3min} + d_{13min},$ $d_{18min} + d_{20min}) + d_{7min}) + d_{8min} + d_{omin},$ $D_i + d_{imax} + d_{1max} + max(d_{2max} + d_{11max} + d_{12max} + d_{4max} + d_{17max},$ $max(d_{2max} + d_{11max}) + d_{12max} + d_{4max} + d_{3max} + d_{13max},$ $d_{18max} + d_{20max}) + d_{7max}) + d_{8max} + d_{omax}]$
Sa2	$[D_{imin} + d_{imin} + d_{1min} + max(d_{18min} + d_{20min}, d_{2min} + d_{11min} + d_{12min} +$ $d_{4min} + max(d_{3min} + d_{13min} + d_{5min}, d_{15min}) + d_{6min} + d_{16min} + d_{9min},$ $d_{2min} + d_{11min} + d_{12min} + d_{4min} + d_{3min} + d_{13min} + d_{5min} + d_{14min}) +$ $d_{10min} + d_{omin},$ $D_{imax} + d_{imax} + d_{1max} + max(d_{18max} + d_{20max}, d_{2max} + d_{11max} + d_{12max} +$ $d_{4max} + max(d_{3max} + d_{13max} + d_{5max}, d_{15max}) + d_{6max} + d_{16max} + d_{9max},$ $d_{2max} + d_{11max} + d_{12max} + d_{4max} + d_{3max} + d_{13max} + d_{5max} + d_{14max}) +$ $d_{10max} + d_{omax}]$

Tabela 4 – Intervalos de datas simbólicas de produção do átomo 'o' para os cenários Sr1, Sr2, Sa1 e Sa2.

Átomo 'o'	Intervalos de Datas Numéricas de Produção
Sr1	[4, 17]
Sr2	[9, 32]
Sa1	[3, 13]
Sa2	[9, 21]

Tabela 5 – Intervalos de datas numéricas de produção do átomo 'o' para os cenários Sr1, Sr2, Sa1 e Sa2.

pertence ao intervalo de data numérica [4, 17] do cenário Sr1; portanto, embora os cenários Sa1 e Sr1 sejam equivalentes em termos de comportamento, é possível concluir que eles não são equivalentes em termos de desempenho. No entanto, os cenários Sa2 e Sr2, além de serem equivalentes em termos de comportamento, são também equivalentes em termos de desempenho, pois o intervalo de data numérica [9, 21] do cenário Sa2 pertence ao intervalo de data numérica [9, 32] do cenário Sr2.

Com o método proposto, é possível identificar que embora a arquitetura respeite os requisitos funcionais (comportamento) do modelo de análise, os requisitos não funcionais (desempenho) não são totalmente respeitados.

	Cenários	Intervalos de Datas Numéricas de Produção do Átomo 'o'
Cenários equivalentes em termos de comportamento	Sr1 e Sa1	[4, 17] e [3, 13]
	Sr2 e Sa2	[9, 32] e [9, 21]
Cenários equivalentes em termos de desempenho	Sr2 e Sa2	[9, 32] e [9, 21]

Tabela 6 – Comparação entre os intervalos de datas numéricas.

3.4 Estudo da complexidade dos métodos propostos

As abordagens apresentadas nas seções 3.2 e 3.3 são principalmente baseadas na construção das árvores de prova da Lógica Linear. Uma árvore de prova da Lógica Linear, correspondente a um dado cenário de uma *WorkFlow net*, terá, no pior caso, todas as transições t_i expressas na forma $c_1 \otimes c_2 \multimap c_3 \otimes c_4$. Desse modo, as três regras $\multimap L$, $\otimes R$, $\otimes L$ serão aplicadas para provar cada transição disparada. Consequentemente, a complexidade em tempo para provar um sequente linear que representa um cenário de uma *WorkFlow net* será $O(3n) = O(n)$, onde n é o número de transições do sequente (PASSOS; JULIA, 2016).

Cada sequente correspondente a um cenário potencial a ser provado é encontrado através do cálculo de componentes repetitivos estacionários que é baseado na resolução de um sistema de equações lineares de inteiros positivos. Um dos métodos para resolver sistemas de equações lineares é o método de eliminação de Gauss que possui complexidade em tempo polinomial (LAZARD, 1983). O método de eliminação de Gauss consiste em manipular um sistema de equações lineares através de determinadas operações elementares para obter uma matriz triangular, chamada também de matriz escalonada do sistema. Uma vez que o sistema foi triangularizado, a solução pode ser obtida via substituição regressiva nas equações originais.

Na quinta etapa do método proposto para a verificação dos requisitos de serviços, apresentado na seção 3.2, deve-se analisar cada componente repetitivo estacionário da U(IOWF-net) (arquitetura) verificando se estes contém todas as transições de um dos componentes repetitivos estacionários da *public* WF-net (modelo de requisitos). Essa verificação é para encontrar os cenários do modelo privado candidatos para atender os requisitos de serviços dados pelo modelo público. Os cenários do modelo privado que não são candidatos para bissimulação de um dos cenários do modelo público são desconsiderados e consequentemente não é necessária a construção das árvores de prova e dos grafos de precedência correspondentes. Esse fato suaviza a complexidade do método já que todos os cenários existentes no modelo privado não serão sistematicamente considerados nas árvores de prova da Lógica Linear.

Os grafos de precedência usados nas abordagens para verificar o comportamento dos cenários de requisitos em modelos de SOA, apresentados na seção 3.2, podem ser re-

presentados utilizando abordagens consolidadas da teoria dos grafos, como por exemplo, listas de adjacências. A representação de listas de adjacências de um grafo $G = (V, E)$ é um vetor de $|V|$ listas ligadas, onde para cada $u \in V$ a lista ligada representa os nós vizinhos de u (CORMEN et al., 2009). A busca em largura é a abordagem mais comumente usada para percorrer grafos. Dado um grafo $G = (V, E)$ e um nó de origem u, a busca em largura explora sistematicamente as arestas de G para descobrir todo nó que é alcançável a partir de u. O algoritmo descobre todos os nós que estão a uma distância k de u antes de descobrir qualquer nó que esteja a uma distância $k + 1$. O tempo total de execução do procedimento de busca em largura é $O(|V| + |E|)$, portanto, a busca em largura é executada em tempo linear considerando o tamanho da lista de adjacência de G (CORMEN et al., 2009). Desse modo, é possível comparar um grafo de precedência do modelo público com um grafo de precedência do modelo privado simplesmente verificando a compatibilidade de suas listas de adjacência. A escolha dos pares de grafos de precedência a serem comparados será baseada na inclusão de um conjunto de atividades de um cenário do modelo público dentro de um conjunto de atividades de um cenário do modelo privado.

Se um grafo de precedência do modelo privado possui todos os nós que um grafo de precedência do modelo público possui e estes nós também respeitam as mesmas restrições de adjacência, então pode-se concluir que os cenários analisados são equivalentes. Os grafos serão percorridos baseado nos nós do grafo de precedência do modelo público. Como o grafo de precedência do modelo privado pode conter arestas extras, ou seja, mais arestas que o grafo de precedência do modelo público, o tempo para percorrer os dois grafos pode ser $O(|Vr| + |Er|)$ (quando os dois grafos são percorridos exatamente da mesma forma) ou $O(|Vr| + |Ea|)$ (quando é necessário visitar arestas extras do grafo de precedência do modelo privado), onde Vr é o número de nós, Er é o número de arestas do grafo de precedência do modelo público e Ea é o número de arestas do grafo de precedência do modelo privado.

Eventualmente, pode ocorrer no modelo público um crescimento exponencial da quantidade de cenários a serem considerados em função do número de roteiros condicionais existentes no modelo. Nesse caso, haveria um crescimento exponencial da quantidade de sequentes a serem provados no modelo público. No entanto, no modelo privado somente os sequentes candidatos à equivalência por bissimulação de um dos cenários do modelo público serão provados. Além disso, quando o objetivo é encontrar cenários específicos, nem todos os cenários existentes nos modelos necessitarão ser provados.

Capítulo 4

Detecção e remoção de requisitos negativos do tipo *deadlock* em SOA

Neste capítulo são apresentados métodos, baseados na construção das árvores de prova da Lógica Linear, que podem contribuir para a confiabilidade de sistemas no contexto de SOA. Na seção 4.1 é apresentada a formalização de um método para identificar requisitos negativos do tipo *deadlock*. A seção 4.2 apresenta a melhoria de um método para remoção de estados de *deadlock* através da aplicação de regras de sincronização.

4.1 Detecção de requisitos negativos

De acordo com (AALST, 1999), um requisito negativo pode ser considerado como um comportamento defeituoso e que, portanto, não faz parte do comportamento desejado. Neste trabalho, um requisito negativo corresponde a um estado de *deadlock*, ou seja, representa um cenário (sequência de ações) que, eventualmente, pode levar o sistema a um estado indesejado e, consequentemente, não permitindo a sua correta finalização. Quando um sistema está em um estado de *deadlock* significa que a sua execução correta foi interrompida deixando, desse modo, o sistema em uma situação de espera que não permite completar o serviço requisitado. A fim de atender o requisito de confiabilidade e de manutenibilidade de uma arquitetura, é necessário diagnosticar as causas (sequência de ações) que levam a ocorrência do *deadlock* e em seguida recuperar o serviço requisitado através da alteração da sequência de ações que corresponde ao requisito negativo.

Algumas possíveis falhas que podem ocorrer em sistemas baseados em SOA são falhas de indisponibilidade de serviço, falhas temporais e falhas de composição (BHANDARI; GUPTA; UPADHYAY, 2018). Devido a incapacidade do mecanismo de composição em detectar as falhas durante a composição, o serviço pode não atender aos seus requisitos e não produzir a saída desejada. Se os critérios especificados e/ou o contrato não forem atendidos, conforme mencionado no serviço, a composição do serviço não poderá satisfazer o funcionamento desejado devido a suas violações de pré-condições, pós-condições e

invariantes (BHANDARI; GUPTA; UPADHYAY, 2018).

Conforme apresentado no Capítulo 3, o modelo de SOA definido neste trabalho é formado pela composição de *private* WF-nets que se comunicam através de mecanismos de comunicação assíncrona. Após a composição, devido à colaboração entre processos distintos, novos requisitos de serviço (comportamentos) poderão surgir. Como tais requisitos de serviço não são previstos pelo modelo público de requisito, pode ocorrer casos em que estados indesejados sejam alcançados (*deadlock*). Além disso, cada parte envolvida no processo possui relativa autonomia e poderá alterar o seu funcionamento interno a qualquer momento. Por isso, mesmo com um modelo de arquitetural detalhado, não há garantias de que, durante o funcionamento normal do sistema, novos comportamentos que não foram previstos surjam para os usuários do sistema. Isso significa que, na prática, somente durante a execução do sistema tais estados indesejados serão encontrados. Portanto, mesmo que as *private* WF-nets sejam *sound*, o modelo de SOA final, que é representado por uma U(IOWF-net), estará sujeito a possíveis situações de *deadlock* que poderão ser introduzidas pelos elementos de comunicação assíncrona. Este tipo de problema já foi mostrado em outros trabalhos como, por exemplo, em (AALST, 1998b) e (XIONG; ZHOU; PU, 2009).

Uma forma de encontrar os estados de *deadlock* em uma rede de Petri é por meio da construção do grafo de alcançabilidade que explora sistematicamente a rede. Uma desvantagem em considerar esse tipo de grafo é que pode ocorrer uma explosão do número de estados discretos, pois o processo de modelagem envolve a enumeração de todos os estados possíveis o que, geralmente, leva a uma complexidade elevada nos algoritmos de verificação utilizados. Como a exploração sistemática gera modelos muito grandes, neste trabalho, as situações de *deadlock* serão consideradas somente quando, de fato, elas ocorrerem. Desse modo, a partir dos estados indesejados, uma análise será realizada para diagnosticar as ações que levaram ao alcance de tais estados. Após o diagnóstico, é importante fornecer meios para recuperar o requisito de serviço que levou a tal situação e também para corrigir a arquitetura para impedir ocorrências futuras de tal comportamento.

Portanto, a partir de uma marcação indesejada, que representa um estado parcial do modelo, serão identificadas todas as sequências de ações, ou seja, todos os cenários que podem tornar um requisito de serviço em um requisito negativo do tipo *deadlock*. Para a identificação destas sequências de ações, será utilizado neste trabalho um raciocínio inverso como já foi utilizado, por exemplo em (DEMMOU et al., 2004) e (BOUALI; ROCHETEAU; BARGER, 2009). Isso significa que todos os arcos da U(IOWF-net) serão invertidos. Desse modo, a partir de uma marcação indesejada traça-se o caminho que foi percorrido para que tal marcação fosse alcançada. A Figura 34 (a) mostra o exemplo de uma WF-net com marcações que levam ao estado de *deadlock* (marcações nos lugares $P2$ e $P3$). Já a Figura 34 (b) mostra a mesma WF-net, porém de forma invertida. No caso da WF-net inversa, os lugares $P2$ e $P3$ se tornam os lugares iniciais e o lugar 'i', que antes

da inversão era o lugar inicial da WF-net, se torna o lugar final.

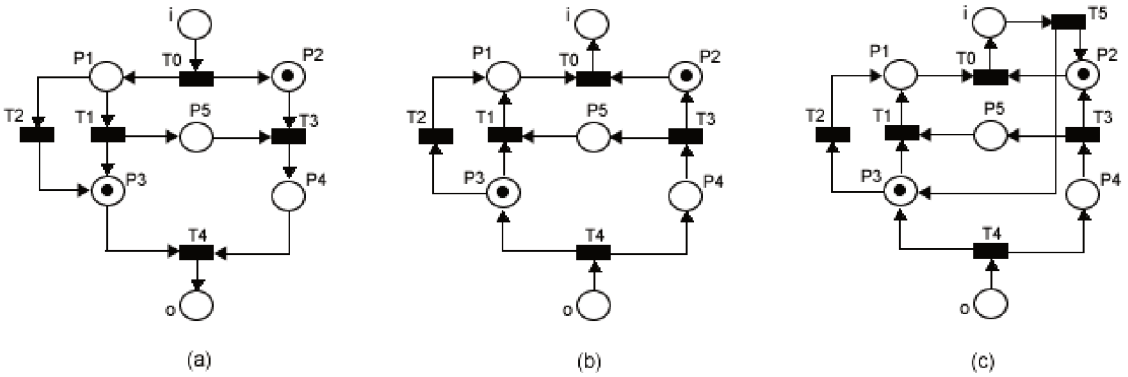


Figura 34 – Exemplo de uma WF-net invertida.

Após a inversão da U(IOWF-net), uma nova representação da rede utilizando a Lógica Linear é obtida, pois as pré condições de disparo de uma transição se transformam em pós condições e vice-versa. Além disso, na U(IOWF-net) inversa a marcação inicial será representada pelas marcações indesejadas e o lugar final será representado pelo lugar inicial da U(IOWF-net) não inversa.

Seguindo a mesma ideia do método utilizado na verificação de requisitos funcionais apresentado no Capítulo 3, para encontrar os possíveis cenários que levam o sistema a um requisito negativo, deve-se primeiramente fazer o cálculo dos componentes repetitivos estacionários da U(IOWF-net) inversa. Para isso, uma transição ligando o lugar final aos lugares com as marcações iniciais é adicionada na U(IOWF-net) inversa para transformá-la em uma rede de Petri inversa parcialmente cíclica. Os componentes repetitivos estacionários encontrados estabelecerão uma condição necessária sobre quais transições deverão ser disparadas para que, eventualmente, um invariante de transição seja encontrado na rede de Petri inversa parcialmente cíclica. Desse modo, é possível encontrar todos os cenários que levam ao alcance das marcações indesejadas. É importante ressaltar que, após a obtenção dos componentes repetitivos estacionários, a transição que liga o lugar final aos lugares iniciais é desconsiderada e a rede de Petri inversa parcialmente cíclica volta a ser uma U(IOWF-net) inversa. Observe no exemplo da Figura 34 (c) que a WF-net inversa foi transformada em uma rede de Petri inversa parcialmente cíclica. Neste exemplo, os lugares $P2$ e $P3$ representam os lugares iniciais e o lugar 'i' representa o lugar final da rede de Petri inversa parcialmente cíclica. Aplicando a análise de invariantes da ferramenta PIPE na WF-net inversa da Figura 34 (c), o componente repetitivo estacionário apresentado na Figura 35 é encontrado. Esse componente repetitivo representa uma sequência não ordenada de ações que poderão pertencer ao cenário correspondente ao requisito negativo.

Assim como foi realizado no método para verificação de requisitos funcionais apresentado no Capítulo 3, os componentes repetitivos estacionários encontrados são representados por sequentes da Lógica Linear e provados através da árvore de prova da Lógica

T0	T1	T2	T3	T4	T5
1	0	1	0	0	1

Figura 35 – Componente repetitivo estacionário da rede de Petri inversa parcialmente cíclica da Figura 34 (c).

Linear. A transição utilizada para transformar a U(IOWF-net) inversa em um rede de Petri inversa parcialmente cíclica não fará parte do sequente, pois será inserida apenas para permitir o cálculo dos componentes repetitivos estacionários. Para o exemplo da Figura 34 (c), o componente repetitivo estacionário apresentado na Figura 35 é representado pelo sequente $P2, P3, T0, T2 \vdash i$ e a sua árvore de prova, já rotulada com as transições de disparo, é apresentada na sequência:

$$\dfrac{\dfrac{\dfrac{}{P2 \vdash P2}\,{}^{i_1\ T0} \quad \dfrac{}{P1 \vdash P1}\,{}^{T2\ T0}}{P2, P1 \vdash P2 \otimes P1}\,{}^{i_1\ T2\ T0\ T0}\otimes_R \quad \dfrac{}{i \vdash i}\,{}^{T0\ f_1}}{P3 \vdash P3, P2, P1, P1 \otimes P2 \multimap i \vdash i}\,{}^{i_2\ T2\ i_1\ T2\ f_1}\multimap_{L\,T2}}\multimap_{L\,T0}$$

$$P2, P3, P3 \multimap P1, T0 \vdash i \quad {}^{i_1\ i_2\ f_1}$$

Para os cenários corretamente provados, os grafos de precedência podem ser gerados a partir dos rótulos (que representam as transições disparadas) das árvores de prova. No caso da U(IOWF-net) inversa, o grafo mostrará a ordem parcial das transições disparadas a partir das marcações iniciais (marcações indesejadas) até o lugar final (lugar inicial da U(IOWF-net) não inversa) conforme mostra o exemplo da Figura 36 (a). Neste exemplo, a marcação inicial i_1 representa uma marcação no lugar $P2$, a marcação inicial i_2 representa uma marcação no lugar $P3$ e a marcação final f_1 representa uma marcação no lugar i. Para visualizar o cenário que causa o *deadlock* a partir do lugar inicial da U(IOWF-net) não inversa, deve-se alterar a direção dos arcos do grafo de precedência conforme mostra o exemplo da Figura 36 (b). Neste caso, a marcação inicial i_1 representa uma marcação no lugar i, a marcação final f_1 representa uma marcação no lugar $P2$ e a marcação final f_2 representa uma marcação no lugar $P3$.

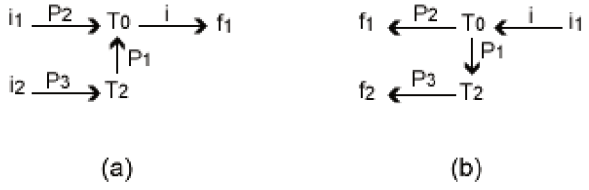


Figura 36 – Grafos de precedência referentes ao sequente $P2, P3, T0, T2 \vdash i$.

As etapas do método para formalmente identificar os cenários que podem tornar um requisito de serviço de uma SOA, representada por uma U(IOWF-net), em um requisito negativo do tipo *deadlock*, resumidamente são:

1. Representar a U(IOWF-net) de forma inversa, ou seja, os arcos de entrada de uma transição se tornarão os arcos de saída e os arcos de saída se tornarão os arcos de entrada. Na U(IOWF-net) inversa, as marcações iniciais serão as marcações que representam o estado indesejado e o lugar final é representado pelo lugar inicial da U(IOWF-net) não inversa.

2. Representar a U(IOWF-net) inversa em fórmulas da Lógica Linear.

3. Calcular os componentes repetitivos estacionários da U(IOWF-net) inversa. Para isso, deve ser adicionada uma transição ligando o lugar final da U(IOWF-net) inversa aos lugares com as marcações iniciais. Cada componente repetitivo que pode levar a U(IOWF-net) inversa a sua marcação inicial deve ser considerado como candidato a cenário de um requisito negativo.

4. Representar os cenários, identificados na etapa 3, em sequentes da Lógica Linear e prová-los por meio da construção das árvores de prova da Lógica Linear. A transição utilizada para transformar a U(IOWF-net) inversa em uma rede de Petri inversa parcialmente cíclia não é considerada na representação do cenário em sequente da Lógica Linear.

5. Gerar o grafo de precedência para os cenários corretamente provados na etapa 4. Uma vez que os cenários são encontrados de forma invertida, o grafo de precedência também é gerado de forma invertida.

Para ilustrar o método proposto, o mesmo exemplo apresentado na seção 3.2 será utilizado. A U(IOWF-net) do exemplo considerado possui cenários que levam o modelo a estados de *deadlock*, como pode ser observado nas Figuras 37 e 38 que mostram, respectivamente, a ocorrência de marcações indesejadas nos lugares C_7, CP_3, C_9 e S_9 e nos lugares C_{10}, CP_4, e S_6. A partir dessas marcações não é possível disparar nenhuma transição.

Analisando a U(IOWF-net) da Figura 37, observa-se que a marcação C_7, CP_3, C_9 e S_9 é alcançada na tentativa de execução do seguinte cenário:

$$\text{Sa3: } i, t_i, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o.$$

Após a composição, o cenário $Sa3$ é gerado, pois a $N^{priv}_{contractor}$ tenta executar a sequência $t_1, t_3, t_{13}, t_5, t_{14}, t_{10}, t'_{18}, t_{20}$ e a $N^{priv}_{subcontractor}$ tenta executar a sequência $t_2, t_{11}, t_{12}, t_4, t_{17}, t_8$. No entanto, estas sequências não são compatíveis e, desse modo, a composição de ambas produz a ocorrência do *deadlock*.

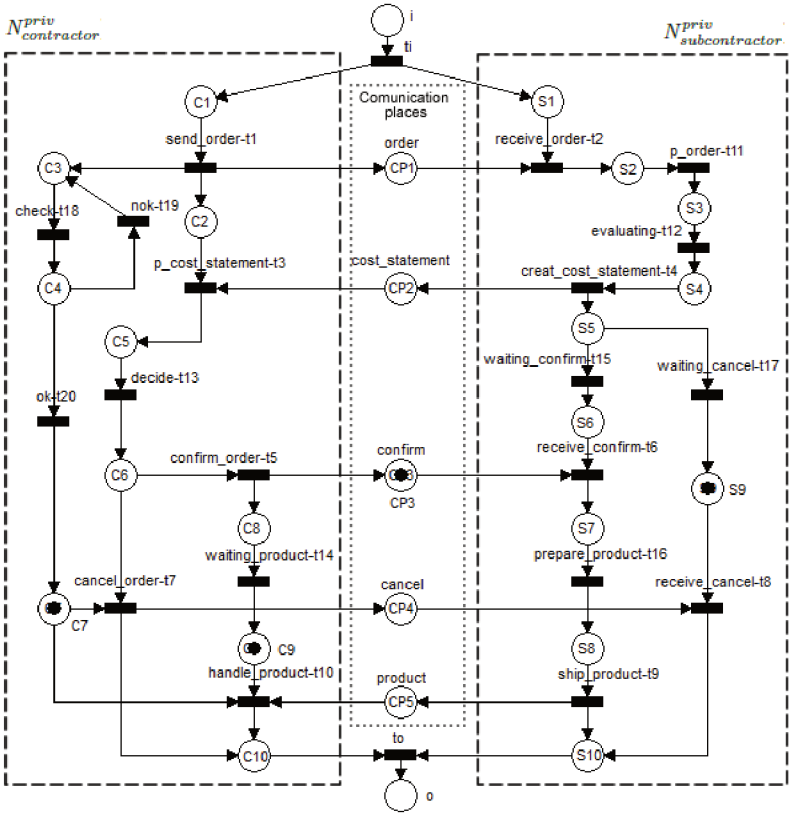


Figura 37 – U(IOWF-net) com marcações indesejadas nos lugares C_7, CP_3, C_9 e S_9.

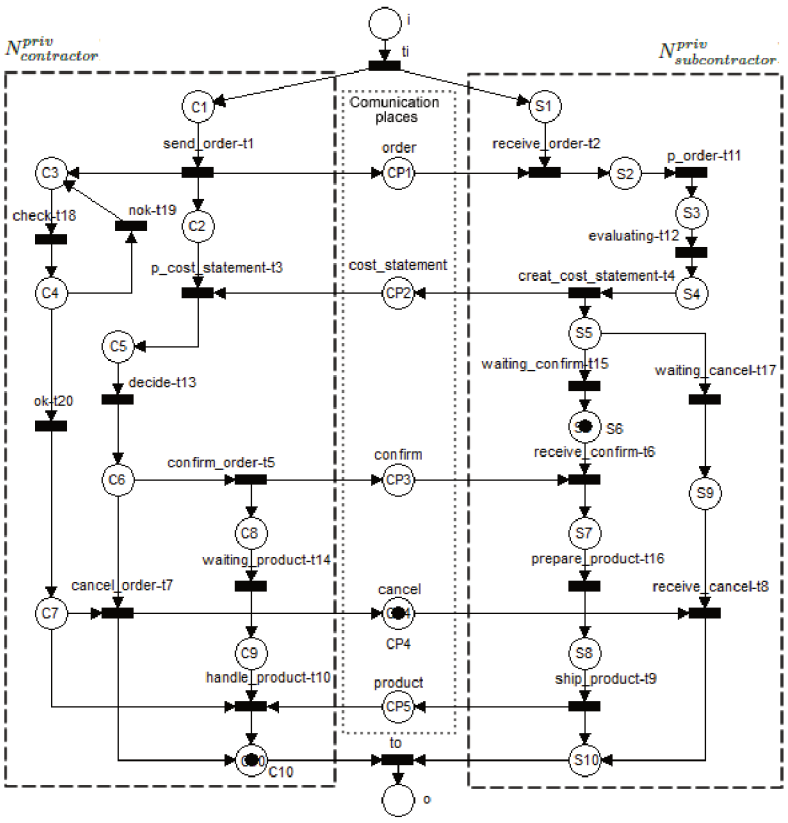


Figura 38 – U(IOWF-net) com marcações indesejadas nos lugares C_{10}, CP_4, e S_6.

Já a marcação C_{10}, CP_4, e S_6 (Figura 38) é alcançada na tentativa de execução do seguinte cenário:

$$\text{Sa4: } i, t_i, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_7, t_{15}, t_6, t_{16}, t_9, t_o \vdash o.$$

Após a composição, o cenário $Sa4$ é gerado, pois a $N^{priv}_{contractor}$ tenta executar a sequência $t_1, t_3, t_{13}, t_7, t'_{18}, t_{20}$ e a $N^{priv}_{subcontractor}$ tenta executar a sequência $t_2, t_{11}, t_{12}, t_4, t_{15}, t_6, t_{16}, t_9$. No entanto, estas sequências também não são compatíveis e, desse modo, a composição de ambas produz a ocorrência do *deadlock*.

Os cenários Sa3 e Sa4 não são executados corretamente, pois acontece um desvio após a composição das *private* WF-nets. O fluxo normal do requisito de serviço solicitado pela $N^{priv}_{contractor}$ é desviado para um requisito de serviço não compatível na $N^{priv}_{subcontractor}$. No cenário Sa3, a ação de confirmação do pedido do lado do contratante (transição t_5) é desviada para a ação de aguardar cancelamento do lado do contratado (transição t_{17}). Já no cenário Sa4, a ação de cancelamento do pedido do lado do contratante (transição t_7) é desviada para a ação de aguardar confirmação do lado do contratado (transição t_{15}).

O método apresentado, será detalhadamente ilustrado para o problema de *deadlock* da U(IOWF-net) mostrado na Figura 37. Na primeira etapa do método, a U(IOWF-net) analisada deve ser representada de forma inversa como mostra a Figura 39.

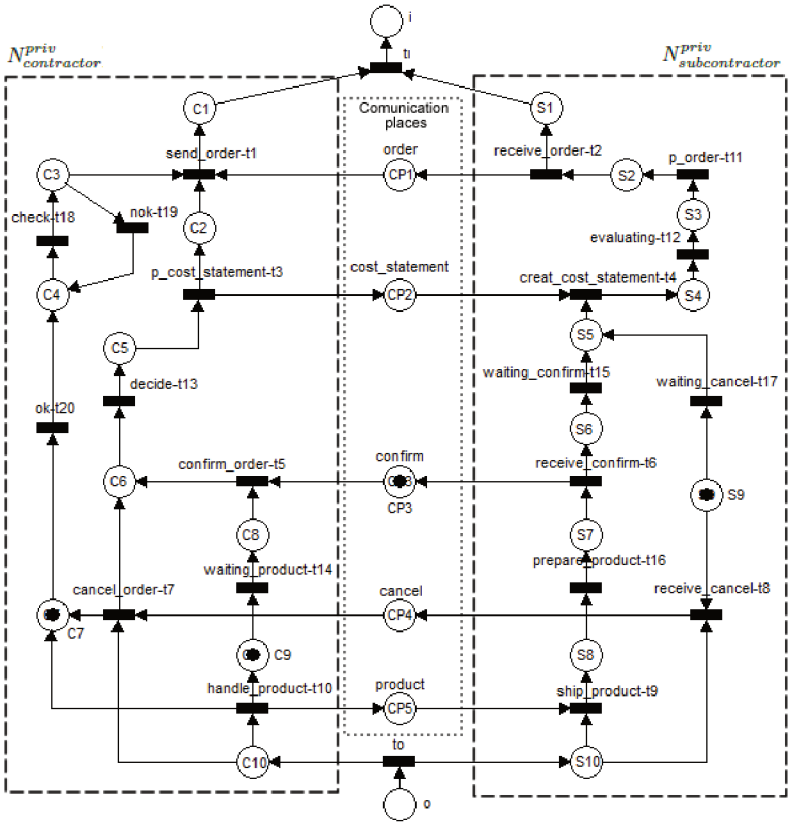


Figura 39 – U(IOWF-net) inversa com marcações nos lugares C_7, CP_3, C_9 e S_9.

Com a inversão da U(IOWF-net), as transições devem ser novamente representadas por fórmulas da Lógica Linear conforme é mostrado na sequência:

$t_i = C_1 \otimes S_1 \multimap i$,

send-order-$t_1 = t_1 = C_2 \otimes C_3 \otimes CP_1 \multimap C_1$,

receive-order-$t_2 = t_2 = S_2 \multimap S_1 \otimes CP_1$,

p-cost-statement-$t_3 = t_3 = C_5 \multimap C_2 \otimes CP_2$,

p-order-$t_{11} = t_{11} = S_3 \multimap S_2$,

check-$t'_{18} = t'_{18} = C'_4 \multimap C'_3$,

evaluating-$t_{12} = t_{12} = S_4 \multimap S_3$,

creat-cost-statement-$t_4 = t_4 = S_5 \otimes CP_2 \multimap S_4$,

decide-$t_{13} = t_{13} = C_6 \multimap C_5$,

ok-$t_{20} = t_{20} = C_7 \multimap C'_4$,

cancel-order-$t_7 = t_7 = C_{10} \otimes CP_4 \multimap C_6 \otimes C_7$,

receive-cancel-$t_8 = t_8 = S_{10} \multimap CP_4 \otimes S_9$,

confirm-order-$t_5 = t_5 = C_8 \otimes CP_3 \multimap C_6$,

waiting-confirm-$t_{15} = t_{15} = S_6 \multimap S_5$,

receive-confirm-$t_6 = t_6 = S_7 \multimap S_6 \otimes CP_3$,

waiting-product-$t_{14} = t_{14} = C_9 \multimap C_8$,

prepare-product-$t_{16} = t_{16} = S_8 \multimap S_7$,

ship-product-$t_9 = t_9 = S_{10} \otimes CP_5 \multimap S_8$,

handle-product-$t_{10} = t_{10} = C_{10} \multimap CP_5 \otimes C_9 \otimes C_7$,

waiting-cancel-$t_{17} = t_{17} = S_9 \multimap S_5$,

$t_o = o \multimap C_{10} \otimes S_{10}$.

Após a inversão da U(IOWF-net), os componentes repetitivos estacionários são calculados para encontrar todos os possíveis cenários que levam a marcação do *deadlock*. Aplicando a análise de invariantes da ferramenta PIPE na U(IOWF-net) inversa da Figura 39, o componente repetitivo estacionário apresentado na Figura 40 é encontrado. Lembrando que para essa análise, a U(IOWF-net) inversa foi transformada em uma rede de Petri inversa parcialmente cíclica.

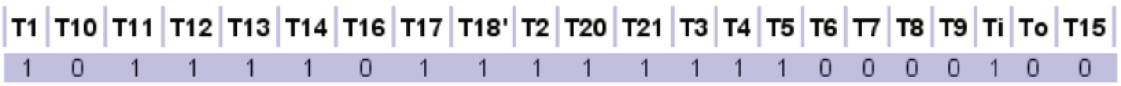


Figura 40 – Cálculo dos componentes repetitivos estacionários para a U(IOWF-net) inversa da Figura 39.

O vetor apresentado na Figura 40 mostra uma lista não ordenada de transições que pode levar a um invariante de transição na rede de Petri inversa parcialmente cíclica. Este componente repetitivo estacionário será considerado como um cenário e será chamado de

Snr1. Portanto, seguindo a etapa 4 do método, o cenário Snr1 da U(IOWF-net) inversa é representado pelo seguinte sequente da Lógica Linear:

$$\text{Snr1: } C_7, CP_3, C_9, S_9, t_1, t_{11}, t_{12}, t_{13}, t_{14}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_5, t_i \vdash i.$$

A árvore de prova completa para o cenário Snr1 é mostrada no Apêndice A.4. Na sequência é apresentada a árvore de prova resumida para este cenário.

$$\dfrac{\dfrac{\dfrac{S_1 \vdash S_1 \quad C_1 \vdash C_1}{S_1, C_1 \vdash S_1 \otimes C_1}\otimes_R \qquad i \vdash i}{\vdots}\multimap_L}{C_7, CP_3, C_9, S_9, t_1, t_{11}, t_{12}, t_{13}, t_{14}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_5, t_i \vdash i}$$

Após a prova do sequente, o grafo de precedência é construido a partir da árvore de prova rotulada. Para o cenário Snr1, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência, sendo que a árvore de prova completa é apresentada no Apêndice A.4.

$$\dfrac{\dfrac{\dfrac{\overset{t_2}{S_1} \vdash \overset{t_i}{S_1} \quad \overset{t_1}{C_1} \vdash \overset{t_i}{C_1}}{\overset{t_2}{S_1}, \overset{t_1}{C_1} \vdash \overset{t_i}{S_1} \otimes \overset{t_i}{C_1}}\otimes_R \qquad \overset{t_i}{i} \vdash \overset{f_1}{i}}{\vdots}\multimap_L}{\overset{i_1}{C_7}, \overset{i_2}{CP_3}, \overset{i_3}{C_9}, \overset{i_4}{S_9}, t_1, t_{11}, t_{12}, t_{13}, t_{14}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_5, t_i \vdash \overset{f_1}{i}}$$

O grafo de precedência correspondente ao cenário Snr1, gerado a partir da árvore de prova rotulada com as transições de disparo, é apresentado na Figura 41. Observando o grafo de precedência é possível verificar a ordem parcial de todas as transições disparadas que leva um requisito de serviço da U(IOWF-net) a um estado de *deadlock*, ou seja, o cenário que corresponde ao requisito negativo. Desse modo, a partir das informações apresentadas no grafo de precedência, a sequência de disparos pode ser analisada para verificar o motivo da ocorrência do *deadlock*. Além disso, tais informações também servirão de base para soluções que controlem a ocorrência do *deadlock*.

No exemplo considerado, existe apenas um cenário que, a partir da marcação inicial i, leva à marcação dos lugares C_7, CP_3, C_9 e S_9, conforme é demonstrado pelo cálculo dos componentes repetitivos estacionários. Por isso, somente um grafo de precedência foi gerado.

Na Figura 42 é apresentado o grafo de precedência com a direção dos arcos invertidos, ou seja, a partir do lugar inicial da U(IOWF-net) não inversa em direção as marcações indesejadas. Este grafo deixa claro que a marcação no lugar $C7$ foi em decorrência do disparo da transição t_{20}, a marcação no lugar $CP3$ em decorrência do disparo da transição

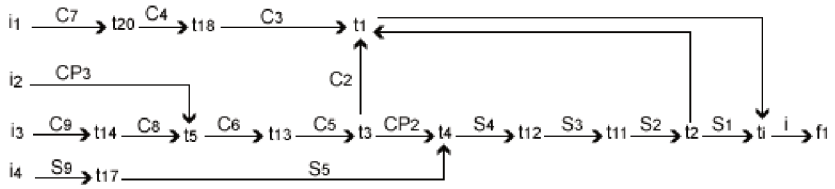


Figura 41 – Grafo de precedência do cenário Srn1.

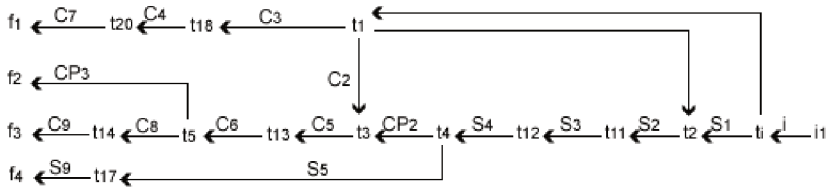


Figura 42 – Grafo de precedência do cenário Srn1 com os arcos invertidos.

t_5, a marcação no lugar $C9$ em decorrência do disparo da transição t_{14} e a marcação no lugar $S9$ em decorrência do disparo da transição t_{17}.

O objetivo do método apresentado nesta seção é apenas para identificar os cenários, ou seja, as sequências de eventos que geram um requisito negativo do tipo *deadlock*. Esta identificação é o primeiro passo para detectar o motivo da ocorrência do requisito negativo para que posteriormente alguma intervenção realizada para evitar futuras ocorrências de tal requisito.

4.2 Controle do *Deadlock*

Conforme apresentado em (PASSOS, 2016), uma forma de controlar a ocorrência do *deadlock* em uma U(IOWF-net) é através do monitoramento da execução dos cenários livres de *deadlock*. Para isso, as informações derivadas das árvores de prova da Lógica Linear com o cálculo de datas podem ser utilizadas. No método apresentado em (PASSOS, 2016), para cada cenário livre de *deadlock*, as datas de produção dos átomos que correspondem aos lugares de entrada das transições a serem monitoradas devem ser consideradas. Desse modo, guardas são associadas as transições de cada cenário livre de *deadlock*. Uma guarda representa uma condição de disparo de uma transição, sendo apresentada por uma expressão ou lista de expressões booleanas entre colchetes. As transições t_i e t_o em um sistema composto não precisam ser monitoradas, pois poderiam ser removidas do modelo sem implicar em alterações semânticas no mesmo. Desta forma, as datas D_i e d_i não precisam aparecer no modelo a ser monitorado. As datas d_j , onde $j \neq i$ e $j \neq o$ que aparecem nas datas de produção dos átomos que representam os lugares de entrada da transição considerada devem compor a guarda de tal transição.

Para exemplificar o método definido em (PASSOS, 2016), considere o exemplo apresentado na seção 3.2. A U(IOWF-net) do exemplo considerado (Figura 19) possui dois cenários que são livres de *deadlock* e são executados corretamente, sendo eles Sa1 e Sa2. No entanto, o exemplo possui dois cenários, Sa3 e Sa4, que levam a situações de *deadlock*, sendo representadas respectivamente pelas marcações indesejadas nos lugares C_7, CP_3, C_9, S_9 e nos lugares C_{10}, CP_4, S_6. Desse modo, para tentar evitar a ocorrência dessas marcações indesejadas, o monitoramento dos cenários Sa1 e Sa2 pode ser realizado.

Considerando as árvores de prova com o cálculo de datas para os cenários Sa1 e Sa2, apresentadas no Apêndice A.2, obtêm-se as seguintes guardas para as transições pertencentes a estes cenários:

$t_2 = [d_1]$,
$t_3 = [d_1, d_2, d_4, d_{11}, d_{12}]$,
$t_4 = [d_1, d_2, d_{11}, d_{12}]$,
$t_5 = [d_1, d_2, d_3, d_4, d_{11}, d_{12}, d_{13}]$,
$t_6 = [d_1, d_2, d_3, d_4, d_5, d_{11}, d_{12}, d_{13}, d_{15}]$,
$t_7 = [d_1, d_2, d_3, d_4, d_{11}, d_{12}, d_{13}, d_{18}, d_{20}]$,
$t_8 = [d_1, d_2, d_3, d_4, d_7, d_{11}, d_{12}, d_{13}, d_{17}, d_{18}, d_{20}]$,
$t_9 = [d_1, d_2, d_3, d_4, d_5, d_6, d_{11}, d_{12}, d_{13}, d_{15}, d_{16}]$,
$t_{10} = [d_1, d_2, d_3, d_4, d_5, d_6, d_9, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}, d_{16}, d_{18}, d_{20}]$,
$t_{11} = [d_1, d_2]$,
$t_{12} = [d_1, d_2, d_{11}]$,
$t_{13} = [d_1, d_2, d_3, d_4, d_{11}, d_{12}]$,
$t_{14} = [d_1, d_2, d_3, d_4, d_5, d_{11}, d_{12}, d_{13}]$,
$t_{15} = [d_1, d_2, d_4, d_{11}, d_{12}]$,
$t_{16} = [d_1, d_2, d_3, d_4, d_5, d_6, d_{11}, d_{12}, d_{13}, d_{15}]$,
$t_{17} = [d_1, d_2, d_4, d_{11}, d_{12}]$,
$t'_{18} = [d_1]$,
$t_{20} = [d_1, d_{18}]$.

Neste caso, uma guarda associada a uma determinada transição indica quais ações devem acontecer antes de tal transição ser disparada. A guarda $t_3 = [d_1, d_2, d_4, d_{11}, d_{12}]$, por exemplo, indica que a transição t_3 somente será disparada se as transições $t_1, t_2, t_4, t_{11}, t_{12}$ tiverem sido disparadas anteriomente. A data de produção do átomo que representa o lugar de entrada (C_1) da transição t_1 é $D_i + d_i$. Como as datas D_i e d_i não devem aparecer no modelo monitorado, nenhuma guarda é associada a esta transição.

Para verificar se o monitoramento evita a situação de *deadlock* para o exemplo considerado, a U(IOWF-net) apresentada na Figura 19 foi implementada no CPN Tools com as guardas associadas as transições, como pode ser observado na Figura 43. Para esta representação, foi criada uma lista de inteiros (INTList) e três variáveis deste tipo, l, l1 e l2, para armazenarem as transições já disparadas. A cada disparo de uma transição t_j

onde $j \neq i$ e $j \neq o$, o inteiro j é adicionado à lista de transições já disparadas, através da função $e :: l$ que insere o elemento e na cabeça da lista l. A função $l\hat{}\hat{}l1$ concatena as listas l e $l1$ e deve ser aplicada quando uma transição tem mais de um lugar de entrada. Já a função *mem* l x retorna *true* (verdadeiro) se o elemento x pertence à lista l. Para este exemplo, não foi necessário verificar mais de uma guarda nas transições, mas caso seja necessário, pode-se utilizar, entre as guardas, vírgulas que representarão conjunções ('e' lógico) ou o elemento sintático *orelse* que corresponderá a uma disjunção ('ou' lógico). Para melhor visualização das transições disparadas, pode ser utilizada a função *remdupl* que remove elementos repetidos da lista l.

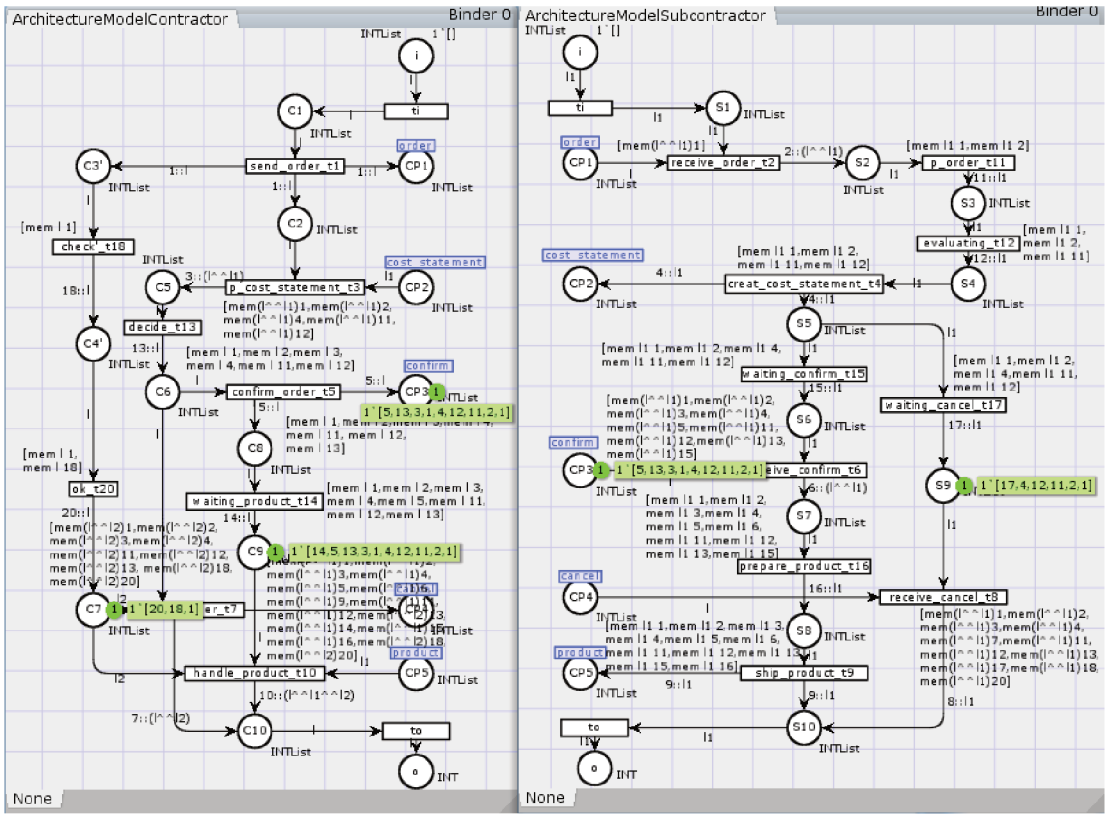


Figura 43 – U(IOWF-net) com guardas associadas as transições.

Como pode ser observado na Figura 43, mesmo após a definição de guardas para todas as transições pertencentes aos cenários que são executados corretamente (Sa1 e Sa2), as marcações indesejadas ainda são alcançadas. Conforme apresentado na seção 4.1, o *deadlock* causado pelos cenários Sa3 e Sa4 é devido a ocorrência de um desvio após a composição das private WF-nets. No cenário Sa3, a ação de confirmação do pedido executada na $N^{priv}_{contractor}$ é desviada para a ação de aguardar cancelamento na $N^{priv}_{subcontractor}$ (transição t_{17}). Já no cenário Sa4, a ação de cancelamento do pedido executada na $N^{priv}_{contractor}$ é desviada para a ação de aguardar confirmação na $N^{priv}_{subcontractor}$ (transição t_{15}). Portanto, observa-se que as transições t_{15} e t_{17} estão em conflito; no entanto, possuem exatamente as mesmas condições de disparo. Desse modo, considerar somente as guardas de disparos

nessas transições não resolve o conflito existente entre elas e, consequentemente, não evita as situações de *deadlock*. Dessa maneira, o monitoramento através das guardas associadas as transições nem sempre funcionará.

De acordo com (AALST, 1998b), uma outra maneira possível para prevenir as situações de *deadlock* causadas pela troca de mensagens em um *workflow* interorganizacional é sincronizar partes dos processos locais. A sincronização força os processos de *workflow* locais executarem simultaneamente certas atividades, removendo, desse modo, a situação de *deadlock* do modelo.

A sincronização é através da inserção de um elemento de comunicação síncrona, ou seja, transições que são sincronizadas e forçadas a serem disparadas ao mesmo tempo. Segundo (AALST, 1998b), a comunicação síncrona corresponde à fusão de transições; portanto, o termo conjunto de fusão também é utilizado para denotar um elemento de comunicação síncrona. Observe que no exemplo da Figura 44, o elemento de comunicação síncrona CT1 força a execução simultanea das transições T3 e T7.

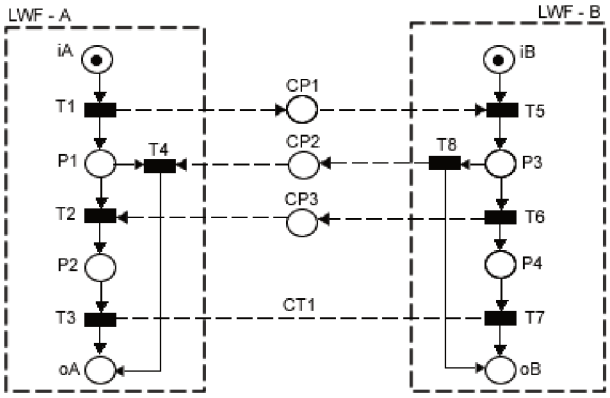


Figura 44 – Exemplo de elemento de comunicação síncrona.

Em (OLIVEIRA et al., 2017b), foi apresentado um método baseado na análise dos sequentes da Lógica Linear para tratar as situações de *deadlock* em processos de *workflow* interorganizacionais modelados por IOWF-nets que são localmente, porém não globalmente *sound*. A substituição de elementos de comunicação assíncrona por novos mecanismos de comunicação parcialmente síncronos força, em particular, os processos de *workflow* locais a executarem simultaneamente certas atividades, removendo a situação de *deadlock* do modelo. Além do método apresentado em (OLIVEIRA et al., 2017b) tratar somente uma situação específica, para que a sincronização ocorra é necessário alterar internamente um dos processos locais, sendo que essa alteração consiste no acréscimo de um lugar e uma transição auxiliar. Desse modo, baseada na abordagem apresenta em (OLIVEIRA et al., 2017b), na sequência, é apresentado um método para prevenir a ocorrência de um requisito negativo do tipo *deadlock* que considera diferentes tipos de situações. O método proposto consiste na substituição de mecanismos de comunicação assíncrona por mecanismos de comunicação síncrona ou na inserção de novos mecanismos de comunicação

síncrona com o objetivo de prevenir a ocorrência de mensagens perdidas que, em caso de situações de *deadlock*, ficam presas em lugares de comunicação assíncrona. Nenhuma alteração interna dos processos locais é necessária, pois as alterações ocorrem somente nos canais de comunicação.

Para aplicar as regras de sincronização, primeiramente é necessário identificar dentro dos potenciais cenários da U(IOWF-net) quais são os responsáveis pelas situações de *deadlock*. A identificação é realizada conforme o método apresentado na seção 4.1.

Como as *private* WF-nets que compõem uma U(IOWF-net) são *sound*, o problema de *deadlock* se caracteriza pelo disparo de uma transição que deixa uma ficha presa em um lugar de comunicação assíncrona. Esta transição, que pertence a uma *private* WF-net A, será chamada de t_{d1}, e o lugar de comunicação que possui a ficha presa será chamado de *cp*.

O estado de *deadlock* ocorre, pois a ficha que é depositada no lugar *cp* não é consumida. Isso significa que a transição de saída do lugar *cp*, que pertence a uma *private* WF-net B e será chamada de t_{d2}, não é disparada. O disparo da transição t_{d2} não ocorre, pois um outro caminho é percorrido na *private* WF-net B, ou seja, o requisito de serviço que deveria ser realizado é desviado após a composição.

Na *private* WF-net B, deve existir um lugar com bifurcações (lugar que permite a escolha de outros caminhos) que diretamente sensibiliza a transição t_{d2} ou que leva à sensibilização de tal transição, pois, se não existisse, o *deadlock* não aconteceria. Uma marcação neste lugar com bifurcações é o que permite o disparo da transição que desvia o fluxo normal do sistema para o fluxo que causa o *deadlock*. Desse modo, se este lugar sensibiliza diretamente a transição t_{d2}, ele será chamado de *pb* e a regra de sincronização aplicada será a substituição do lugar de comunicação assíncrona, que possui uma ficha presa, por um elemento de comunicação síncrona, conforme mostra a representação na Figura 45. Mas se este lugar não sensibiliza diretamente a transição t_{d2}, então será chamado de pb' e será necessário identificar qual transição é diretamente sensibilizada por pb'. Essa transição será chamada de t'_{d2} e um elemento de comunicação síncrona será criado entre as transições t_{d1} e t'_{d2}, conforme mostra a representação na Figura 46.

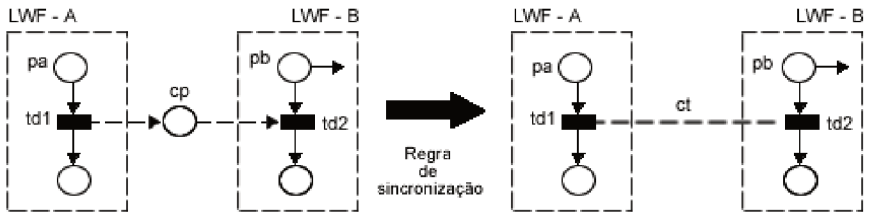


Figura 45 – Caso 1: Substituição de comunicação assíncrona por comunicação síncrona.

Como pode ser observado na Figura 45, antes da aplicação da regra de sicronização, para que a transição t_{d1} seja disparada, é necessário somente a disponibilidade de uma

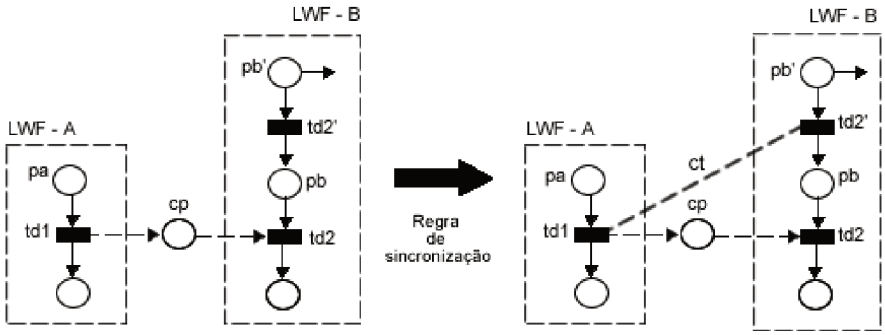


Figura 46 – Caso 2: Inserção de mecanismo de comunicação síncrona.

ficha no lugar *pa*. O disparo da transição t_{d1} gera uma ficha no lugar *cp*, mas não garante o disparo da transição t_{d2} que depende da disponiblidade de uma ficha no lugar *cp* e no lugar *pb*. Desse modo, caso não haja a disponibilidade de uma ficha no lugar *pb*, a transição t_{d2} não será disparada e a situação de *deadlock* ocorrerá com uma ficha presa no lugar *cp*. Por isso, a solução é exigir que as transições t_{d1} e t_{d2} sejam disparadas ao mesmo tempo, sendo necessário a disponibilidade de uma ficha no lugar *pa* e no lugar *pb*. Nesse caso, o lugar de comunicação assíncrona será substituído por um elemento de comunicação síncrona entre as transições t_{d1} e t_{d2}.

No caso apresentado na Figura 46, o lugar com bifurcações não sensibiliza diretamente a transição t_{d2}. Por isso, não será necessário substituir o lugar de comunicação assíncrona que possui a ficha presa, mas sim criar um elemento de comunicação síncrona entre a transição t_{d1} e a transição t'_{d2} que é diretamente sensiblizada por uma ficha no lugar pb'.

Pode existir na *private* WF-net A um lugar com bifurcações que não seja diretamente o lugar de entrada da transição t_{d1}, mas que leva a sua sensibilização. Caso este lugar, que será chamado de pa', exista, será considerada para a sincronização a transição que é diretamente sensibilizada por uma ficha neste lugar. Desse modo, a transição de saída do lugar pa', que será chamada de t'_{d1}, será considerada para a sincronização. Observe nas Figuras 47 e 48 as possibilidades de sincronização para este caso.

A Figura 47 mostra que, quando a sincronização deve ocorrer entre as transições t'_{d1} e t_{d2}, o lugar de comunicação assíncrona que existia entre as transições t_{d1} e t_{d2} é substituído por um elemento de comunicação síncrona entre as transições t'_{d1} e t_{d2}. Já a Figura 48 mostra que, quando a sincronização deve ocorrer entre as transições t'_{d1} e t'_{d2}, um elemento de comunicação síncrona é inserido entre essas duas transições.

Existem casos onde o lugar de comunicação *cp* pode possuir mais de um arco de saída. Desse modo, todos os cenários gerados por essas saídas devem ser considerados. Isso significa que, neste caso, mais de uma sincronização deverá ser realizada no modelo. O interessante de usar o método apresentado na seção 4.1 para a identificação dos cenários responsáveis pelas situações de *deadlock* é que, caso o lugar *cp* possua mais de um arco de entrada, o método verifica todos os possíveis cenários que deverão ser tratados.

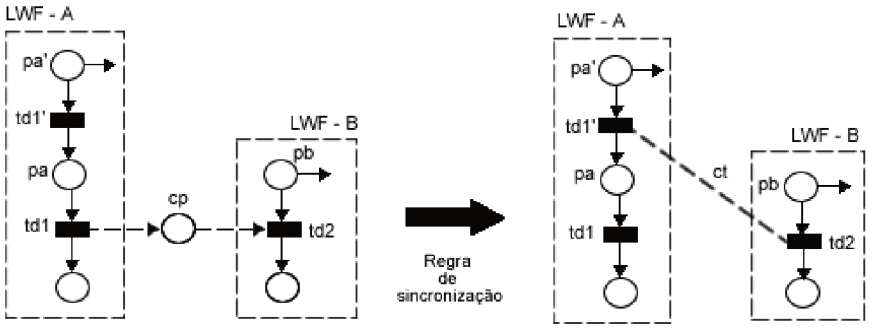


Figura 47 – Caso 3: Lugar com bifurcações na *private* WF-net A - substituição de comunicação assíncrona por comunicação síncrona.

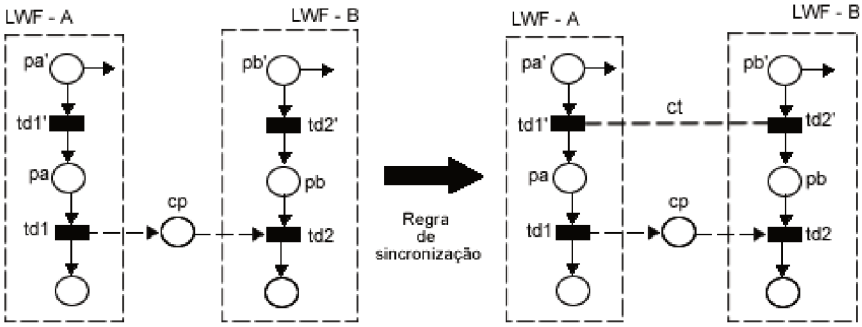


Figura 48 – Caso 4: Lugar com bifurcações na *private* WF-net A - inserção de mecanismo de comunicação síncrona.

Resumidamente, as etapas do método para formalmente realizar a sincronização que evita que um requisito de serviço de uma SOA, representada por uma U(IOWF-net), se transforme em um requisito negativo do tipo *deadlock* são:

1. Identificar dentro dos potenciais cenários do modelo, quais são os responsáveis pelas situações de *deadlock*.

2. Identificar o lugar de comunicação, chamado de *cp*, que possui uma ficha presa.

3. Identificar a transição de entrada do lugar *cp*. Tal transição, que pertence a uma *private* WF-net A, será chamada de t_{d1}.

4. Verificar se há algum lugar com bifurcações que uma vez marcado leva a sensibilização da transição t_{d1}. Caso este lugar exista e caso seja o lugar de entrada da transição t_{d1}, ele será chamado de *pa* e a transição t_{d1} será considerada para a sincronização. Mas caso o lugar com bifurcações não seja o lugar de entrada da transição t_{d1}, ele será chamado de pa' e a transição de saída do lugar pa', chamada de t'_{d1}, será considerada para a sincronização.

5. Identificar qual é a transição de saída do lugar *cp*. Essa transição, que pertence a uma *private* WF-net B, será chamada de t_{d2}. Caso exista mais de uma transição de saída, todos os cenários gerados por estas transições devem ser considerados.

6. Identificar o lugar com bifurcações que uma vez marcado leva a sensibilização da transição t_{d2}. Caso este lugar seja o lugar de entrada da transição t_{d2}, ele será chamado de *pb* e a transição t_{d2} será considerada para a sincronização. Mas caso o lugar com bifurcações não seja o lugar de entrada da transição t_{d2}, ele será chamado de pb' e a transição de saída do lugar pb', chamada de t'_{d2}, será considerada para a sincronização.

7. Aplicar as regras de sincronização no modelo:

   - sincronização da transição t_{d1} com a transição t_{d2} - substitui o lugar de comunicação assíncrona que existe entre essas duas transições por um elemento *ct* de comunicação síncrona;
   - sincronização da transição t_{d1} com a transição t'_{d2} - cria um elemento *ct* de comunicação síncrona entre essas duas transições;
   - sincronização da transição t'_{d1} com a transição t_{d2} - substitui o lugar de comunicação assíncrona que existe entre as transições t_{d1} e t_{d2} por um elemento *ct* de comunicação síncrona entre as transições t'_{d1} e t_{d2};
   - sincronização da transição t'_{d1} com a transição t'_{d2} - cria um elemento *ct* de comunicação síncrona entre essas duas transições.

Considerando que *pe* representa os lugares de entrada e *po* representa os lugares de saída de uma transição, a representação da sincronização por meio das fórmulas da Lógica Linear se dará da seguinte maneira:

- ❏ na sincronização da transição t_{d1} com a transição t_{d2}, a condição de disparo para estas duas transições será: $pe_{t_{d1}} \otimes pe_{t_{d2}} \multimap po_{t_{d1}} \otimes po_{t_{d2}}$;
- ❏ na sincronização da transição t_{d1} com a transição t'_{d2}, a condição de disparo para estas duas transições será: $pe_{t_{d1}} \otimes pe_{t'_{d2}} \multimap po_{t_{d1}} \otimes po_{t'_{d2}}$;
- ❏ na sincronização da transição t'_{d1} com a transição t_{d2}, a condição de disparo para estas duas transições será: $pe_{t'_{d1}} \otimes pe_{t_{d2}} \multimap po_{t'_{d1}} \otimes po_{t_{d2}}$;
- ❏ na sincronização da transição t'_{d1} com a transição t'_{d2}, a condição de disparo para estas duas transições será: $pe_{t'_{d1}} \otimes pe_{t'_{d2}} \multimap po_{t'_{d1}} \otimes po_{t'_{d2}}$;

Com a realização da sincronização, é necessário reprocessar as árvores de prova da Lógica Linear para os cenários que possuem as transições que foram sincronizadas. Considerando que até o momento de disparo da transição t_{d1} ou t'_{d1} o cenário não é alterado,

é necessário reprocessar as árvores de prova apenas a partir do sequente onde a transição t_{d1} ou t'_{d1} é disparada. Uma grande vantagem desta abordagem é que quando uma situação de *deadlock* ocorre, somente a parte alterada pela regra de sincronização precisa ser reprocessada sendo que o restante da árvore de prova da Lógica Linear permanecerá inalterada. Desse modo, não é necessário reiniciar o processo desde o início, mas a partir do ponto que deixou de seguir o fluxo normal para seguir o fluxo que causa o *deadlock*.

Para ilustração, considere o problema de *deadlock* apresentado na Figura 37. Neste exemplo, as marcações indesejadas estão presentes nos lugares C_7, CP_3, C_9 e S_9.

Utilizando o método apresentado na seção 4.1, é possível identificar quais cenários de requisitos de serviços foram transformados em um requisito negativo devido a ocorrência do *deadlock*. Após a aplicação do método, como pode ser visto na seção 4.1, foi encontrado apenas um cenário, chamado de Srn1, que, a partir da marcação inicial i, leva a marcação dos lugares C_7, CP_3, C_9 e S_9. Este cenário é representado de forma inversa pelo grafo de precedência apresentado na Figura 41.

Na segunda etapa do método, é necessário identificar o lugar cp, ou seja, o lugar de comunicação que possui uma ficha presa. Portanto, basta analisar quais dos lugares que possuem as marcações indesejadas é um lugar de comunicação. No exemplo considerado, $cp = CP_3$.

Na terceira etapa do método, é necessário identificar a transição de entrada t_{d1} do lugar $cp = CP_3$. Esta identificação pode ser realizada analisando a U(IOWF-net) ou analisando o grafo de precedência que pode ser gerado após a identificação do cenário que causa o *deadlock*. Como o grafo de precedência é gerado de forma inversa, a transição t_{d1} será aquela que é disparada pela marcação do lugar $cp = CP_3$. No exemplo considerado, $t_{d1} = t_5$.

Na quarta etapa do método, deve-se verificar se há algum lugar com bifurcações que uma vez marcado leva à sensibilização da transição $t_{d1} = t_5$. Para verificar se este lugar existe, pode-se identificar no grafo de precedência inverso os lugares que são marcados a partir da transição t_{d1} e em seguida verificar se estes lugares possuem bifurcações ou não. Os lugares no grafo de precedência são representados pelos rótulos das arestas. No exemplo considerado, o lugar de entrada da transição $t_{d1} = t_5$ é o lugar que possui bifurcações; desse modo, $pa = C_6$ e a transição $t_{d1} = t_5$ será considerada para a sincronização.

Na quinta etapa do método, deve-se identificar qual é a transição de saída t_{d2} do lugar $cp = CP_3$. Como a transição t_{d2} não faz parte do cenário que causa o *deadlock*, não é possível identificá-la pelo grafo de precedência inverso; desse modo, é preciso verificar na U(IOWF-net) qual é a transição de saída do lugar $cp = CP_3$ que, neste caso, é $t_{d2} = t_6$. Como não há outras transições de saída no lugar $cp = CP_3$, não existe outros cenários a serem considerados para sincronização.

Na sexta etapa do método, é necessário identificar o lugar com bifurcações que uma vez marcado leva à sensibilização da transição $t_{d2} = t_6$. Analisando a U(IOWF-net), identifica-

se que o lugar de entrada da transição $t_{d2} = t_6$ não é o lugar que possui bifurcações; desse modo, analisando os outros lugares, identifica-se que o lugar com bifurcações é $pb' = S_5$. Portanto, a transição de saída do lugar $pb' = S5$ será considerada para a sincronização, ou seja, $t'_{d2} = t_{15}$.

Na sétima etapa do método, as regras de sincronização são aplicadas de acordo com as informações encontradas nas etapas anteriores. No exemplo analisado, a transição $t_{d1} = t_5$ deve ser sincronizada com a transição $t'_{d2} = t_{15}$; assim, de acordo com as regras de sincronização, um elemento *ct* de comunicação síncrona deve ser criado entre essas duas transições.

Para o problema de *deadlock* apresentado na Figura 38, o mesmo processo é realizado. Neste caso, a transição $t_{d1} = t_7$ deve ser sincronizada com a transição $t'_{d2} = t_{17}$ e, de acordo com as regras de sincronização, um elemento *ct* de comunicação síncrona deve ser criado entre essas duas transições.

Com a aplicação das regras de sincronização, o requisito negativo deixa de existir, pois o cenário que causa o *deadlock* será sempre evitado mediante a condição de disparo simultâneo associada as transições t_5 e t_{15} e entre as transições t_7 e t_{17}. Isso significa que somente os cenários seguros, que no caso são Sa1 e Sa2, serão executados.

A Figura 49 apresenta a U(IOWF-net) com a correção do *deadlock* tanto para o problema apresentado na Figura 37 quanto para o problema apresentado na Figura 38.

Com a introdução dos mecanismos de sincronização, as transições t_5 e t_{15} passam a ser representadas pela seguinte fórmula da Lógica Linear: $C_6 \otimes S_5 \multimap CP_3 \otimes C_8 \otimes S_6$. Já as transições t_7 e t_{17} passam a ser representadas pela seguinte fórmula da Lógica Linear: $C_6 \otimes C_7 \otimes S_5 \multimap CP_4 \otimes C_{10} \otimes S_9$. Com isso, na árvore de prova, somente serão alteradas as linhas onde ocorre o disparo destas transições.

O lugar *pa* ou *pa*$'$ define a condição que levou a marcação do lugar de comunicação *cp*, já o lugar *pb* ou *pb*$'$ define a condição que levou a mudança do fluxo normal para o fluxo que causa o *deadlock*. Portanto, após a correção do *deadlock*, uma marcação parcial nesses lugares estabelece uma condição suficiente para reiniciar o processo a partir do ponto que levou a ocorrência do *deadlock* sem a necessidade de repetir as partes do processo que foram executadas corretamente. No exemplo considerado nesta seção, para o primeiro problema de *deadlock*, o processo pode ser reiniciado a partir da marcação dos lugares C_6 e S_5, já para o segundo problema de *deadlock* o processo pode ser reiniciado a partir da marcação dos lugares C_6, C_7 e S_5, conforme mostra a Figura 49.

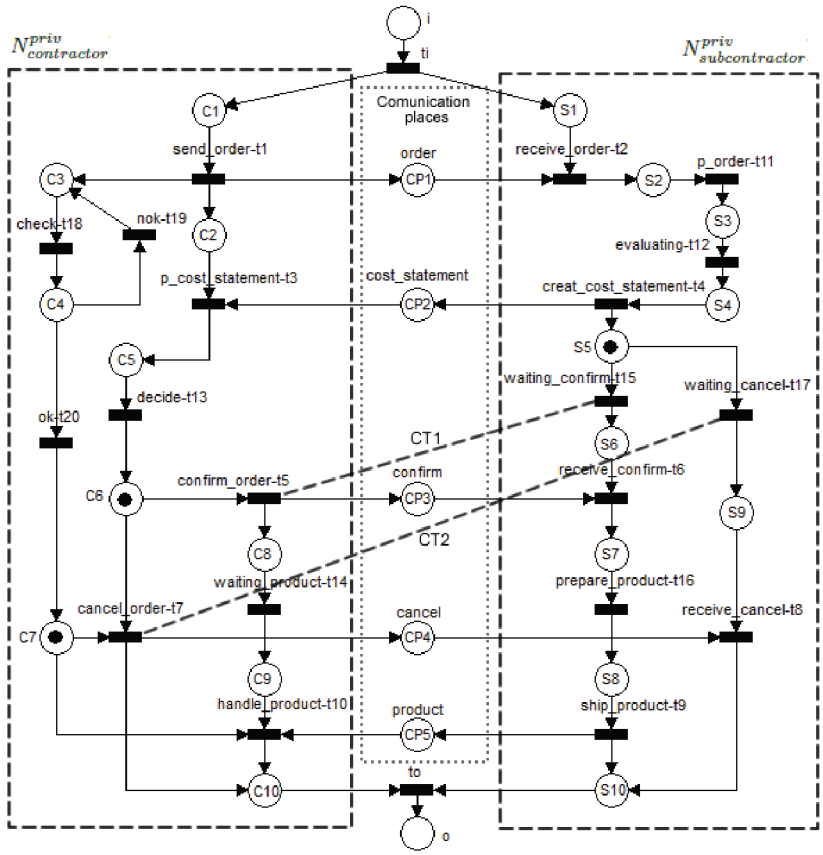


Figura 49 – U(IOWF-net) com correção do *deadlock* através de mecanismos de comunicação síncrona.

Capítulo 5

Estudo de Caso

A computação orientada a serviços (SOC) emergiu como um importante paradigma de computação e mudou a maneira como os aplicativos de software são projetados, entregues e consumidos. Para criar o modelo de serviço, SOC conta com a Arquitetura Orientada a Serviços (SOA), que é uma maneira de reorganizar aplicações e infraestrutura de software em um conjunto de serviços que interagem (PAPAZOGLOU, 2003). SOA tem sido amplamente usada para integrar sistemas por meio de serviços que podem ser reutilizados por vários sistemas. A maioria das implementações de SOA é baseada em tecnologias de serviços *Web* (ROSEN et al., 2012). De acordo com (ERL, 2009), serviços *Web* corresponde a plataforma tecnológica mais associada à realização de SOA.

Uma das principais características dos serviços *Web* é o fraco acoplamento existente entre os serviços e a existência de padrões de interoperabilidade que permitem que determinados serviços sejam agrupados. As composições de serviços podem ser implementadas de várias maneiras, como, por exemplo, através de orquestrações e coreografias (ERL, 2005). Nos dois casos, os mecanismos de composições são baseados em mecanismos de interações síncronas ou assíncronas.

Para a especificação e análise de composições de serviços *Web*, muitos estudos já consideraram as redes de Petri como um modelo apropriado (HAMADI; BENATALLAH, 2003), (MARTENS, 2005), (XIONG; FAN; ZHOU, 2010), (VALERO et al., 2012), (KLAI; OCHI; TATA, 2013), (PASSOS; JULIA, 2015). Em (PASSOS; JULIA, 2015), por exemplo, foi apresentado um método para a identificação de cenários livres de *deadlock* em composições de serviços *Web* com base na análise das árvores de prova da Lógica Linear; a abordagem apresentada pelos autores detecta cenários seguros específicos. Em (MARTENS, 2005), os serviços, denominados módulos, são classificados como utilizáveis (módulos que podem ser usados em qualquer composição) ou não utilizáveis (módulos que não podem ser usados em qualquer composição); na abordagem proposta, foi apresentado um *framework* para a modelagem e análise de serviços *Web* com base em processos de negócios com a ajuda de redes de Petri. Em (KLAI; OCHI; TATA, 2013), os autores abordam o problema de abstrair e verificar a exatidão das composições de serviços *Web*

levando em consideração quatro variantes da propriedade *Soundness* (*Soundness*, *Weak Soundness*, *Relaxed Soundness* e *Easy Soundness*).

Desse modo, considerando os métodos baseados na Lógica Linear para a verificação de requisitos funcionais e não funcionais em SOA, apresentados no Capítulo 3, e também os métodos para detecção e remoção de requisitos negativos do tipo *deadlock* em SOA, apresentados no Capítulo 4, um estudo de caso que considera a verificação de serviços *Web* após a composição é apresentado. Na seção 5.1, é apresentada a relação direta entre a modelagem de processos de *workflow* e a modelagem de serviços *Web*. Na seção 5.2, é apresentada a verificação de requisitos funcionais, e na seção 5.3, a verificação de requisitos não funcionais de desempenho em serviços *Web* após a composição. Já na seção 5.4 é realizada a detecção e remoção de requisitos negativos do tipo *deadlock* em serviços *Web* após a composição. Por fim, uma simulação do estudo de caso é apresentada na seção 5.5.

5.1 Módulos de *Workflow*

Em geral, um serviço *Web* é visto como uma aplicação acessível a outras aplicações da *Web* (NGHIEM, 2002). Pode ser definido como um aplicativo modular independente e autoexplicativo que pode ser publicado, localizado e invocado em uma rede, geralmente na Internet (ALONSO et al., 2004).

De acordo com (MARTENS, 2005), um serviço *Web* pode ser modelado com a ajuda de uma rede Petri acíclica chamada de módulo de *workflow*, conforme definição apresentada na sequência.

Definição 5.1.1 *Um módulo de workflow é uma rede de Petri* $M = (P, T, F)$*, de modo que:*

1. *O conjunto de lugares é dividido em três conjuntos disjuntos: lugares internos* P^N*, lugares de entrada* P^I *e lugares de saída* P^O*.*
2. *A relação de fluxo é dividida em fluxo interno* $F^N \subseteq (P^N \times T) \cup (T \times P^N)$ *e fluxo de comunicação* $F^C \subseteq (P^I \times T) \cup (T \times P^O)$*.*
3. *A rede* $PM = (P^N, T, F^N)$ *é uma WorkFlow net.*
4. *Nenhuma transição é conectada ao mesmo tempo a um lugar de entrada e a um lugar de saída.*

Para ilustrar a abordagem apresentada nesta seção, considere o exemplo de um sistema para reserva de passagens aéreas (ATRS) apresentado em (VALERO et al., 2012). O sistema ATRS foi descrito em (VALERO et al., 2012) na forma de *Web Services Choreography Description Language* (WS-CDL) conforme mostra a Figura 50. As especificações

WS-CDL são contratos que contêm definições globais das condições comuns de pedidos e restrições sob as quais as mensagens são trocadas.

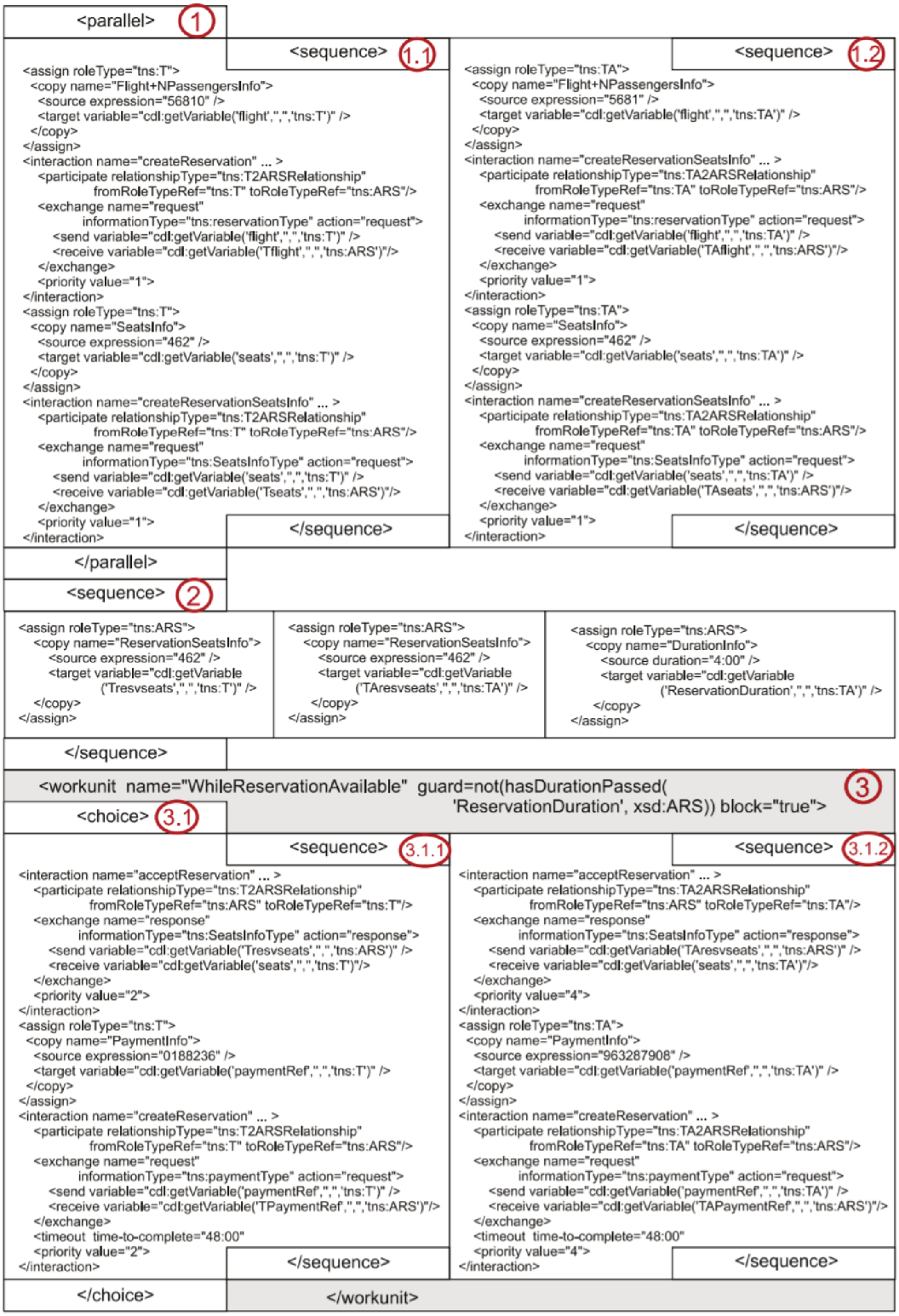


Figura 50 – Sistema para reserva de passagens aéreas representado em WS-CDL (VALERO et al., 2012).

O sistema ATRS apresentado na Figura 50 possui três tipos de funções: Viajante (T),

Agente de Viagens (A), e Sistema de Reserva de Companhias Aéreas (R). O sistema R recebe uma solicitação de viagem para uma data e um voo específico (para simplificar, presumi-se que haja assentos livres) sendo que as solicitações podem ser enviadas por um viajante ou por um agente de viagem. Os viajantes possuem prioridade mais alta, ou seja, em caso de conflito as solicitações dos viajantes são atendidas primeiramente. As reservas são válidas apenas por um período de 48 horas, ou seja, se não forem confirmadas e pagas em dois dias, serão canceladas e as vagas pré-reservadas serão liberadas. A Figura 50 contém as partes relevantes de um documento WS-CDL que descreve esse sistema. O documento contém três seções numeradas que correspondem a uma estrutura paralela (T e A solicitam as informações do assento em paralelo), uma estrutura de sequência (as informações de reserva do assento são definidas) e uma estrutura de unidade de trabalho para atrasar a execução por 4 horas. Essa unidade de trabalho (número 3 na figura) consiste em uma escolha, cujas primeiras atividades são as interações de reserva de T e A que possuem prioridades diferentes (viajantes possuem prioridade mais alta). A atividade final de ambas as filiais corresponde ao pagamento, a qual possui um tempo limite associado.

Para representar o sistema ATRS em módulos de *workflow*, conforme a Definição 5.1.1, uma adaptação foi realizada. Isso significa que somente um módulo cliente é utilizado para representar tanto o viajante quanto o agente de viagens. A Figura 51 mostra a formalização dos principais serviços definidos na Figura 50 utilizando os conceitos de módulos de *workflow* para representar serviços *Web*. O módulo *Costumer* representa o cliente e o módulo *AirlineReservationSystem* representa o serviço de reserva realizado pela companhia aérea. No módulo *Costumer*, os lugares internos são iC, $C1$, $C2$ e oC, o lugar $P4$ é lugar de entrada, e os lugares $P1$, $P2$ e $P3$ são lugares de saída. No módulo *AirlineReservationSystem*, os lugares internos são iS, $S1$, $S2$ e oS, os lugares de entrada são $P1$, $P2$ e $P3$, e o lugar de saída é $P4$. No módulo *Costumer*, o cliente envia uma solicitação de reserva para a companhia aérea (t_1) e, em seguida, o pagamento deve ser realizado. Se o pagamento não for realizado (t_4), a reserva é cancelada, mas caso o pagamento seja realizado (t_3), o cliente receberá a confirmação da reserva (t_8). No módulo *AirlineReservationSystem*, a companhia aérea recebe um pedido de reserva (t_2) e, caso receba o pagamento (t_5), envia a confirmação da reserva (t_7); caso contrário, a reserva é cancelada (t_6).

Um serviço *Web* pode ser implementado invocando outros serviços *Web*, possivelmente fornecidos por diferentes empresas. Um serviço *Web* implementado chamando outros serviços *Web* é chamado de serviço composto (ALONSO et al., 2004). Quando dois módulos distintos compõem um serviço *Web*, seus locais comuns são mesclados e os locais de entrada e saída se tornam a nova interface (MARTENS, 2005). Para obter um módulo de *workflow* sintaticamente correto, é necessário adicionar novos componentes para inicialização e finalização, definindo assim a noção de sistema composto (MARTENS, 2005).

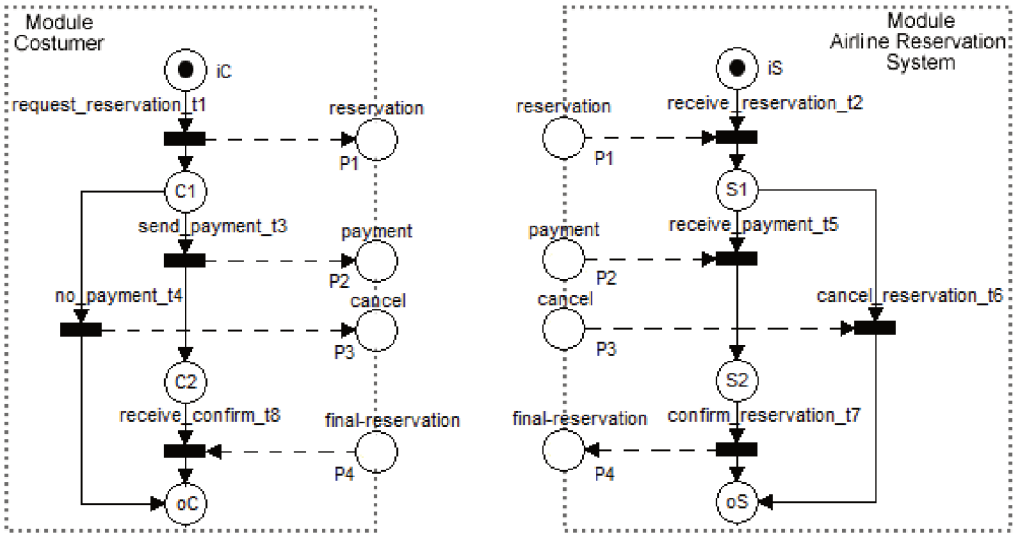


Figura 51 – Módulos de *workflow*.

Definição 5.1.2 *Um sistema composto* $A \oplus B$ *é dado por* (P_s, T_s, F_s), *tal que* $P_s = P_a \cup P_b \cup \{i, o\}, T_s = T_a \cup T_b \cup \{t_i, t_o\}$ *e* $F_s = F_a \cup F_b \{(i, t_i), (t_i, \alpha_a), (t_i, a_b), (\omega_a, t_o), (\omega_b, t_o), (t_o, o)\}$, *onde:*

1. $A = (P_a, T_a, F_a)$ *e* $B = (P_b, T_b, F_b)$ *são dois módulos compatíveis sintaticamente (ambos processos internos são disjuntos e cada lugar comum é um lugar de saída de um módulo e lugar de entrada do outro).*

2. $i, o \notin (P_a \cup P_b)$ *são dois novos lugares e* $t_i, t_o \notin (T_a \cup T_b)$ *são duas novas transições.*

A Figura 52 mostra o sistema composto $Costumer \oplus AirlineReservationSystem$. Os dois módulos são considerados como sintaticamente compatíveis, pois os processos internos são disjuntos e cada lugar comum é um lugar de saída de um módulo e um lugar de entrada do outro.

Verificando a definição de módulo de *workflow*, apresentada por (MARTENS, 2005), e a definição de IOWF-net, apresentada por (AALST, 1998b), observa-se que um módulo de *workflow* corresponde a uma *WorkFlow* net Local (LWF-net) adicionada de sua estrutura de comunicação (lugares e fluxos de comunicação). Já a definição de sistema composto é equivalente à definição de uma U(IOWF-net). Desse modo, os métodos apresentados neste trabalho podem ser diretamente aplicados nos modelos de serviços *Web* representados por módulos de *workflow*, pois estes métodos utilizam IOWF-net e U(IOWF-net) na representação dos modelos.

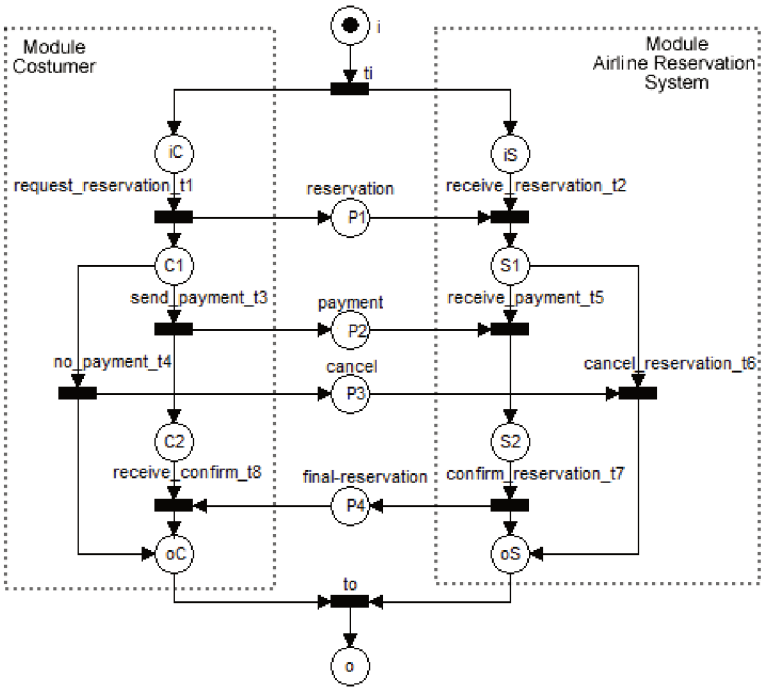


Figura 52 – Módulos de *workflow* compostos ($Costumer \oplus AirlineReservationSystem$).

5.2 Verificação de requisitos funcionais em serviços *Web* compostos

Esta seção apresenta a aplicação do método de verificação de requisitos funcionais em SOA, apresentado na Seção 3.2, no contexto de serviços *Web* compostos.

O modelo de requisitos é considerado como um contrato de serviços *Web* que contém os recursos adequados para a chamada de um consumidor externo. O contrato de serviços *Web* especifica os requisitos de sistema esperados que as partes envolvidas terão que executar; portanto, ele contém apenas as tarefas que são do interesse de todas as partes. O exemplo de um sistema para reserva de passagens aéreas, apresentado na Figura 51, será considerado para ilustrar a aplicação do método. Na Figura 51, é mostrado o contrato de serviço entre as partes envolvidas, ou seja, o modelo de requisitos. Mas para a análise do modelo, é necessário considerar a versão composta, conforme mostra a Figura 52.

A partir da composição, é possível identificar os cenários que as partes envolvidas no processo terão que executar. Nesse contexto, um cenário corresponde a uma rota bem definida mapeada no correspondente módulo de *workflow* e, se o módulo de *workflow* tiver mais de uma rota (locais com dois ou mais arcos de saída), mais de um cenário deverá ser considerado. A abordagem apresentada neste trabalho considera que o contrato de serviço *Web* composto é *sound*. Essa afirmação não significa necessariamente que o modelo de arquitetural correspondente também será *sound* e livre de *deadlocks*.

Um modelo de arquitetural geralmente contém várias tarefas que são somente de inter-

esse local e que não aparecem no contrato de serviço, ou seja, são tarefas privadas. Nesta abordagem, a arquitetura também é modelada por módulos de *workflow*; no entanto, ele contém as tarefas detalhadas internamente de cada módulo de *workflow*. Portanto,a arquitetura corresponde aos serviços *Web* privados. A Figura 53 mostra os módulos de *workflow* privados correspondentes ao contrato de serviços *Web* apresentado na Figura 51. Já a Figura 54 mostra a composição dos módulos de *workflow* dos serviços *Web* privados.

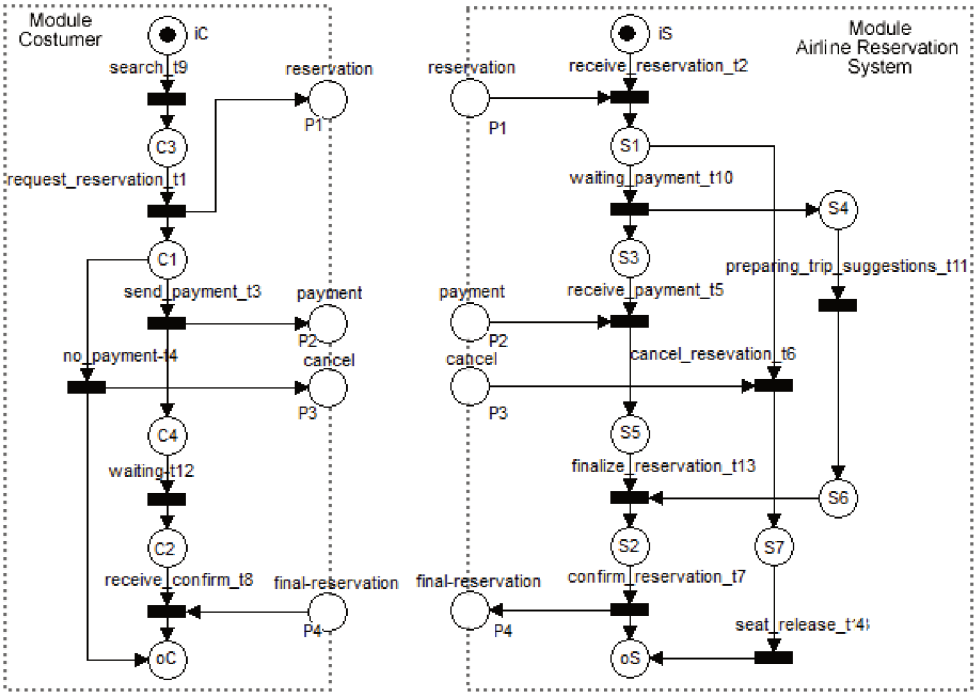


Figura 53 – Serviços *Web* privados (arquitetura).

Como os modelos de requisitos e de arquitetura são analisados baseados na construção das árvores de prova da Lógica Linear, é necessário representar o modelo de contrato de serviços *Web* e o modelo de serviços *Web* privados através das fórmulas da Lógica Linear, lembrando que esses modelos sempre serão considerados nas versões compostas. Desse modo, as transições (ações) do contrato de serviços *Web*, apresentado na Figura 52, são representadas pelas seguintes fórmulas da Lógica Linear:

$t_i = i \multimap iC \otimes iS$,

request-reservation-$t_1 = t_1 = iC \multimap P_1 \otimes C_1$,

receive-reservation-$t_2 = t_2 = iS \otimes P_1 \multimap S_1$,

send-payment-$t_3 = t_3 = C_1 \multimap C_2 \otimes P_2$,

no-payment-$t_4 = t_4 = C_1 \multimap oC$,

receive-payment-$t_5 = t_5 = S_1 \otimes P_2 \multimap S_2$,

cancel-reservation-$t_6 = t_6 = S_1 \otimes P_3 \multimap oC$,

confirm-reservation-$t_7 = t_7 = S_2 \multimap P_4 \otimes oS$,

receive-confirm-$t_8 = t_8 = C_2 \otimes P_4 \multimap oC$,

$t_o = oC \otimes oS \multimap o$.

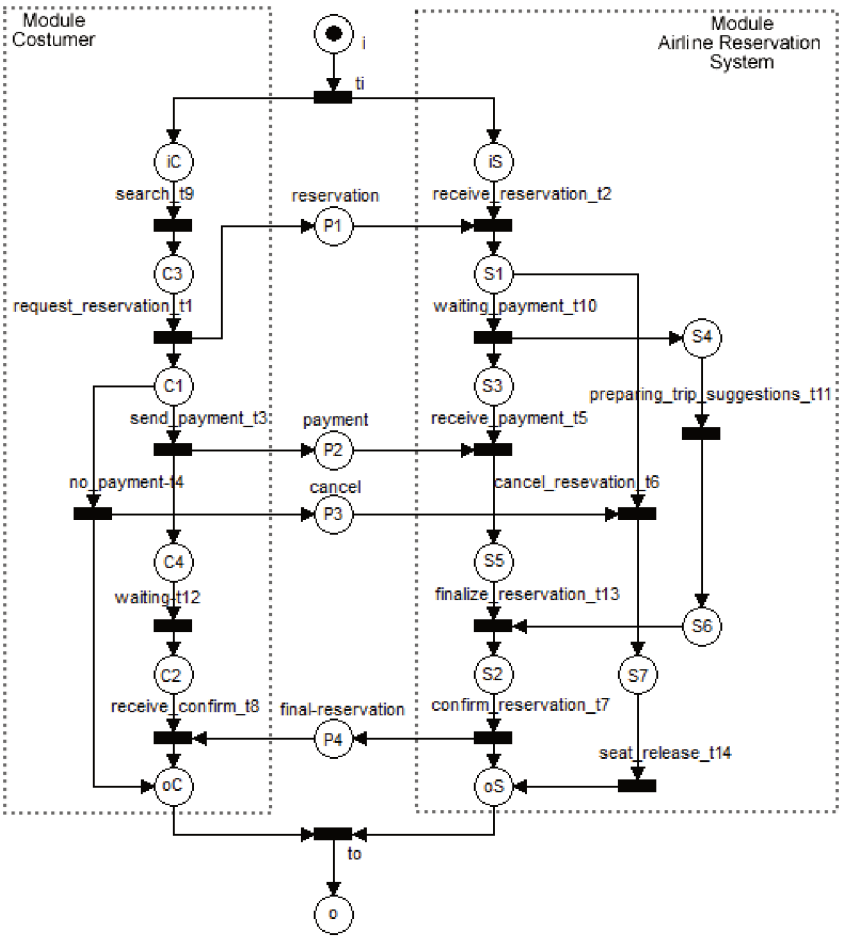


Figura 54 – Composição dos serviços *Web* privados (arquitetura).

As transições (ações) dos serviços *Web* privados, apresentado na Figura 54, são representadas pelas seguintes fórmulas da Lógica Linear:

$t_i = i \multimap iC \otimes iS$,

search-$t_9 = t_9 = iC \multimap C_3$,

request-reservation-$t_1 = t_1 = C_3 \multimap P_1 \otimes C_1$,

receive-reservation-$t_2 = t_2 = iS \otimes P_1 \multimap S_1$,

waiting-payment-$t_{10} = t_{10} = S_1 \multimap S_3 \otimes S_4$,

preparing-trip-suggestions-$t_{11} = t_{11} = S_4 \multimap S_6$,

send-payment-$t_3 = t_3 = C_1 \multimap C_4 \otimes P_2$,

waiting-$t_{12} = t_{12} = C_4 \multimap C_2$,

no-payment-$t_4 = t_4 = C_1 \multimap oC$,

receive-payment-$t_5 = t_5 = S_3 \otimes P_2 \multimap S_5$,

finalize-reservation-$t_{13} = t_{13} = S_5 \otimes S_6 \multimap S_2$,

cancel-reservation-$t_6 = t_6 = S_1 \otimes P_3 \multimap S_7$,

seat-release-$t_{14} = t_{14} = S_7 \multimap oS$,

confirm-reservation-$t_7 = t_7 = S_2 \multimap P_4 \otimes oS$,

receive-confirm-$t_8 = t_8 = C_2 \otimes P_4 \multimap oC$,

$t_o = oC \otimes oS \multimap o$.

Na primeira etapa do método é necessário identificar todos os cenários do modelo de contrato de serviços *Web* por meio do cálculo dos componentes repetitivos estacionários. Para isso, o modelo de contrato deve ser transformado em uma rede de Petri cíclica adicionando uma transição que liga o lugar final ao lugar inicial do modelo. Aplicando a análise de invariantes na ferramenta PIPE, os componentes repetitivos estacionários apresentados na Figura 55 foram encontrados.

Ti	T1	T2	T3	T4	T5	T6	T7	T8	To	T10
1	1	1	0	1	0	1	0	0	1	1
1	1	1	1	0	1	0	1	1	1	1

Figura 55 – Componentes repetitivos estacionários do modelo de contrato de serviços *Web*.

Os vetores apresentados na Figura 55 mostram dois componentes repetitivos estacionários que correspondem a uma lista não ordenada de transições que devem ser disparadas para a rede voltar a sua marcação inicial. Cada componente repetitivo estacionário encontrado para o modelo de contrato de serviços *Web* representará um cenário que deverá ser proposto pela arquitetura correspondente. Portanto, seguindo a etapa 2 do método, os cenários do modelo de contrato de serviços *Web* da Figura 52, chamados respectivamente de Sr3 e Sr4, são representados pelos seguintes sequentes da Lógica Linear:

- Sr3: $i, t_i, t_1, t_2, t_4, t_6, t_o \vdash o$
- Sr4: $i, t_i, t_1, t_2, t_3, t_5, t_7, t_8, t_o \vdash o$

A transição t_{10}, que aparece no cálculo dos componentes repetitivos estacionários, não é considerada nos sequentes, pois foi utilizada apenas para transformar o modelo de contrato de serviços *Web* em uma rede de Petri cíclica. Após a representação dos cenários do modelo de contrato de serviços *Web* em sequentes da Lógica Linear, os mesmos devem ser provados utilizando as árvores de prova da Lógica Linear. As árvores de prova mostradas na sequência são apresentadas de forma resumida, sendo que as árvores de prova completas são mostradas no Apêndice B.1. Para o cenário Sr3, a árvore de prova resumida é:

$$\dfrac{\dfrac{oC \vdash oC \quad oS \vdash oS}{oC, oS \vdash oC \otimes oS}\otimes_R \quad o \vdash o}{\vdots}\multimap_L$$

$$i, i \multimap iC \otimes iS, t_1, t_2, t_4, t_6, t_{14}, t_o \vdash o$$

Como pode ser observado na última linha (lembrando que a leitura é realizada de baixo para cima), a árvore de prova para o cenário Sr3 é corretamente finalizada, pois o sequente identidade $o \vdash o$ ('o' corresponde ao lugar final) é produzido, não há transições para serem provadas e não há átomos para serem consumidos. Consequentemente, o cenário Sr3 é especificado corretamente.

A árvore de prova resumida do cenário Sr4 é apresentada na sequência:

$$\dfrac{\dfrac{oC \vdash oC \quad oS \vdash oS}{oC, oS \vdash oC \otimes oS}\otimes_R \qquad o \vdash o}{\vdots}\multimap_L$$

$$i, i \multimap iC \otimes iS, t_1, t_2, t_3, t_5, t_7, t_8, t_o \vdash o$$

Como pode ser observado na última linha, a árvore de prova do cenário Sr4 é corretamente finalizada. Consequentemente, o cenário Sr4 também é especificado corretamente.

Na etapa 3 do método, deve-se construir os grafos de precedência para as árvores de prova corretamente finalizadas que correspondem aos cenários do modelo de contrato de serviços *Web*. Para gerar o grafo de precedência, as árvores de prova dos respectivos cenários devem ser rotuladas com o disparo das transições. Para o cenário Sr3, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência:

$$\dfrac{\dfrac{\overset{t_4}{oC} \vdash \overset{t_o}{oC} \quad \overset{t_6}{oS} \vdash \overset{t_o}{oS}}{\underset{oC, oS}{\overset{t_4 \; t_6}{\vdash}} \; \underset{oC \otimes oS}{\overset{t_o \; t_o}{}}}\otimes_R \qquad \overset{t_o}{o} \vdash \overset{f_1}{o}}{\vdots}\multimap_L \; t_o$$

$$\overset{i_1}{i}, i \multimap iC \otimes iS, t_1, t_2, t_4, t_6, t_o \vdash \overset{f_1}{o}$$

O grafo de precedência correspondente ao cenário Sr3, gerado a partir da árvore de prova rotulada com as transições de disparo, é apresentado na Figura 56.

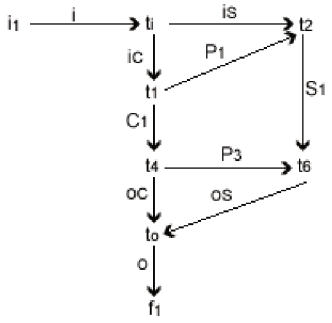


Figura 56 – Grafo de precedência do cenário Sr3.

Para o cenário Sr4, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência:

$$\dfrac{\dfrac{\dfrac{}{oC \vdash oC^{\,t_8}_{\,t_o}} \quad \dfrac{}{oS \vdash oS^{\,t_7}_{\,t_o}}}{oC, oS \vdash oC \otimes oS^{\,t_8,t_7}_{\,t_o,t_o}} \otimes R \quad \dfrac{}{t_o \vdash t_o^{\,f_1}} \multimap L}{i \vdash i^{\,i_1}, i \multimap iC \otimes iS, t_1, t_2, t_3, t_5, t_7, t_8, t_o \vdash o^{\,f_1}}$$

O grafo de precedência correspondente ao cenário Sr4, gerado a partir da árvore de prova rotulada com as transições de disparo, é apresentado na Figura 57.

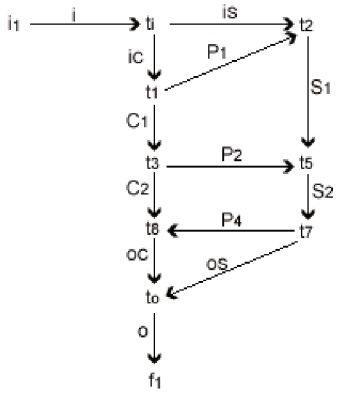


Figura 57 – Grafo de precedência do cenário Sr4.

Na etapa 4 do método, todos os cenários potenciais do modelo de serviços *Web* privados (arquitetura) são identificados por meio do cálculo dos componentes repetitivos estacionários. Aplicando a análise de invariantes da ferramenta PIPE no modelo de serviços *Web* privados da Figura 54, lembrando que para essa etapa da análise o modelo também deve ser transformado em uma rede de Petri cíclica, os componentes repetitivos estacionários apresentados na Figura 58 foram encontrados.

T1	T10	T11	T12	T13	T15	T2	T3	T4	T5	T6	T7	T8	T9	Ti	To	T14
1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0
1	0	0	0	0	1	1	0	1	0	1	0	0	1	1	1	1

Figura 58 – Componentes repetitivos estacionários do modelo de serviços *Web* privados.

Na etapa 5 do método, para cada componente repetitivo estacionário encontrado para o modelo de serviços *Web* privados, deve-se verificar se contém o subconjunto das transições que aparecem nos cenários do contrato de serviços *Web* que foram identificados na etapa 1. Para os dois componentes repetitivos estacionários do modelo de serviços *Web* privados, essa verificação se confirma, ou seja, todas as transições do primeiro vetor da Figura 55 estão presentes no primeiro vetor da Figura 58 e todas as transições do segundo vetor da Figura 55 estão presentes no segundo vetor da Figura 58. Portanto, os cenários

correspondentes aos componentes repetitivos estacionários analisados são potenciais candidatos à equivalência por bissimulação.

Os cenários potenciais do modelo de serviços *Web* privados, encontrados na etapa 5, são chamados respectivamente de Sa5 e Sa6, e, executando a etapa 6 do método, tais cenários potenciais são representados pelos seguintes sequentes da Lógica Linear:

- cenário Sa5: $i, t_i, t_1, t_2, t_4, t_6, t_{14}, t_9, t_o \vdash o$
- cenário Sa6: $i, t_i, t_1, t_2, t_3, t_5, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_o \vdash o$

A transição t_{15}, que aparece no cálculo dos componentes repetitivos estacionários, não é considerada nos sequentes, pois foi utilizada apenas para transformar o modelo de serviços *Web* privados em uma rede de Petri cíclica.

Para o cenário Sa5, a árvore de prova resumida é a seguinte:

$$\dfrac{\dfrac{\dfrac{oC \vdash oC \quad oS \vdash oS}{oC, oS \vdash oC \otimes oS}\otimes_R \quad o \vdash o}{\vdots}\multimap_L}{i, i \multimap iC \otimes iS, t_1, t_2, t_4, t_6, t_{14}, t_9, t_o \vdash o}$$

Na sequência, a árvore de prova resumida do cenário Sa6 é apresentada.

$$\dfrac{\dfrac{\dfrac{oC \vdash oC \quad oS \vdash oS}{oC, oS \vdash oC \otimes oS}\otimes_R \quad o \vdash o}{\vdots}\multimap_L}{i, i \multimap iC \otimes iS, t_1, t_2, t_3, t_5, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_o \vdash o}$$

A última linha das árvores de prova dos cenários Sa5 e Sa6, mostram que estes cenários são corretamente finalizados e, consequentemente, podem ser corretamente executados no modelo arquitetural.

Na etapa 7 do método, deve-se construir os grafos de precedência para as árvores de prova sintaticamente corretas que correspondem aos cenários do modelo de serviços *Web* privados. Para o cenário Sa5, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência.

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{t_4}{oC} \vdash \overset{t_o}{oC} \quad \overset{t_6}{oS} \vdash \overset{t_o}{oS}}{\overset{t_4}{oC}, \overset{t_6}{oS} \vdash \overset{t_o}{oC} \otimes \overset{t_o}{oS}}\otimes_R \quad \overset{t_o}{o} \vdash \overset{f_1}{o}}{\vdots}\multimap_L}{\overset{t_o}{i}, i \multimap iC \otimes iS, t_1, t_2, t_4, t_6, t_{14}, t_9, t_o \vdash \overset{f_1}{o}}$$

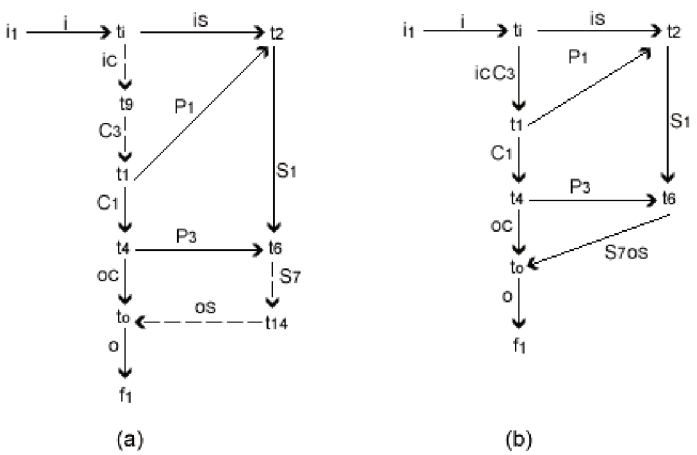


Figura 59 – (a) Grafo de precedência do cenário Sa5. (b) Grafo de precedência do cenário Sa5 simplificado.

O grafo de precedência correspondente ao cenário Sa5 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 59 (a).

A árvore de prova resumida e rotulada com as transições de disparo do cenário Sa6 é a seguinte:

$$\dfrac{\dfrac{\dfrac{}{oC \vdash oC}\ {\scriptstyle t_7,t_o} \qquad \dfrac{}{oS \vdash oS}\ {\scriptstyle t_8,t_o}}{oC, oS \vdash oC \otimes oS}\otimes_R \qquad \dfrac{}{o \vdash o}\ {\scriptstyle t_o,f_1}}{\vdots}\ \multimap_L\ {\scriptstyle t_o}}{i, i \multimap iC \otimes iS, t_1, t_2, t_3, t_5, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_o \vdash o}\ {\scriptstyle i_1,f_1}$$

O grafo de precedência correspondente ao cenário Sa6 gerado a partir da árvore de prova rotulada com as transições de disparo é apresentado na Figura 60 (a).

Na etapa 8 do método, os grafos de precedência correspondentes ao modelo de serviços *Web* privados, identificados na etapa 7, são simplificados retirando do grafo as ações silenciosas, ou seja, ações que são somente de interesse local (lembrando que os arcos que estão ligados as ações silenciosas são representados no grafo de precedência por linhas tracejadas). Ao remover todas as ações silenciosas dos grafos de precedência dos cenários Sa5 e Sa6 mostrados, respectivamente, nas Figuras 59 (a) e 60 (a), os grafos de precedência simplificados das Figuras 59 (b) e 60 (b) são obtidos.

Na etapa 9 do método, deve-se comparar os grafos de precedência do modelo de contrato de serviços *Web* com os grafos de precedência simplificados associados ao modelo de serviços *Web* privados. A razão para isso é verificar se os requisitos funcionais especificados no modelo de requisitos também estão presentes na proposta de arquitetura especificada no modelo de serviços *Web* privados. Portanto, os grafos de precedência obtidos a partir dos cenários Sr3 e Sr4 devem então ser comparados com os grafos de precedência obtidos

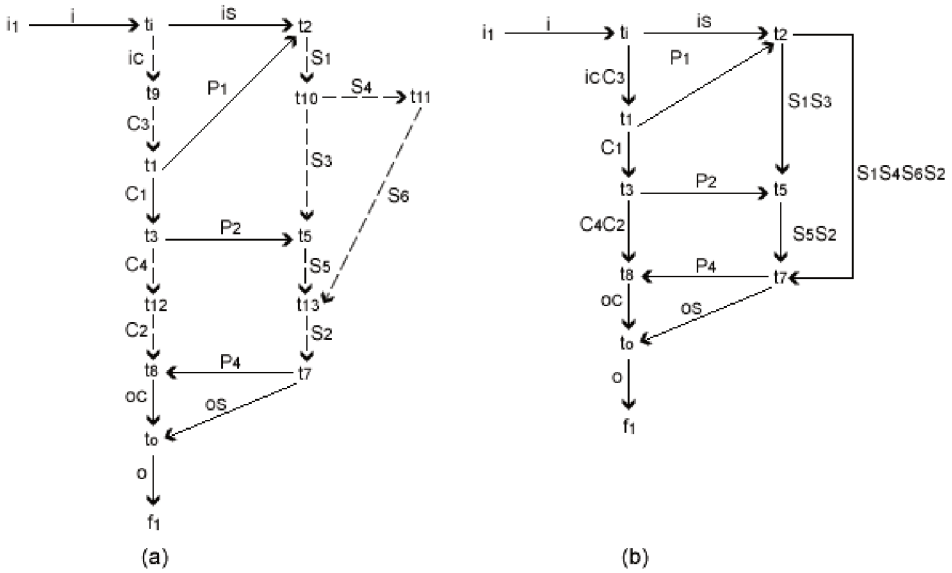


Figura 60 – (a) Grafo de precedência do cenário Sa6. (b) Grafo de precedência do cenário Sa6 simplificado.

a partir dos cenários Sa5 e Sa6 usando a semântica operacional baseada na noção de bissimulação *branching*.

A partir das comparações entre os grafos de precedência, foi encontrado que o grafo de precedência do cenário Sr3 (Figura 56) e o grafo de precedência simplificado do cenário Sa5 (Figura 59 (b)) são equivalentes do ponto de vista da bissimulação *branching* definida para este estudo. O grafo de precedência do cenário Sr4 (Figura 57) e o grafo de precedência simplificado do cenário Sa6 (Figura 60 (b)) também são equivalentes do ponto de vista da bissimulação *branching*. O arco adicional $S_1S_4S_6S_2$ que existe no grafo de precedência simplificado do cenário Sa6, mas que não existe no grafo de precedência do cenário Sr4, é uma restrição redundante, a qual pode ser desconsiderada sem modificar as especificações do modelo de requisitos. Este arco simplesmente especifica que a ação t_2 deve acontecer antes da ação t_7. No entanto, essa afirmação já existe através da sequência de arcos S_1S_3 e S_5S_2.

No modelo de serviços *Web* privados, existe outro cenário, chamado de Sa7, que não foi identificado pelo cálculo dos componentes repetitivos estacionários na etapa 4 do método proposto, pois este cenário leva o modelo a um estado de *deadlock* (este cenário será detalhado na seção 5.4). Embora o modelo de SOA representado por um modelo de serviços *Web* privados (Figura 54) não seja livre de situações de *deadlock* (cenário Sa7), os cenários Sa5 e Sa6 são corretamente executados e verificam os requisitos especificados no modelo de contrato de serviços *Web* (Figura 52). Portanto, mesmo que um modelo de SOA não seja livre de *deadlock* ou possua requisitos de serviços adicionais que não são do interesse do modelo de contrato, é possível verificar se este modelo possui os cenários que corretamente satisfazem as necessidades do negócio definidas no modelo de análise. Para

o exemplo apresentado nessa seção, pode-se concluir que os requisitos definidos no modelo de análise estão bem definidos no modelo de arquitetural. Em particular, os requisitos presentes nos cenários Sr3 e Sr4 do modelo de análise estão presentes, respectivamente, nos cenários Sa5 e Sa6 do modelo de arquitetural.

5.3 Verificação de requisitos não funcionais de desempenho em serviços *Web* compostos

Para a análise de desempenho, conforme o método apresentado na Seção 3.3, serão consideradas explícitas restrições de tempo nos modelos de requisitos e de arquitetura; por isso, será associada a cada transição $t \in T$ do modelo de contrato de serviços *Web* e do modelo de serviços *Web* privados um intervalo $[\theta_{min(t)}, \theta_{max(t)}]$ que corresponde a uma duração de sensibilização da transição. Desse modo, os modelos temporizados representados nas Figuras 61 e 62 são considerados para a verificação do desempenho em serviços *Web* compostos.

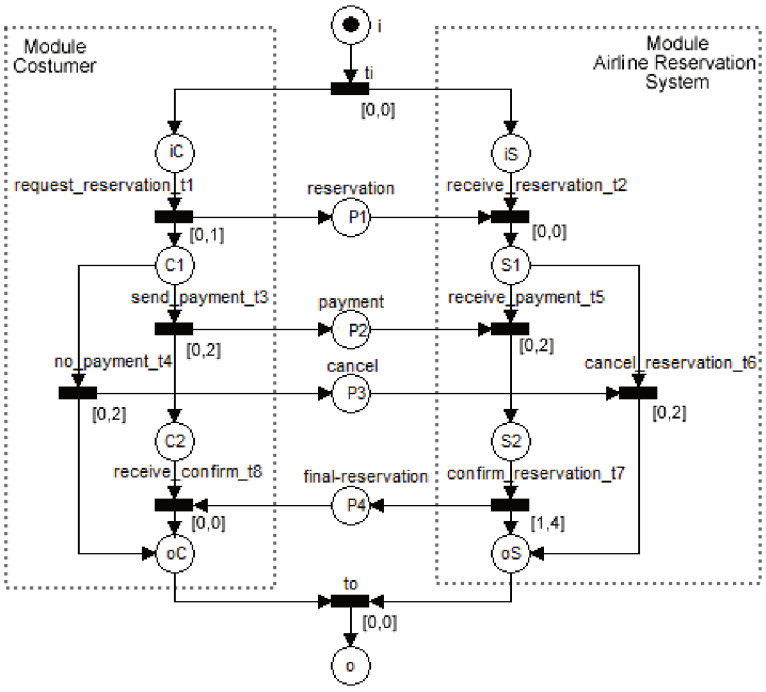


Figura 61 – Modelo de contrato de serviços *Web* temporizado.

Na etapa 1 do método, é necessário identificar os cenários da arquitetura (modelo de serviços *Web* privados) que satisfazem o comportamento dos cenários especificados no modelo de requisitos (modelo de contrato de serviços *Web*). Essa identificação foi realizada na seção 5.2; desse modo, o cenário Sa5 do modelo arquitetural produz um comportamento equivalente, respeitando a noção de bissimulação *branching* definida para este estudo, com

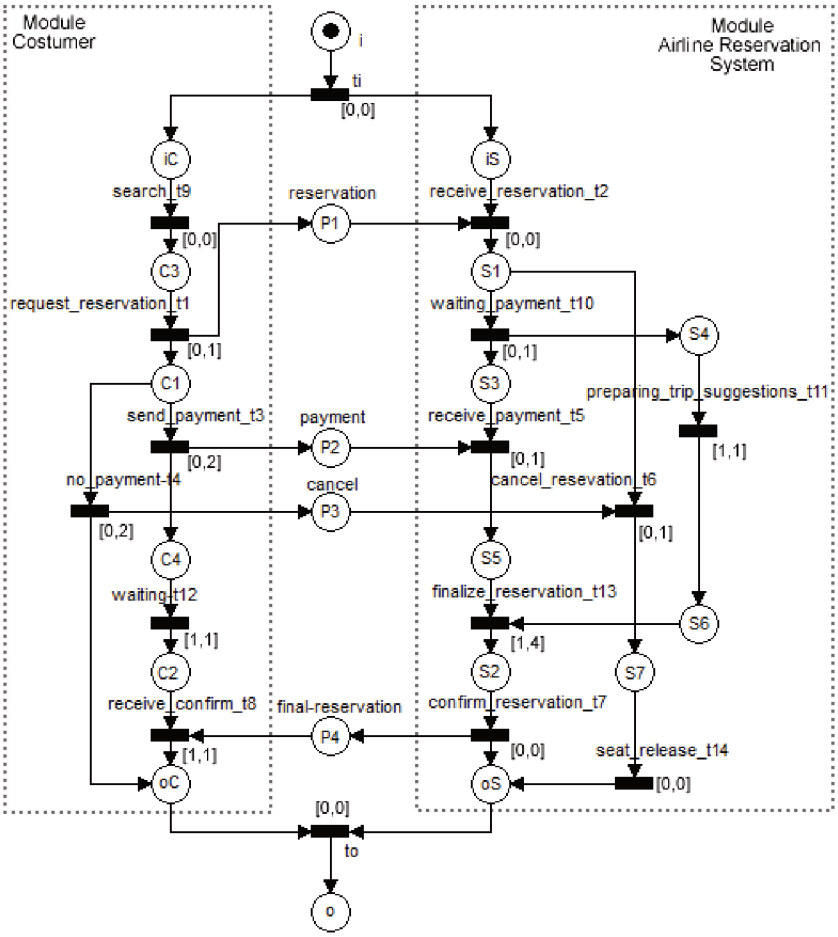


Figura 62 – Modelo de serviços *Web* privados temporizado.

o cenário Sr3 do modelo de requisitos, e o cenário Sa6 do modelo arquitetural produz um comportamento equivalente com o cenário Sr4 do modelo de requisitos.

Na etapa 2 do método, é necessário calcular para cada cenário identificado na etapa 1 (Sr3, Sr4, Sa5 e Sa6) as datas simbólicas de produção e de consumo dos átomos (fichas) envolvidos na execução dos cenários, e, em seguida, obter os intervalos de datas simbólicas de produção dos átomos que finalizam a execução dos cenários. As árvores de prova completas com o cáculo de datas simbólicas e também as datas simbólicas de produção e de consumo representadas em tabelas são mostradas no Apêndice B.2. Os intervalos de datas simbólicas do átomo 'o' (que corresponde a conclusão dos cenários) são mostradas na Tabela 7. Lembrando que tais datas serão usadas para comparar o desempenho entre os cenários do modelo de requisitos e os cenários do modelo arquitetural.

Na etapa 3 do método, é necessário calcular os intervalos de datas numéricas a partir das informações de intervalos de datas simbólicas apresentadas na Tabela 7. Considerando os intervalos de tempo do modelo de contrato de serviços *Web*, mostrada na Figura 61, e do modelo de serviços *Web* privados, mostrada na Figura 62, e o fato que o processo inicia na data 0 ($D_i = 0$), os intervalos de datas numéricas dos átomos que finalizam os cenários Sr3, Sr4, Sa5 e Sa6 são apresentados na Tabela 8.

Átomo 'o'	Intervalo de Datas Simbólicas de Produção
Sr3	$[D_i + d_{imin} + d_{1min} + max(d_{2min}, d_{4min}) + d_{6min} + d_{omin},$ $D_i + d_{imax} + d_{1max} + max(d_{2max}, d_{4max}) + d_{6max} + d_{omax}]$
Sr4	$[D_i + d_{imin} + d_{1min} + max(d_{2min}, d_{3min}) + d_{5min} + d_{7min} + d_{8min} + d_{omin},$ $D_i + d_{imax} + d_{1max} + max(d_{2max}, d_{3max}) + d_{5max} + d_{7max} + d_{8max} + d_{omax}]$
Sa5	$[D_i + d_{imin} + d_{9min} + d_{1min} + max(d_{2min}, d_{4min}) + d_{6min} + d_{14min} + d_{omin},$ $D_i + d_{imax} + d_{9max} + d_{1max} + max(d_{2max}, d_{4max}) + d_{6min} + d_{14max} + d_{omax}]$
Sa6	$[D_i + d_{imin} + d_{9min} + d_{1min} + max(d_{2min} + d_{10min}, d_{3min}) +$ $max(d_{12min}, max(d_{5min}, d_{11min}) + d_{13min} + d_{7min}) + d_{8min} + d_{omin},$ $D_i + d_{imax} + d_{9max} + d_{1max} + max(d_{2max} + d_{10max}, d_{3max}) +$ $max(d_{12max}, max(d_{5max}, d_{11max}) + d_{13max} + d_{7max}) + d_{8max} + d_{omax}]$

Tabela 7 – Intervalos de datas simbólicas de produção do átomo 'o' para os cenários Sr3, Sr4, Sa5 e Sa6.

Átomo 'o'	Intervalos de Datas Numéricas de Produção
Sr3	[0, 5]
Sr4	[2, 9]
Sa5	[0, 4]
Sa6	[3, 9]

Tabela 8 – Intervalos de datas numéricas de produção do átomo 'o' para os cenários Sr3, Sr4, Sa5 e Sa6.

Na etapa 4 do método, deve-se comparar os intervalos de datas númericas dos cenários do modelo de contrato de serviços *Web* com as datas númericas dos cenários do modelo de serviços *Web* privados que são equivalentes em termos de comportamento. Como mostrado na Tabela 9, o intervalo de data numérica [0, 4] do cenário Sa5 pertence ao intervalo de data numérica [0, 5] do cenário Sr3; portanto, é possível concluir que estes cenários, além de serem equivalentes em termos de comportamento, são também equivalentes em termos de desempenho. Os cenários Sa6 e Sr4 também são equivalentes em termos de desempenho, pois o intervalo de data numérica [3, 9] do cenário Sa6 pertence ao intervalo de data numérica [2, 9] do cenário Sr4.

	Cenários	Intervalos de Datas Numéricas de Produção do Átomo 'o'
Cenários equivalentes em termos de comportamento e desempenho	Sr3 e Sa5	[0, 5] e [0, 4]
	Sr4 e Sa6	[2, 9] e [3, 9]

Tabela 9 – Comparação entre os intervalos de datas numéricas para os cenários do estudo de caso.

5.4 Detecção e remoção de requisitos negativos do tipo *deadlock* em serviços *Web* compostos

O modelo de serviços *Web* privados apresentado na Figura 54 possui um cenário, chamado de Sa7, que leva o sistema a um estado de *deadlock*. Observe na Figura 63 que as marcações indesejadas nos lugares oC, P_3, S_3 e S_6 caracteriza o *deadlock*.

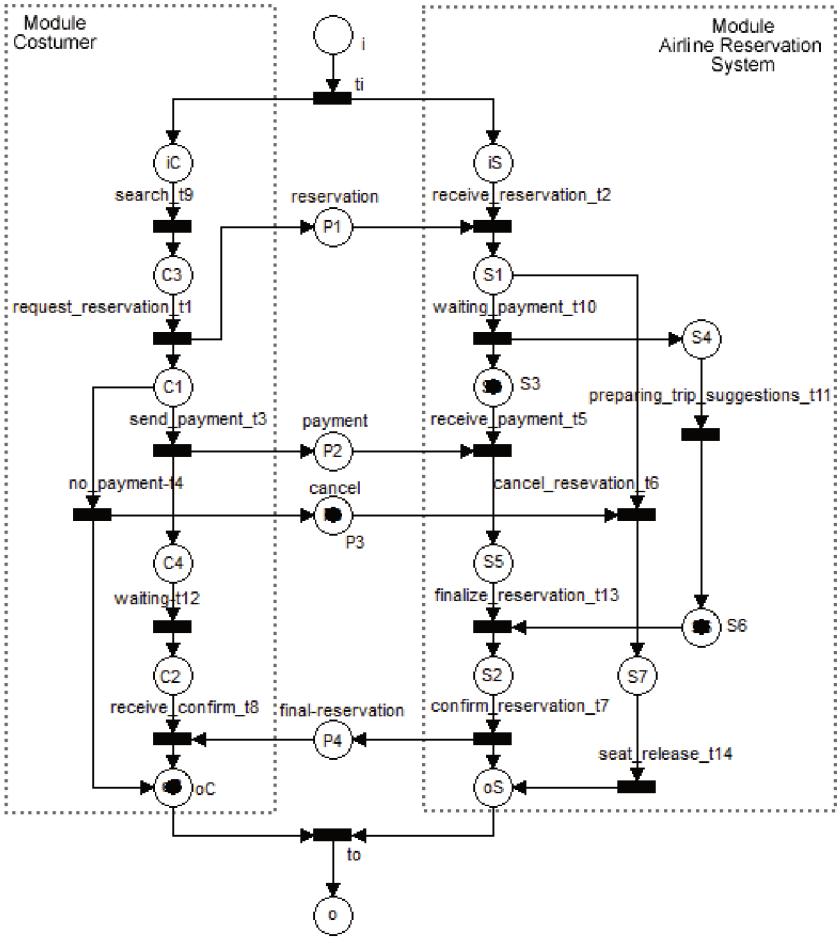


Figura 63 – Modelo de contrato de serviços *Web* privados com marcações indesejadas nos lugares oC, P_3, S_3 e S_6.

Analisando o modelo da Figura 63, observa-se que a marcação nos lugares oC, P_3, S_3 e S_6 é alcançada na tentativa de execução do seguinte cenário:

$$\text{Sa7: } i, t_i, t_9, t_1, t_4, t_2, t_{10}, t_{11}, t_5, t_{13}, t_7, t_o \vdash o.$$

Após a composição dos serviços *Web*, o cenário $Sa7$ é gerado, pois o módulo *Costumer* tenta executar a sequência t_9, t_1, t_4 e o módulo *AirlineReservationSystem* tenta executar a sequência $t_2, t_{10}, t_{11}, t_5, t_{13}, t_7$. No entanto, estas sequências não são compatíveis e, desse modo, a composição de ambas produz a ocorrência do *deadlock*. O cenário Sa7 não é executado corretamente, pois acontece um desvio após a composição dos serviços *Web*.

O fluxo normal do requisito funcional solicitado pelo módulo *Costumer* é desviado para um requisito funcional não compatível no módulo *AirlineReservationSystem*, ou seja, a ação de não pagamento no módulo *Costumer* (transição t_4) é desviada para a ação de aguardar pagamento no módulo *AirlineReservationSystem* (transição t_{10}). Portanto, o cenário Sa7 é considerado um requisito negativo, ou seja, uma sequência de ações que pode levar o sistema a um estado indesejado e, consequentemente, não permitindo a sua correta finalização.

Para formalmente identificar este requisito negativo, o método apresentado na Seção 4.1 é utilizado.

Na primeira etapa do método, o modelo de serviços *Web* privados deve ser representado de forma inversa como mostra a Figura 64.

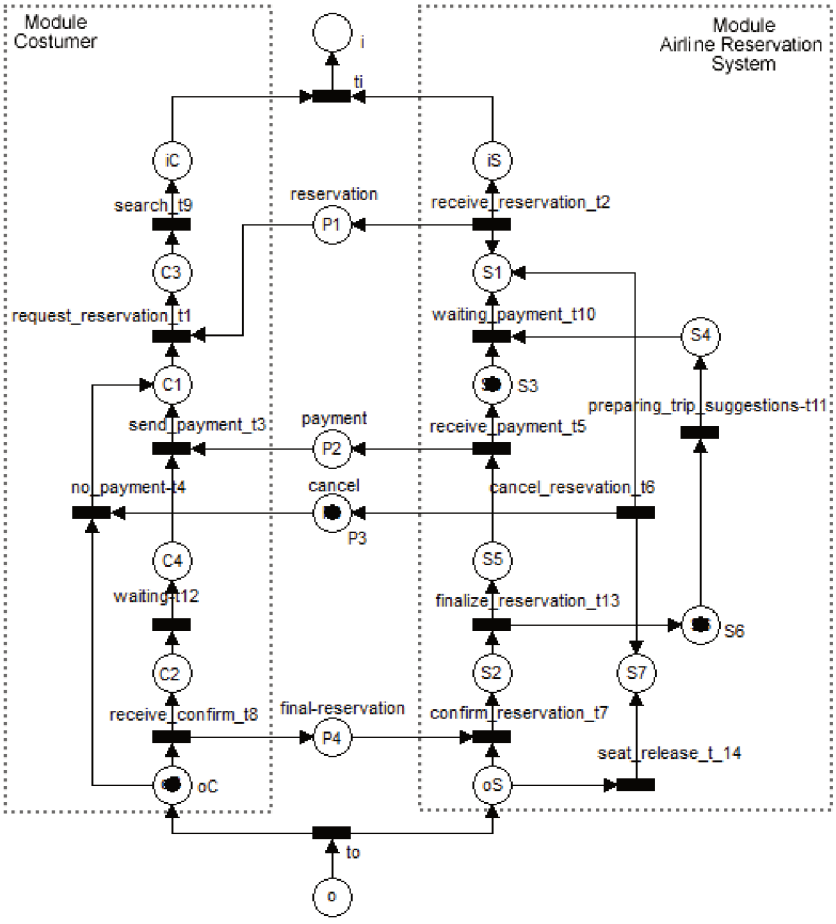


Figura 64 – Modelo inverso dos serviços *Web* privados com marcações nos lugares P_3, oC, S_3 e S_6.

Com a inversão do modelo de serviços *Web* privados, as transições devem ser novamente representadas por fórmulas da Lógica Linear conforme é mostrado na sequência:

$t_i = iC \otimes iS \multimap i$,

search-$t_9 = t_9 = C_3 \multimap iC$,

request-reservation-$t_1 = t_1 = P_1 \otimes C_1 \multimap C_3$,

receive-reservation-$t_2 = t_2 = S_1 \multimap iS \otimes P_1$,
waiting-payment-$t_{10} = t_{10} = S_3 \otimes S_4 \multimap S_1$,
check-reservation-$t_{11} = t_{11} = S_6 \multimap S_4$,
send-payment-$t_3 = t_3 = C_4 \otimes P_2 \multimap C_1$,
waiting-$t_{12} = t_{12} = C_2 \multimap C_4$,
no-payment-$t_4 = t_4 = oC \multimap C_1$,
receive-payment-$t_5 = t_5 = S_5 \multimap S_3 \otimes P_2$,
finalize-reservation-$t_{13} = t_{13} = S_2 \multimap S_5 \otimes S_6$,
cancel-reservation-$t_6 = t_6 = S_7 \multimap S_1 \otimes P_3$,
seat-release-$t_{14} = t_{14} = oS \multimap S_7$,
confirm-reservation-$t_7 = t_7 = P_4 \otimes oS \multimap S_2$,
receive-confirm-$t_8 = t_8 = oC \multimap C_2 \otimes P_4$,
$t_o = o \multimap oC \otimes oS$.

Após a inversão do modelo de serviços *Web* privados, os componentes repetitivos estacionários são calculados para encontrar todos os possíveis cenários que levam à marcação do *deadlock*. Aplicando a análise de invariantes da ferramenta PIPE no modelo inverso de serviços *Web* privados da Figura 64, o componente repetitivo estacionário apresentado na Figura 65 é encontrado. Lembrando que para essa análise, o modelo inverso de serviços *Web* privados foi transformado em uma rede de Petri inversa parcialmente cíclica.

T1	T10	T11	T12	T13	T15	T2	T3	T4	T5	T6	T7	T8	T9	Ti	To	T14
1	1	1	0	0	1	1	0	1	0	0	0	0	1	1	0	0

Figura 65 – Cálculo dos componentes repetitivos estacionários para o modelo inverso de serviços *Web* privados da Figura 64.

O vetor apresentado na Figura 65 mostra uma lista não ordenada de transições que pode levar a um invariante de transição na rede de Petri inversa parcialmente cíclica. Este componente repetitivo estacionário será considerado como um cenário e será chamado de Snr2. Portanto, seguindo a etapa 4 do método, o cenário Snr2 do modelo inverso de serviços *Web* privados é representado pelo seguinte sequente da Lógica Linear:

$$\text{Snr2: } oC, P_3, S_3, S_6, t_1, t_{10}, t_{11}, t_2, t_4, t_9, t_i \vdash i.$$

A árvore de prova completa para o cenário Snr2 é mostrada no Apêndice B.4. Na sequência é apresentada a árvore de prova resumida para este cenário.

$$\dfrac{\dfrac{\dfrac{iS \vdash iS \quad iC \vdash iC}{iS, iC \vdash iS \otimes iC}\otimes_R \quad i \vdash i}{\vdots}\multimap_L}{oC, P_3, S_3, S_6, S_6 \multimap S_4, t_1, t_{10}, t_2, t_4, t_9, t_i \vdash i}$$

Após a prova do sequente, o grafo de precedência é construido a partir da árvore de prova rotulada. Para o cenário Snr2, a árvore de prova resumida e rotulada com as transições de disparo é mostrada na sequência, sendo que a árvore de prova completa também é apresentada no Apêndice B.4.

$$\dfrac{\dfrac{\dfrac{}{iS \overset{t_2}{\vdash} iS}\ t_i \qquad \dfrac{}{iC \overset{t_9}{\vdash} iC}\ t_i}{iS, iC \overset{t_2\ t_9}{\vdash} iS \otimes iC}\ \otimes_R \qquad \dfrac{}{i \overset{t_i}{\vdash} i}\ f_1}{\vdots}\ \multimap_L\ t_i$$

$$oC, P_3, S_3, S_6, S_6 \multimap S_4, t_1, t_{10}, t_2, t_4, t_9, t_i \overset{i_1\ i_2\ i_3\ i_4}{\vdash} i \quad f_1$$

O grafo de precedência correspondente ao cenário Snr2, gerado a partir da árvore de prova rotulada com as transições de disparo, é apresentado na Figura 66. Observando o grafo de precedência é possível verificar a ordem parcial de todas as transições disparadas que levam a um estado de *deadlock* no modelo de serviços *Web* privados, ou seja, o cenário que corresponde ao requisito negativo. Desse modo, a partir das informações apresentadas no grafo de precedência, a sequência de disparos pode ser analisada para verificar o motivo da ocorrência do *deadlock*. Além disso, tais informações também servirão de base para soluções que controlem a ocorrência do *deadlock*.

No estudo de caso analisado, existe apenas um cenário que, a partir da marcação inicial i, leva à marcação dos lugares oC, P_3, S_3 e S_6, conforme é demonstrado pelo cálculo dos componentes repetitivos estacionários.

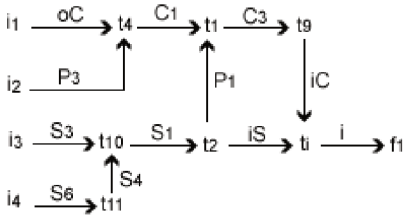


Figura 66 – Grafo de precedência do cenário Srn2.

Na Figura 67 é apresentado o grafo de precedência com a direção dos arcos invertidos, ou seja, a partir do lugar inicial do modelo de serviços *Web* privados em direção as marcações indesejadas. Este grafo deixa claro que as marcações nos lugares oC e P_3 foram em decorrência do disparo da transição t_4, a marcação no lugar S_3 em decorrência do disparo da transição t_{10} e a marcação no lugar S_6 em decorrência do disparo da transição t_{11}.

A partir da indentificação do requisito negativo do tipo *deadlock*, o método que considerada regras de sincronização, apresentado na Seção 4.2, pode ser utilizado para fazer a correção do *deadlock*.

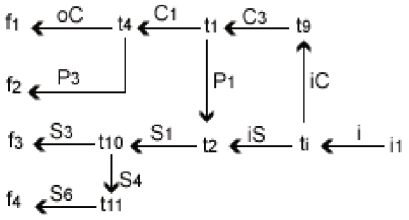


Figura 67 – Grafo de precedência do cenário Srn2 com os arcos invertidos.

A primeira etapa do método para corrigir o *deadlock* é identificar quais cenários de requisitos funcionais foram transformados em um requisito negativo devido a ocorrência do *deadlock*. Como demonstrado logo acima, foi encontrado apenas um cenário, chamado de Srn2, que, a partir da marcação inicial i, leva à marcação dos lugares P_3, oC, S_3 e S_6. Este cenário também pode ser visualizado nos grafos de precedência apresentados nas Figuras 66 e 67.

Na segunda etapa do método, é necessário identificar o lugar cp, ou seja, o lugar de comunicação que possui uma ficha presa. Portanto, basta analisar quais dos lugares que possuem as marcações indesejadas é um lugar de comunicação. No estudo de caso analisado $cp = P_3$.

Na terceira etapa do método, é necessário identificar a transição de entrada t_{d1} do lugar $cp = P_3$. Esta identificação pode ser realizada analisando o modelo de serviços *Web* privados ou analisando o grafo de precedência gerado após a identificação do cenário que causa o *deadlock*. Como, primeiramente, o grafo de precedência é gerado de forma inversa, a transição t_{d1} será aquela que é disparada pela marcação do lugar $cp = P_3$. No estudo de caso analisado $t_{d1} = t_4$.

Na quarta etapa do método, deve-se verificar se há algum lugar com bifurcações que uma vez marcado leva à sensibilização da transição $t_{d1} = t_4$. Para verificar se este lugar existe, pode-se identificar no grafo de precedência da Figura 66 os lugares que são marcados a partir da transição t_{d1} e em seguida verificar se estes lugares possuem bifurcações ou não. Os lugares no grafo de precedência são representados pelos rótulos das arestas. No estudo de caso analisado, o lugar de entrada da transição $t_{d1} = t_4$ é o lugar que possui bifurcações; desse modo, $pa = C_1$ e a transição $t_{d1} = t_4$ será considerada para a sincronização.

Na quinta etapa do método, deve-se identificar qual é a transição de saída t_{d2} do lugar $cp = P_3$. Como a transição t_{d2} não faz parte do cenário que causa o *deadlock*, não é possível identificá-la pelo grafo de precedência; desse modo, é preciso verificar no modelo de serviços *Web* privados qual é a transição de saída do lugar $cp = P_3$; neste caso $t_{d2} = t_6$. Como não há outras transições de saída no lugar $cp = P_3$, não existe outros cenários a serem considerados para sincronização.

Na sexta etapa do método, é necessário identificar o lugar com bifurcações que, uma

vez marcado, leva à sensibilização da transição $t_{d2} = t_6$. Analisando o modelo de serviços *Web* privados, identifica-se que o lugar de entrada ($pb = S_1$) da transição $t_{d2} = t_6$ é o lugar que possui bifurcações; portanto, a transição t_6 será considerada para sincronização.

Na sétima etapa do método, as regras de sincronização são aplicadas de acordo com as informações encontradas nas etapas anteriores. No estudo de caso analisado, a transição $t_{d1} = t_4$ deve ser sincronizada com a transição $t_{d2} = t_6$; assim, de acordo com as regras de sincronização apresentadas na Seção 4.2, o lugar de comunicação P_3 será substituído por um elemento de comunicação síncrona.

Com a aplicação da regra de sincronização, o requisito negativo deixa de existir, pois o cenário que causa o *deadlock* será sempre evitado mediante a condição de disparo simultâneo associada as transições t_4 e t_6. Isso significa que somente os cenários seguros, que no caso são Sa5 e Sa6, serão executados.

A Figura 68 apresenta o modelo de serviços *Web* privados com a correção do *deadlock*.

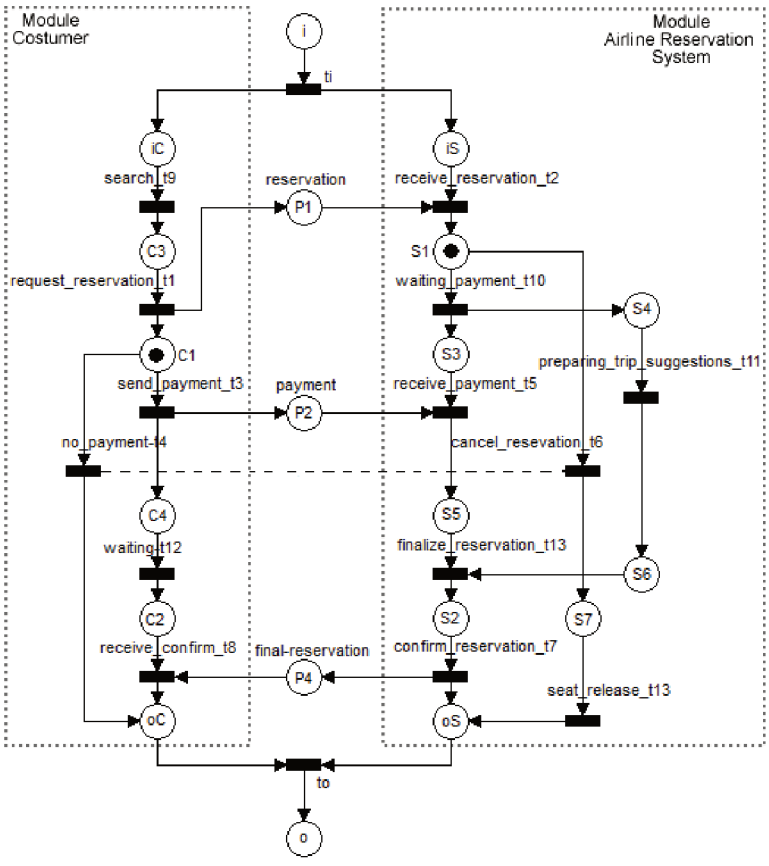


Figura 68 – Modelo de serviços *Web* privados com correção do *deadlock* através de mecanismos de comunicação síncrona.

Com a introdução do mecanismo de sincronização, as transições t_4 e t_6 passam a ser representadas pela seguinte fórmula da Lógica Linear: $S_1 \otimes C_1 \multimap oC \otimes oS$. Com isso, na árvore de prova, somente serão alteradas as linhas onde ocorre o disparo destas transições.

O lugar $pa = C_1$ define a condição que levou à marcação do lugar de comunicação $cp = P_3$; já o lugar $pb = S_1$ define a condição que levou à mudança do fluxo normal

para o fluxo que causa o *deadlock*. Portanto, após a correção do *deadlock*, uma marcação parcial nos lugares C_1 e S_1 estabelece uma condição suficiente para reiniciar o processo sem a necessidade de repetir as outras atividades que já foram executadas corretamente. Os caminhos que permitem seguir o fluxo normal do modelo a partir da marcações dos lugares C_1 e S_1 são os cenários $Sa5$ e $Sa6$ que foram provados na seção 5.2. Nas próximas execuções, a partir da marcação inicial, o requisito negativo não será mais percorrido.

5.5 Validação dos métodos propostos por simulação

Para verificar a efetividade dos métodos propostos para a verificação de requisitos funcionais de serviços e não funcionais de desempenho, e também do método para remoção de requisitos negativos do tipo *deadlock*, um modelo de simulação baseado no simulador CPN Tools (WESTERGAARD, 2013) foi projetado. O CPN Tools permite a modelagem, verificação e simulação dos modelos distribuídos, mostrando claramente a relação entre os diferentes processos através dos lugares de comunicação.

O contrato de serviços *Web* (modelo de requisitos), apresentado na Figura 52, e os serviços *Web* privados (arquitetura), apresentado na Figura 54, foram transformados em modelos de redes de Petri coloridas (CPN - *Coloured Petri Net*) e simulados considerando 50 replicações. Para representar os intervalos de tempo associados às transições que correspondem às atividades do processo, foi escolhida a distribuição uniforme com limites mínimos e máximos. As Figuras 69 e 70 mostram, respectivamente, o contrato de serviços *Web* e os serviços *Web* privados modelados no simulador CPN Tools.

Os lugares *reservation*, *payment*, *cancel* e *final-reservation* foram modelados utilizando os lugares de fusão, disponíveis no simulador CPN Tools, para mostrar a interação entre os diferentes processos ($ModuleCostumer$ e $ModuleAirlineReservationSystem$) através de mecanismos de comunicação assíncrona. Em um conjunto de lugares de fusão, tudo o que acontece em cada lugar do conjunto também acontece a todos os outros lugares do mesmo conjunto. A utilização de lugares de fusão não adiciona nada de fundamentalmente novo nos modelos, pois são funcionalmente idênticos. Se todos os membros de um conjunto de fusão estiverem na mesma página, por exemplo, pode-se substituir o conjunto por um único lugar e conectar a ele todos os arcos conectados aos membros do conjunto. A vantagem de se utilizar os lugares de fusão na simulação é permitir a separação dos processos e também a representação explícita dos mecanismos de comunicação.

Para identificar os cenários executados pelo modelo de requisitos e, posteriormente, verificar se estes cenários também são executados na arquitetura, os arquivos de *log* gerados pela simulaçao no CPN Tools foram utilizados. Os arquivos de *log* mostram a sequência de transições disparadas em cada simulação. As Figuras 71 e 72 mostram, respectivamente, as informações geradas nos registros de *log* das simulações realizadas no modelo de contrato de serviços *Web* e no modelo de serviços *Web* privados.

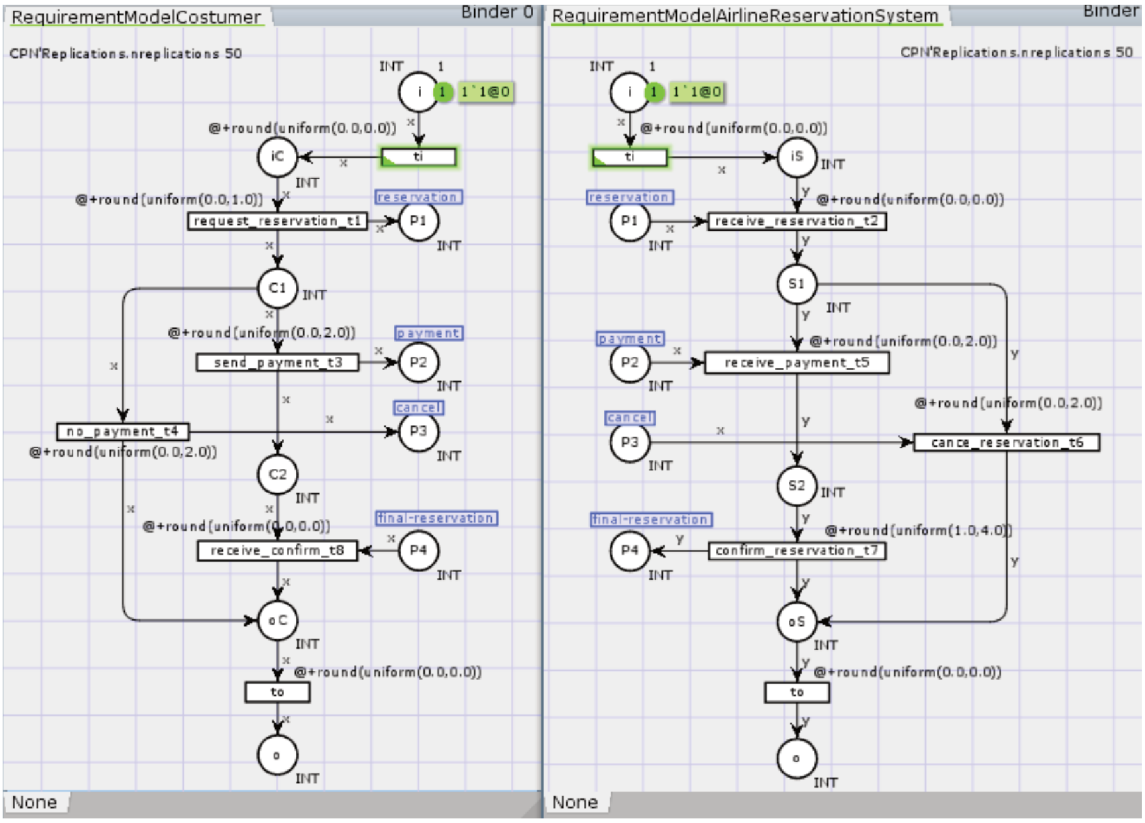


Figura 69 – Contrato de serviços *Web* (modelo de requisitos) modelado no simulador CPN Tools.

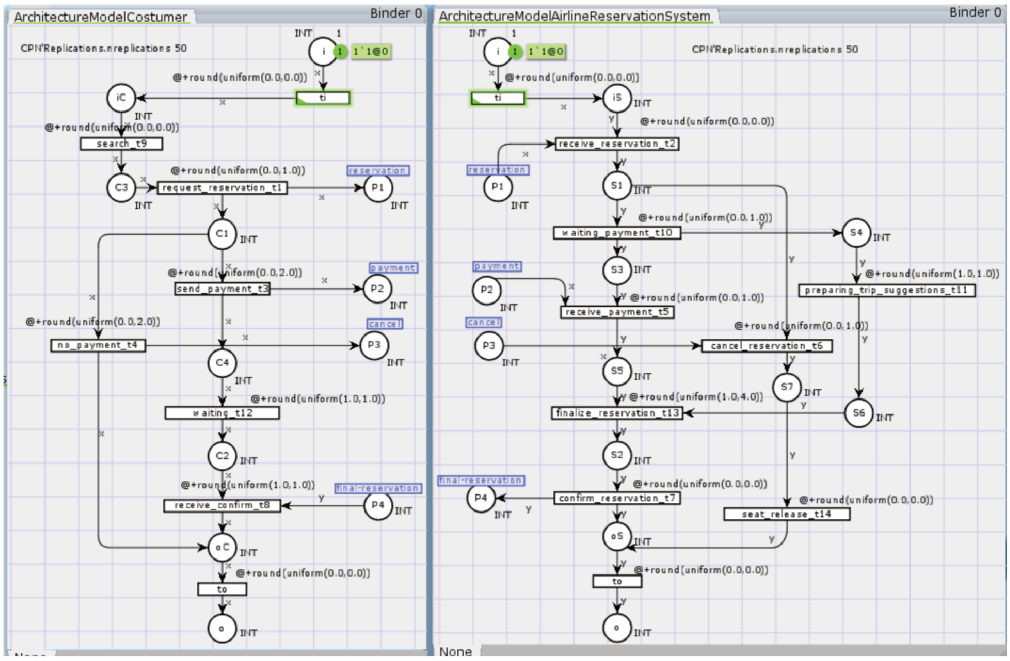


Figura 70 – Serviços *Web* privados (arquitetura) modelados no simulador CPN Tools.

Os registros de *log* apresentados na Figura 71 mostram que as sequências de transições executadas correspondem, respectivamente, aos cenários Sr4 e Sr3 do modelo de requisitos

```
CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Case Study/CaseStudyRequirementModel.cpn
Report generated: Mon Jun 29 16:14:12 2020

1       0       ti @ (1:ArchitecturalModelAirlineReservationSystem)
2       0       ti @ (1:RequirementModelCostumer)
3       0       request_reservation_t1 @ (1:RequirementModelCostumer)
4       1       receive_reservation_t2 @ (1:ArchitecturalModelAirlineReservationSystem)
5       1       send_payment_t3 @ (1:RequirementModelCostumer)
6       2       receive_payment_t5 @ (1:ArchitecturalModelAirlineReservationSystem)
7       3       confirm_reservation_t7 @ (1:ArchitecturalModelAirlineReservationSystem)
8       6       receive_confirm_t8 @ (1:RequirementModelCostumer)
9       6       to @ (1:ArchitecturalModelAirlineReservationSystem)
10      6       to @ (1:RequirementModelCostumer)



CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Case Study/CaseStudyRequirementModel.cpn
Report generated: Mon Jun 29 16:17:50 2020

1       0       ti @ (1:ArchitecturalModelAirlineReservationSystem)
2       0       ti @ (1:RequirementModelCostumer)
3       0       request_reservation_t1 @ (1:RequirementModelCostumer)
4       0       receive_reservation_t2 @ (1:ArchitecturalModelAirlineReservationSystem)
5       0       no_payment_t4 @ (1:RequirementModelCostumer)
6       0       cance_reservation_t6 @ (1:ArchitecturalModelAirlineReservationSystem)
7       0       to @ (1:RequirementModelCostumer)
8       1       to @ (1:ArchitecturalModelAirlineReservationSystem)]
```

Figura 71 – Registro de *log* do modelo de contrato de serviços *Web*.

```
CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Case Study/CaseStudyArchitectureModel.cpn
Report generated: Mon Jun 29 16:28:38 2020

1       0       ti @ (1:ArchitectureModelCostumer)
2       0       ti @ (1:ArchitectureModelAirlineReservationSystem)
3       0       search_t9 @ (1:ArchitectureModelCostumer)
4       0       request_reservation_t1 @ (1:ArchitectureModelCostumer)
5       0       receive_reservation_t2 @ (1:ArchitectureModelAirlineReservationSystem)
6       0       send_payment_t3 @ (1:ArchitectureModelCostumer)
7       0       waiting_payment_t10 @ (1:ArchitectureModelAirlineReservationSystem)
8       0       waiting_t12 @ (1:ArchitectureModelCostumer)
9       1       receive_payment_t5 @ (1:ArchitectureModelAirlineReservationSystem)
10      1       preparing_trip_suggestions_t11 @ (1:ArchitectureModelAirlineReservationSystem)
11      2       finalize_reservation_t13 @ (1:ArchitectureModelAirlineReservationSystem)
12      6       confirm_reservation_t7 @ (1:ArchitectureModelAirlineReservationSystem)
13      6       receive_confirm_t8 @ (1:ArchitectureModelCostumer)
14      6       to @ (1:ArchitectureModelAirlineReservationSystem)
15      7       to @ (1:ArchitectureModelCostumer)



CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Case Study/CaseStudyArchitectureModel.cpn
Report generated: Mon Jun 29 16:38:39 2020

1       0       ti @ (1:ArchitectureModelCostumer)
2       0       ti @ (1:ArchitectureModelAirlineReservationSystem)
3       0       search_t9 @ (1:ArchitectureModelCostumer)
4       0       request_reservation_t1 @ (1:ArchitectureModelCostumer)
5       1       receive_reservation_t2 @ (1:ArchitectureModelAirlineReservationSystem)
6       1       no_payment_t4 @ (1:ArchitectureModelCostumer)
7       1       cancel_reservation_t6 @ (1:ArchitectureModelAirlineReservationSystem)
8       1       seat_release_t14 @ (1:ArchitectureModelAirlineReservationSystem)
9       2       to @ (1:ArchitectureModelCostumer)
10      2       to @ (1:ArchitectureModelAirlineReservationSystem)
```

Figura 72 – Registro de *log* do modelo de serviços *Web* privados.

que são representados pelos de grafos de precedência das Figuras 57 e 56. Já os registros de *log* apresentados na Figura 72 mostram que as sequências de transições executadas correspondem, respectivamente, aos cenários Sa6 e Sa5 do modelo arquitetural que são representados pelos grafos de precedência das Figuras 60 e 59. Comparando os registros

de *log*, observa-se que as sequências de atividades geradas para o modelo de requisitos, ou seja, os cenários encontrados, se confirmam nos registros de *log* do modelo arquitetural, demonstrando assim a equivalência comportamental entre os modelos.

Para verificar o requisito não funcional de desempenho, os cenários identificados nas simulações e que correspondem aos cenários Sr3, Sr4, Sa5 e Sa6 foram modelados em janelas distintas no CPN Tools. As Figuras 73 e 74 mostram, respectivamente, os cenários Sr3 e Sr4 do modelo de contrato de serviços *Web* e as Figuras 75 e 76 mostram, respectivamente, os cenários Sa5 e Sa6 do modelo de serviços *Web* privados modelados no CPN Tools.

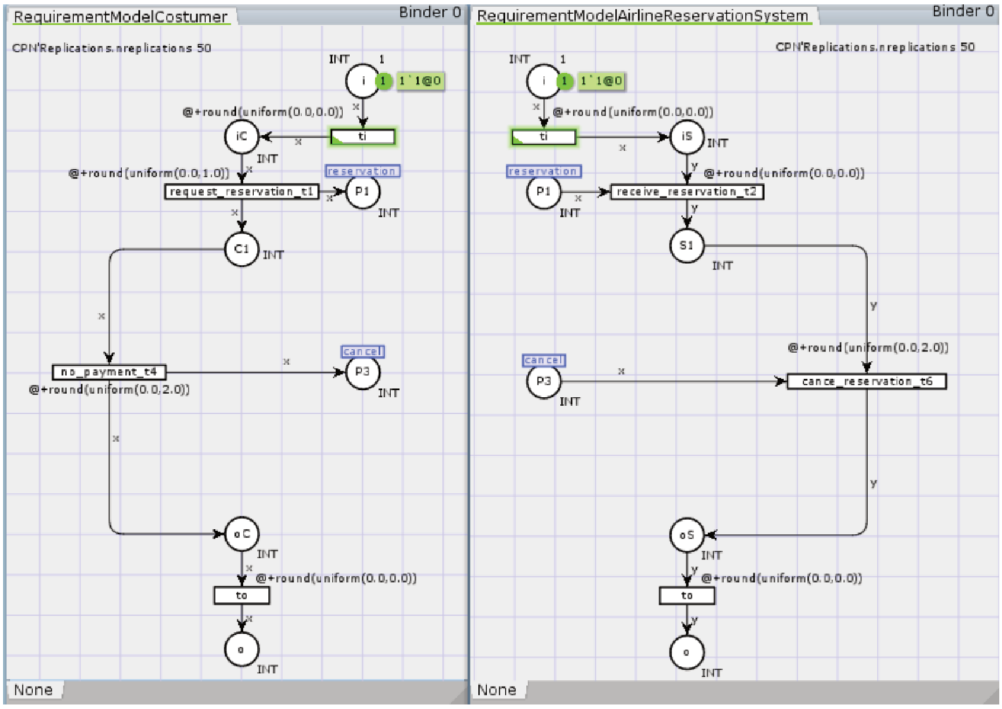


Figura 73 – Cenário Sr3 do modelo de contrato de serviços *Web* (modelo de requisitos) modelado no simulador CPN Tools.

Como no método proposto o requisito não funcional de desempenho é verificado em relação ao intervalo de data que finaliza a execução de um cenário de requisito de serviço, um monitor foi adicionado ao lugar 'o' que representa o lugar final dos modelos. A simulação foi aplicada para cada cenário do modelo de contrato de serviços *Web* e para cada cenário do modelo de serviços *Web* privados.

Os relatórios do CPN Tools com replicações da simulação são mostrados nas Figuras 77, 78, 79 e 80 que correspondem, respectivamente, aos cenários Sr3, Sr4, Sa5 e Sa6.

Considerando o tempo de finalização do cenário Sr3, por exemplo, os resultados estatísticos mostram que o cenário Sr3 finalizará ao mais cedo na data 0 e ao mais tardar na data 5, como pode ser observado na área destacada na Figura 77. O método analítico também produziu para o cenário Sr3 o intervalo de tempo $[0, 5]$, como é mostrado na Tabela 10. Portanto, observa-se que os valores de simulação pertencem ao intervalo numérico

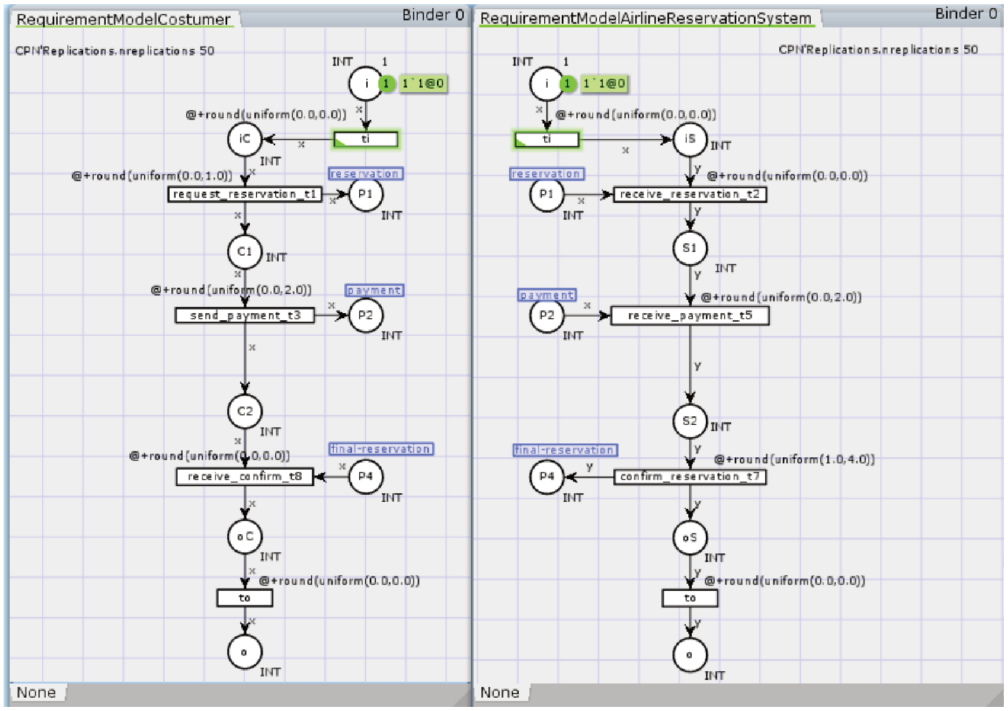


Figura 74 – Cenário Sr4 do modelo de contrato de serviços *Web* (modelo de requisitos) modelado no simulador CPN Tools.

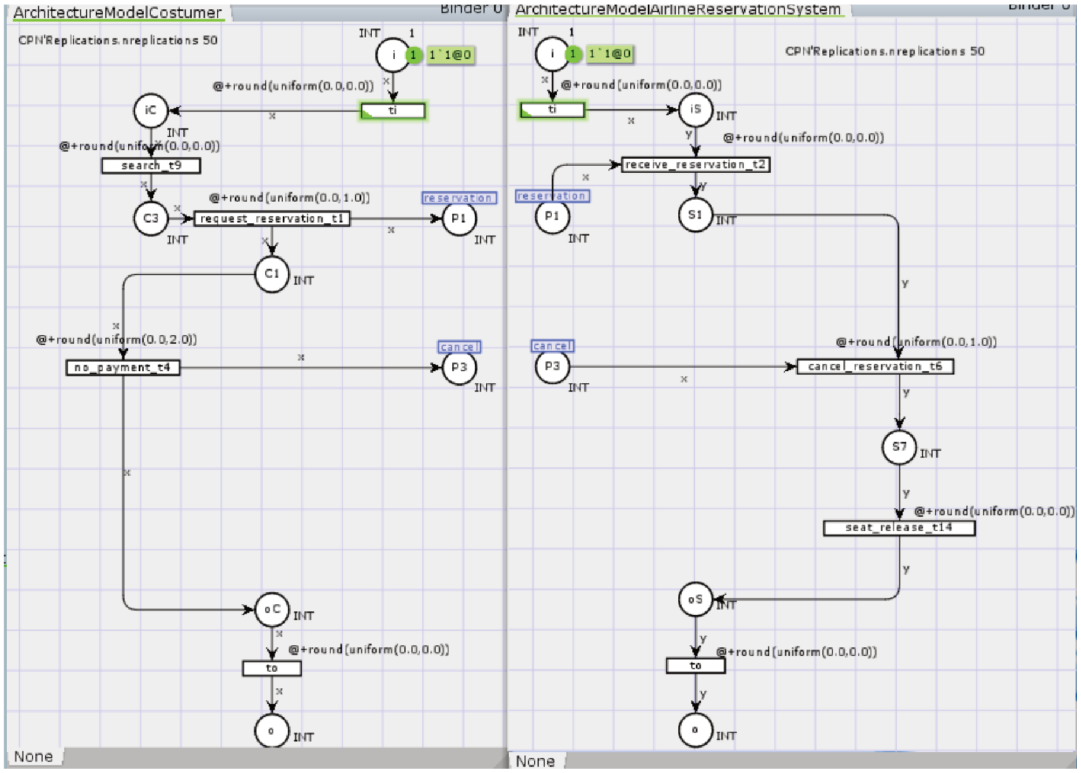


Figura 75 – Cenário Sa5 do modelo de serviços *Web* privados (arquitetura) modelado no simulador CPN Tools.

definido pelo método analítico. Resultados similares foram encontrados para os outros cenários como mostra a Tabela 10, reproduzindo o comportamento do método analítico

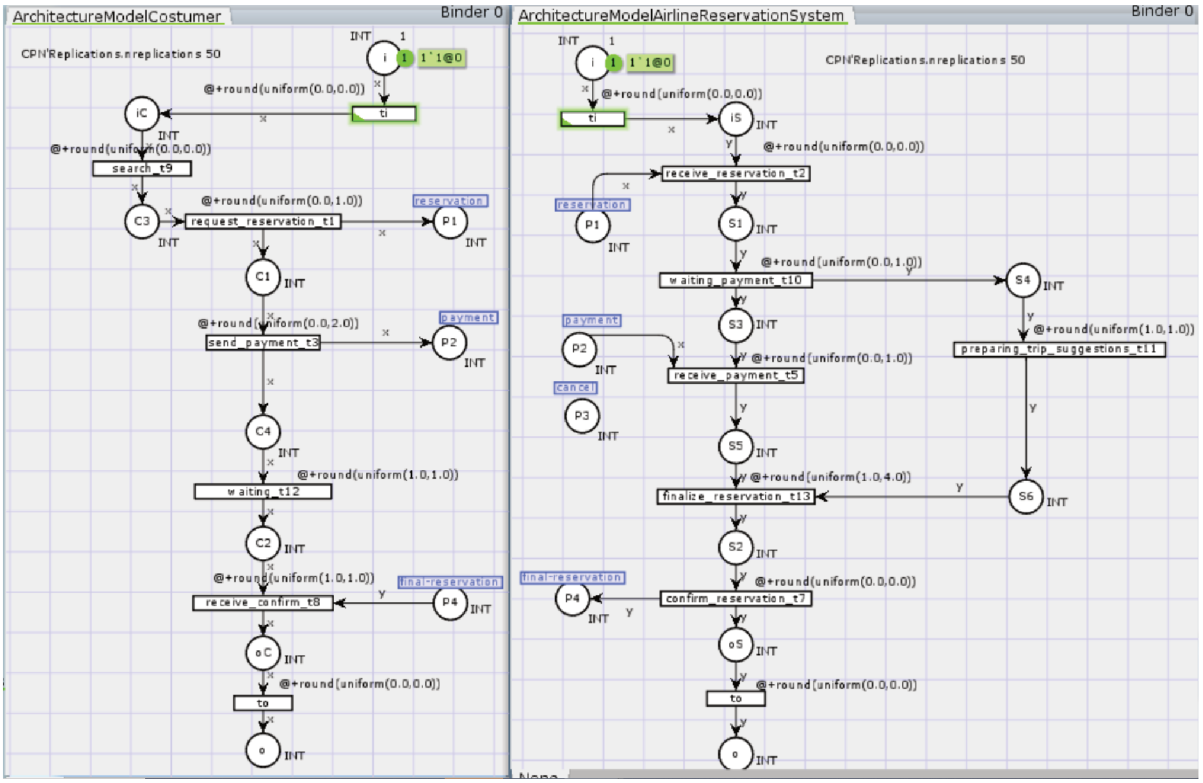


Figura 76 – Cenário Sa6 do modelo de serviços *Web* privados (arquitetura) modelado no simulador CPN Tools.

Statistics							
Name	**Avrg**	**90% Half Length**	**95% Half Length**	**99% Half Length**	**StD**	**Min**	**Max**
oPlaceScenario3							
count_iid	8.000000	0.000000	0.000000	0.000000	0.000000	8	8
max_iid	2.620000	0.271707	0.326081	0.436281	1.140891	0	5
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
sum_iid	6.860000	0.800796	0.961050	1.285838	3.362518	0	13
avrg_iid	0.857500	0.100099	0.120131	0.160730	0.420315	0.000000	1.625000

Figura 77 – Simulação estatística do cenário Sr3.

Statistics							
Name	**Avrg**	**90% Half Length**	**95% Half Length**	**99% Half Length**	**StD**	**Min**	**Max**
oPlaceScenarioSr4							
count_iid	10.000000	0.000000	0.000000	0.000000	0.000000	10	10
max_iid	4.840000	0.327601	0.393161	0.526030	1.375589	2	8
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
sum_iid	19.260000	1.418197	1.702004	2.277199	5.954967	8	32
avrg_iid	1.926000	0.141820	0.170200	0.227720	0.595497	0.800000	3.200000

Figura 78 – Simulação estatística do cenário Sr4.

utilizado para produzir as fórmulas.

Para verificar o método que permite corrigir o *deadlock* causado pela troca de mensagens entre os módulos de *workflow*, os grafos de alcançabilidade e também o relatório

Statistics							
Name	**Avrg**	**90% Half Length**	**95% Half Length**	**99% Half Length**	**StD**	**Min**	**Max**
oPlaceScenarioSa5							
count_iid	10.000000	0.000000	0.000000	0.000000	0.000000	10	10
max_iid	1.980000	0.265676	0.318843	0.426597	1.115567	0	4
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
avrg_iid	0.418333	0.108738	0.130498	0.174601	0.456588	0.000000	1.250000

Figura 79 – Simulação estatística do cenário Sa5.

Statistics							
Name	**Avrg**	**90% Half Length**	**95% Half Length**	**99% Half Length**	**StD**	**Min**	**Max**
oPlaceScenarioSa6							
count_iid	15.000000	0.000000	0.000000	0.000000	0.000000	15	15
max_iid	5.680000	0.305971	0.367201	0.491297	1.284762	3	8
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
avrg_iid	2.029762	0.107354	0.128838	0.172379	0.450778	1.000000	2.857143

Figura 80 – Simulação estatística do cenário Sa6.

Cenários	Intervalos de datas numéricas produzidos a partir das fórmulas	Intervalos de datas numéricas produzidos a partir das simulações
Sr3	$[0, 5]$	$[0, 5]$
Sa5	$[0, 4]$	$[0, 4]$
Sr4	$[2, 9]$	$[2, 8]$
Sa6	$[3, 9]$	$[3, 8]$

Tabela 10 – Comparação entre os intervalos de datas numéricas produzidos a partir das fórmulas e das simulações.

de análise foram gerados. Na Figura 81 é apresentado o grafo de alcançabilidade para o modelo de serviços *Web* privados que foi modelado no CPN Tools conforme mostra a Figura 70. Já a Figura 82 mostra a parte do relatório de análise onde consta a verificação da propriedade de Vivacidade. De acordo com o relatório, a marcação 18 é uma marcação morta, o que confirma a presença de *deadlock* no modelo analisado. Como pode ser observado na Figura 81, a marcação 18 do grafo de alcançabilidade mostra que há fichas presas nos lugares P_3, o_C, S_3 e S_6.

No CPN Tools não é possível representar um conjunto de transições de fusão, ou seja, um conjunto de transições que disparam ao mesmo tempo. Por isso, conforme mostra a Figura 83, que apresenta o modelo de serviços *Web* privados com a correção do *deadlock*, para o exemplo considerado, foi necessário criar um lugar de comunicação auxiliar para receber a informação de qual transição (t_3 ou t_4) é disparada no módulo *Costumer*. Já no módulo *Airline Reservation System* foi necessário inserir guardas nas transições t_{10} e t_6, ou seja, a transição t_{10} somente é disparada caso a transição t_3 tenha sido disparada anteriormente; já a trasição t_6 somente é disparada caso a transição t_4 tenha sido disparada anteriormente. O lugar de comunicação auxiliar, que permite a troca de mensagens entre

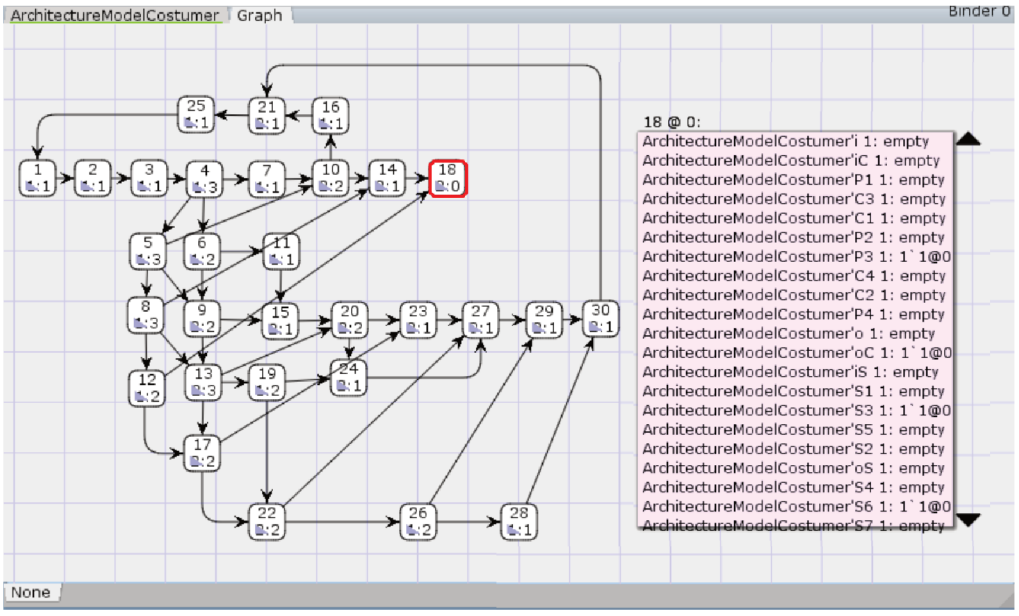


Figura 81 – Grafo de alcançabilidade gerado no CPN Tools para o modelo de serviços *Web* privados com *deadlock*.

```
Liveness Properties
----------------------------------------------------------

 Dead Markings
    [18]

 Dead Transition Instances
    None

 Live Transition Instances
    None
```

Figura 82 – Relatório de análise gerado no CPN Tools para o modelo de serviços *Web* privados antes da correção do *deadlock*.

as partes envolvidas no processo, juntamente com as guardas nas transições simulam a sincronização das transições t_3 e t_{10} e das transições t_4 e t_6. Desse modo, os processos distintos podem ser mantidos em janelas separadas e o modelo de simulação pode ser visto como um protótipo de uma proposta de arquitetura distribuída para correção de *deadlocks* em serviços *Web*.

Como pode ser observado na Figura 83, após a correção do *deadlock*, o processo pode ser reiniciado a partir da marcação parcial C_1 e S_1, não sendo necessário repetir as partes do processo que foram executadas corretamente, conforme mostrou o modelo analítico na seção 5.4. A Figura 84 mostra o registro de *log* do modelo de serviços *Web* privados após a correção do *deadlock*. Nesta simulação, foram consideradas as marcações parciais C_1 e S_1 e também foram realizadas 50 replicações. Após as replicações, foi verificado que todos os registros de *log* apresentaram a execução completa do modelo a partir das marcações

parciais.

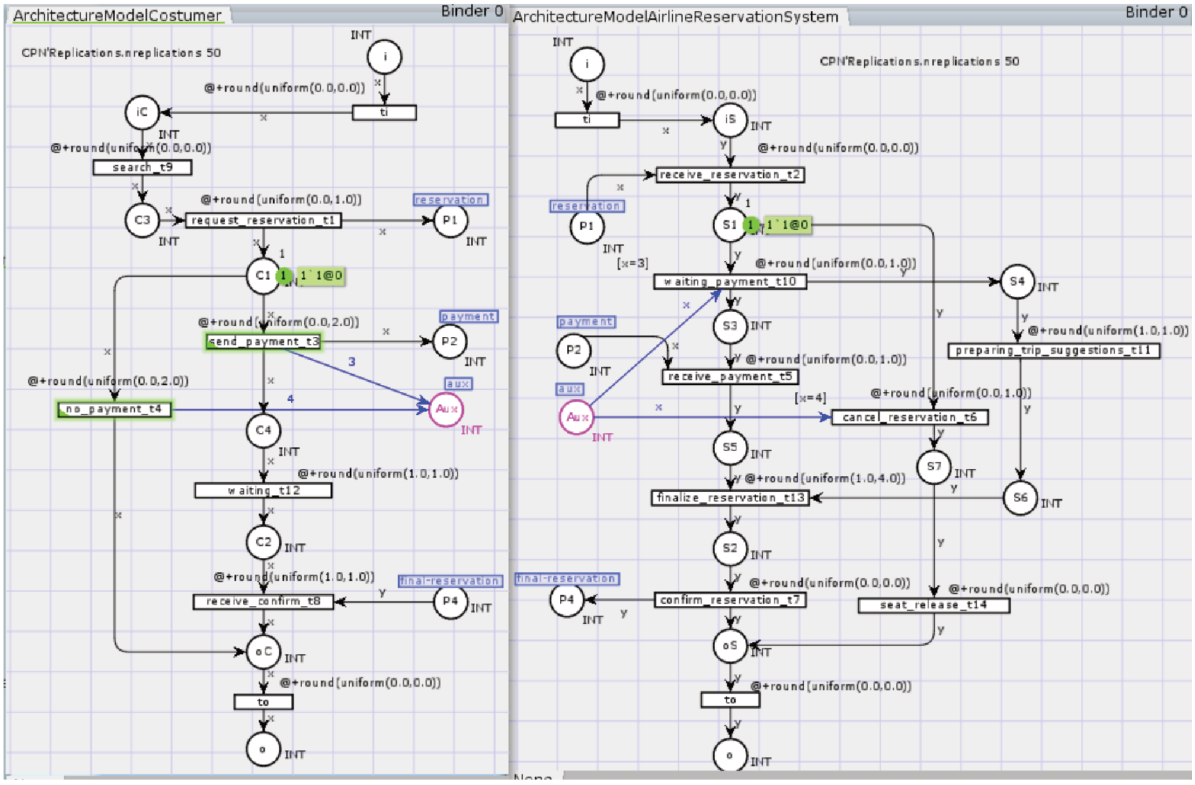


Figura 83 – Modelo de serviços *Web* privados representado no CPN Tools com sincronização de atividades para correção do *deadlock*.

```
CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Modelagem das redes utilizadas nos trabalhos/Case Study/CaseStudyArchitectureModel-
Deadlock-corrigido.cpn
Report generated: Wed Sep  2 16:49:27 2020

1       0       no_payment_t4 @ (1:ArchitectureModelCostumer)
2       2       cancel_reservation_t6 @ (1:ArchitectureModelAirlineReservationSystem)
3       2       seat_release_t14 @ (1:ArchitectureModelAirlineReservationSystem)
4       3       to @ (1:ArchitectureModelCostumer)
5       3       to @ (1:ArchitectureModelAirlineReservationSystem)


CPN Tools simulation report for:
/cygdrive/D/Doutorado/CPN/Modelagem das redes utilizadas nos trabalhos/Case Study/CaseStudyArchitectureModel-
Deadlock-corrigido.cpn
Report generated: Wed Sep  2 16:49:27 2020

1       0       send_payment_t3 @ (1:ArchitectureModelCostumer)
2       0       waiting_t12 @ (1:ArchitectureModelCostumer)
3       0       waiting_payment_t10 @ (1:ArchitectureModelAirlineReservationSystem)
4       0       receive_payment_t5 @ (1:ArchitectureModelAirlineReservationSystem)
5       0       preparing_trip_suggestions_t11 @ (1:ArchitectureModelAirlineReservationSystem)
6       1       finalize_reservation_t13 @ (1:ArchitectureModelAirlineReservationSystem)
7       3       confirm_reservation_t7 @ (1:ArchitectureModelAirlineReservationSystem)
8       3       receive_confirm_t8 @ (1:ArchitectureModelCostumer)
9       3       to @ (1:ArchitectureModelAirlineReservationSystem)
10      4       to @ (1:ArchitectureModelCostumer)
```

Figura 84 – Registro de *log* do modelo de serviços *Web* privados após a correção do *deadlock*.

Na Figura 85 é apresentado o grafo de alcançabilidade para o modelo de serviços *Web* privados apresentado na Figura 83. Já a Figura 86 mostra a parte do relatório de análise

onde consta a verificação da propriedade de Vivacidade. De acordo com o relatório, não há mais marcações mortas no modelo de serviços *Web* privados analisado, confirmando que não há mais a presença de *deadlock*.

Com as simulações apresentadas nesta seção utilizando o simulador CPN Tools, verifica-se que a abordagem proposta pode ser utilizada em sistemas reais não necessariamente *sound* com possibilidade de detecção, diagnóstico e remoção de *deadlock* em tempo de execução a partir de modificações localizadas na estrutura dos processos em serviços *Web*.

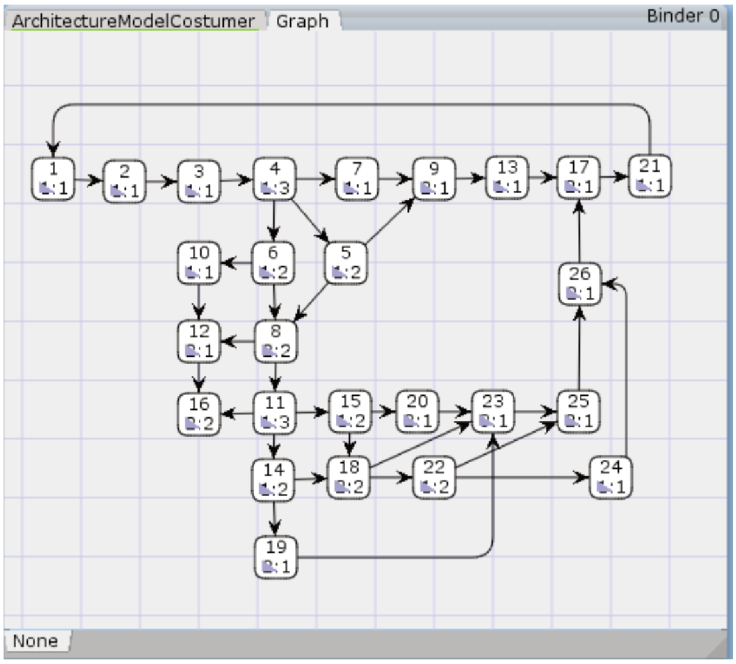


Figura 85 – Grafo de alcançabilidade gerado no CPN Tools para o modelo de serviços *Web* privados após a correção do *deadlock*.

```
Liveness Properties
-------------------------------------------

 Dead Markings
    None

 Dead Transition Instances
    None

 Live Transition Instances
    All
```

Figura 86 – Relatório de análise gerado no CPN Tools para o modelo de serviços *Web* privados após a correção do *deadlock*.

Como já citado anteriormente, no CPN Tools não é possível representar um conjunto de transições de fusão. Desse modo, para simulação, uma solução seria representar os processos em uma única janela, onde as transições que devem ser sincronizadas seriam substituídas por uma única transição, como pode ser observado na Figura 87 que mostra

o caso 1 de sincronização (apresentado na seção 4.2, Figura 45) modelado em uma única janela no CPN Tools. Na Figura 87 as transições t_{d1} e t_{d2}, que devem ser sincronizadas, são substituídas pela transição t_{d1}/t_{d2}. Esta representação em uma única janela pode ser utilizada como um modelo de análise; no entanto, é importante manter os modelos de processos em janelas separadas para que a solução proposta sirva de protótipo de um projeto de serviços *Web* em que a localidade dos processos deve ser mantida. Por isso, é utilizado no CPN Tools um lugar de comunicação assíncrona auxiliar que permite a troca de mensagens entre as partes envolvidas no processo que, juntamente com a utilização de guardas, simulam a sincronização de transições. Desse modo, é possível manter os processos em janelas distintas que se comunicam, mostrando os modelos de forma distribuída.

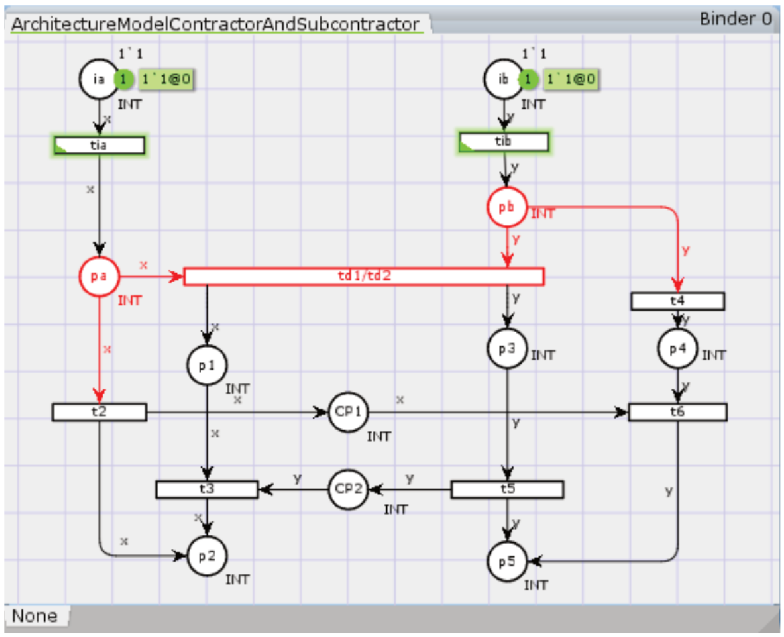


Figura 87 – Caso 1 de sincronização (Figura 45) representado no CPN Tools em uma única janela.

Para simular no CPN Tools os casos de sincronização apresentados na seção 4.2, Figuras 45, 46, 47 e 48, através da utilização de lugares de comunicação assíncrona, é necessário que todos as transições de saída do lugar *pb* ou *pb′* (definidos na seção 4.2 para representar lugares que permitem a escolha de outros caminhos) recebam a informação de quando poderão ser disparadas, uma vez que estas transições estão em conflito. Desse modo, todas estas transições terão como entrada um lugar de comunicação assíncrona auxiliar que receberá a informação de condição de disparo destas transições. Portanto, cada transição de saída do lugar *pb* ou *pb′* deverá ter uma condição de disparo associada.

A Figura 88 representa o caso 1 de sincronização (Figura 45) representado no CPN Tools. A Figura 88 (a) apresenta o modelo antes da sincronização e a Figura 88 (b) apresenta o modelo depois da sincronização. Neste caso, como pode ser observado na Figura 88 (b), o lugar de comunicação *cp* se torna um lugar auxiliar para comunicar

o possível disparo da transição t_{d1}. Se a transição t_{d1} é disparada então o numero 1 é depositado no lugar auxiliar e assim a transição t_{d2}, que possui uma condição de disparo $x = 1$, será disparada. Caso contrário, a transição t_4, que está em conflito com t_{d1}, será disparada.

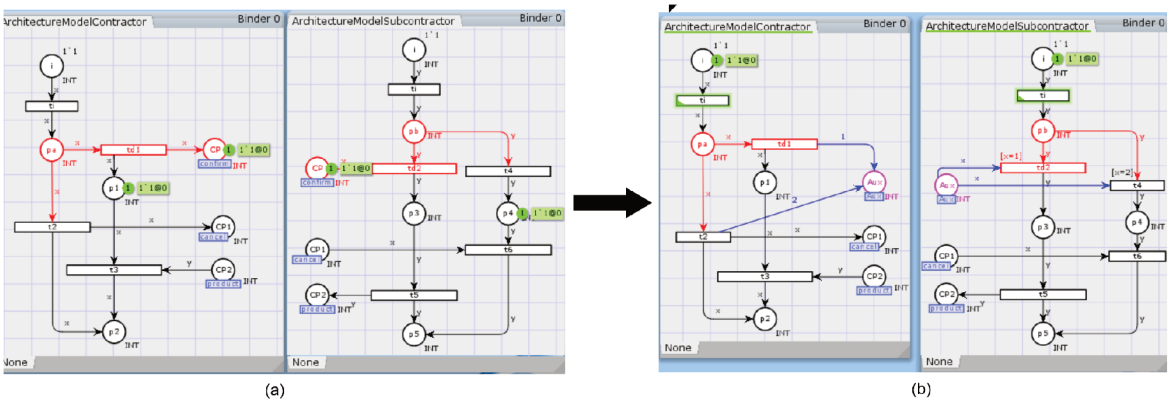


Figura 88 – Caso 1 de sincronização (Figura 45) representado no CPN Tools. Figura (a) antes da sincronização e Figura (b) depois da sincronização.

As Figuras 89, 90 e 91 representam respectivamenete os casos 2 (Figura 46), 3 (Figura 47) e 4 (Figura 48) de sincronização representados no CPN Tools. Nestes casos, o lugar *cp* se mantém e um lugar de comunicação auxiliar é criado.

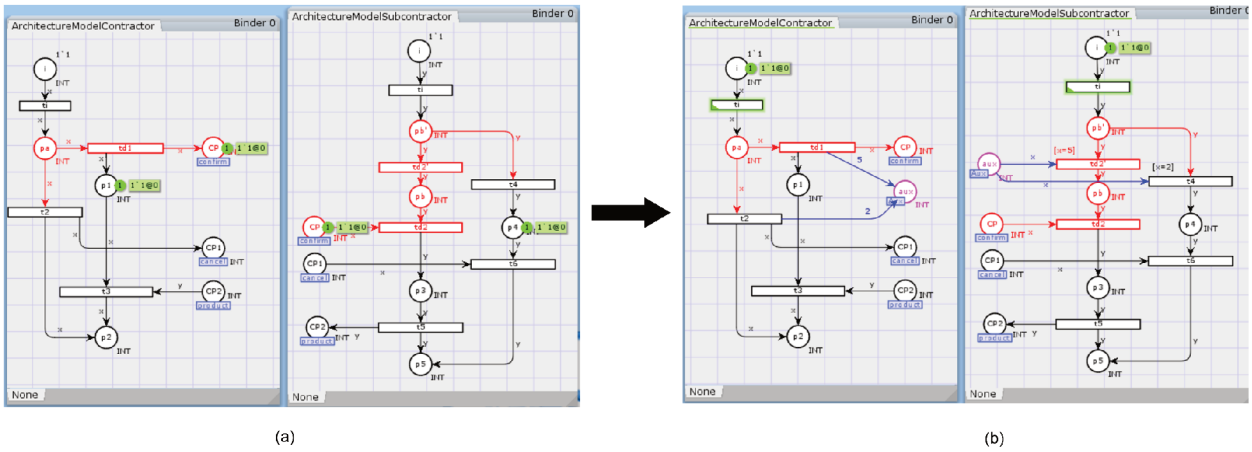


Figura 89 – Caso 2 de sincronização (Figura 46) representado no CPN Tools. Figura (a) antes da sincronização e Figura (b) depois da sincronização.

Existem casos onde, além de um lugar de comunicação auxiliar, também é necessário criar uma transição auxiliar que fará parte do protocolo de comunicação para sincronização das transições responsáveis pela situação de *deadlock*. Observe no exemplo da Figura 92 (a) que a transição t_{d2} recebe uma ficha do lugar de comunicação *cp* e a transição t_4 deposita uma ficha no lugar de comunicação $CP1$. Como t_{d2} e t_4 são as transições de saída do lugar *pb* e, portanto, estão em conflito, é necessário, neste caso, criar uma

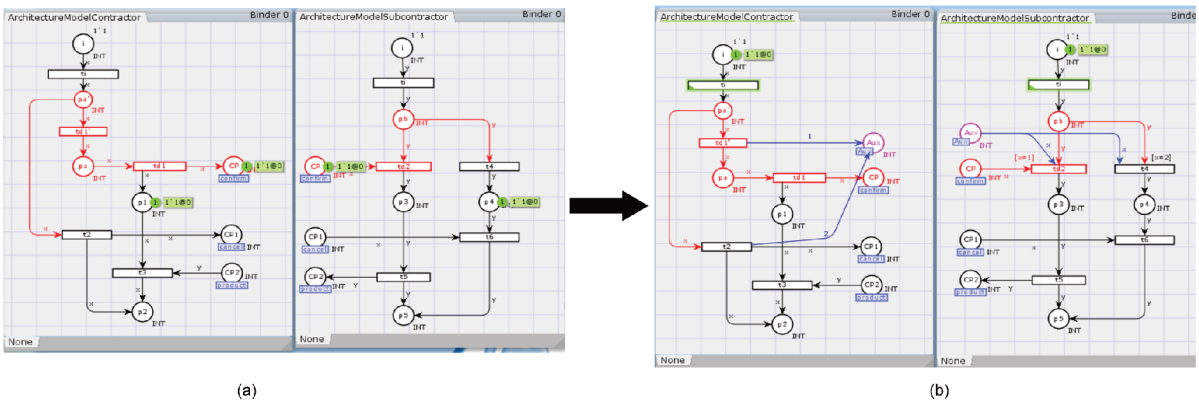


Figura 90 – Caso 3 de sincronização (Figura 47) representado no CPN Tools. Figura (a) antes da sincronização e Figura (b) depois da sincronização.

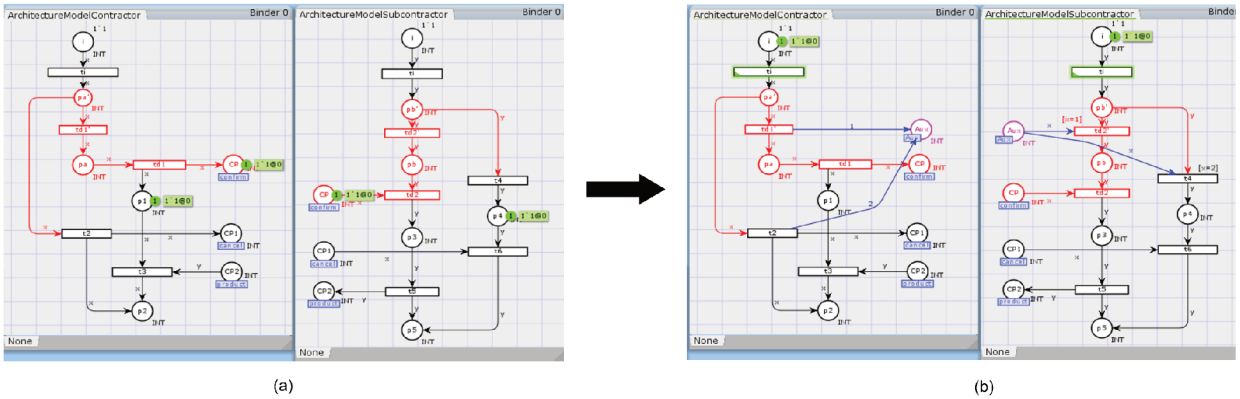


Figura 91 – Caso 4 de sincronização (Figura 48) representado no CPN Tools. Figura (a) antes da sincronização e Figura (b) depois da sincronização.

transição auxiliar no protocolo de comunicação para indicar o não disparo da transição t_{d1}. Para especificar tal controle, uma variável booleana $b1$ é utilizada. Desse modo, quando a transição auxiliar *taux* é disparada, $b1$ recebe o valor verdadeiro e a transição t_{d1} não pode mais ser disparada e, portanto, a transição t_4, que possui uma condição de disparo $b1 = true$, é disparada.

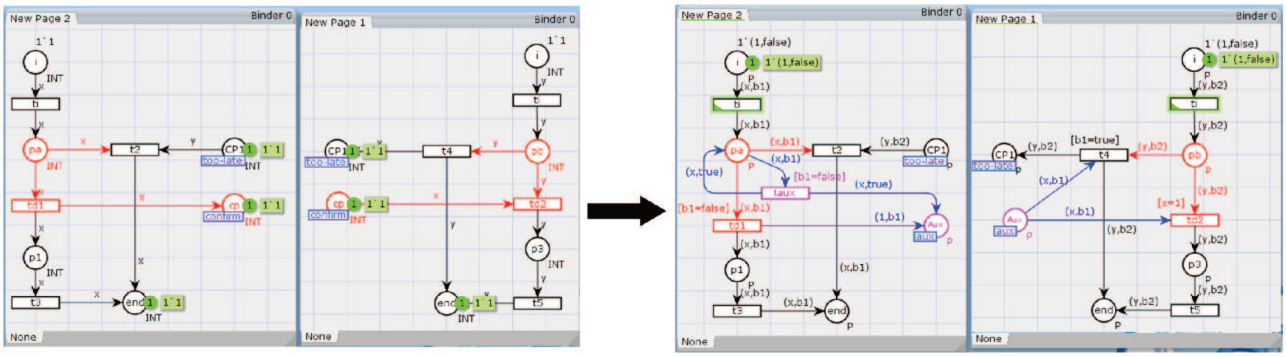


Figura 92 – Transição auxiliar no protocolo de comunicação.

Capítulo 6

Conclusão

Este capítulo apresenta as conclusões deste trabalho de pesquisa. A seção 6.1 apresenta as principais contribuições, na seção 6.2 são apresentados os trabalhos futuros que poderão ser desenvolvidos considerando os resultados já obtidos, e a seção 6.3 apresenta as contribuições em produção bibliográfica obtidas no contexto desse trabalho de pesquisa.

6.1 Principais Contribuições

Este trabalho de pesquisa apresentou métodos para a verificação comportamental e de desempenho em modelos de SOA. O propósito foi mostrar que, no contexto de SOA, todos os cenários presentes em um modelo de requisitos (*public* WF-net) também estão presentes na arquitetura correspondente (U(IOWF-net)) em termos de comportamento e desempenho. Neste trabalho, foram apresentados também métodos para a detecção e remoção de requisitos negativos do tipo *deadlock* em SOA. Os métodos foram baseados, em particular, na construção das árvores de prova da Lógica Linear, de grafos de precedência e de datas simbólicas associadas as fórmulas da Lógica Linear. Além disso, um estudo de caso foi realizado no contexto da verificação de composições de serviços *Web*, uma vez que há uma relação estreita entre a modelagem de um processo de *workflow* interorganizacional e uma composição de serviços *Web*, e verificado através de simulações realizadas no simulador CPN Tools.

Com as definições apresentadas no Capítulo 3, seções 3.2 e 3.3, e considerando a hipótese H1 (*Representando modelos de requisitos de sistemas de gerenciamento de processos de negócios e modelos de SOA por meio de redes de Petri (WorkFlow nets), é possível, baseando-se no cálculo dos sequentes da Lógica Linear, definir um algoritmo de verificação de requisitos funcionais e não funcionais em SOA.*), a principal questão de pesquisa Q1 (*É possível verificar formalmente que modelos de SOA atendem requisitos especificados em modelos de análise?*) foi atendida.

As questões de pesquisa Q2 (*Como representar uma SOA utilizando WorkFlow nets?*) e Q3 (*Como representar o modelo de especificação de requisitos que servirá como base para*

identificar os requisitos que deverão ser verificados no modelo de uma SOA?) também foram atendidas com as definições apresentadas no Capítulo 3, seção 3.1.

O método para a verificação dos requisitos de serviços foi definido no Capítulo 3, seção 3.2. Utilizando um tipo de grafo, chamado de grafo de precedência, foi possível verificar a equivalência comportamental entre os modelos público (modelo de requisito) e privado (arquitetura), em particular, provando que tais modelos simulam o comportamento um do outro respeitando a noção de bissimulação *branching*. No método apresentado, os grafos de precedência são construídos para todos os cenários do modelo público. Já no modelo privado, os grafos de precedência são somente construídos para os cenários que finalizam corretamente os processos de negócio modelados, ou seja, livres de *deadlock*, e que, possivelmente, correspondam a algum cenário expresso no modelo público. Uma das principais vantagens do método proposto foi definir, através do uso da Lógica Linear e dos grafos de precedência, um novo tipo de semântica operacional associada aos requisitos de negócio. Com as definições e o método apresentados no Capítulo 3, seção 3.2, a questão de pesquisa Q4 (*Considerando modelos de requisitos e de SOA, representados por WorkFlow nets, como definir um algoritmo para verificar a equivalência entre esses modelos em relação aos requisitos funcionais no que diz respeito ao comportamento dos modelos?*) foi atendida.

Considerando a noção de intervalos de datas simbólicas, foi possível verificar a equivalência em termos de desempenho entre um modelo público e um modelo privado. O método para esta verificação foi definido no Capítulo 3, seção 3.3. A maior vantagem do uso de datas simbólicas é que uma vez que tenham sido calculadas para um cenário específico, elas podem ser usadas em qualquer caso que será tratado pelo mesmo cenário, mesmo que as datas numéricas de início dos casos considerados sejam diferentes. Portanto, uma vez calculadas, as datas simbólicas podem ser reutilizadas para valores numéricos variados associados às diversas ações dos processos. Com as definições e o método apresentados no Capítulo 3, seção 3.3, a questão de pesquisa Q5 (*Considerando modelos de requisitos e de SOA, representados por WorkFlow nets, como definir um algoritmo para verificar a equivalência entre esses modelos em relação aos requisitos não funcionais no que diz respeito ao desempenho dos modelos?*) foi atendida.

Os métodos apresentados no Capítulo 3 consideram modelos de SOA não necessariamente *sound*. No entanto, somente os cenários livres de *deadlock* e compatíveis com as atividades especificadas nos cenários do modelo de requisitos (modelo público de contrato de serviços) serão considerados a fim de verificar formalmente a equivalência dos requisitos de comportamento e de desempenho. Portanto, as organizações não precisam ser restringidas por atores externos para construir seus processos de *workflow* privados (como geralmente acontece nos sistemas empresariais existentes), pois podem, simplesmente, verificar se um conjunto de cenários de requisitos de um modelo de análise estão presentes em uma SOA sem necessariamente considerar todos os serviços existentes e

fornecidos pela arquitetura. Nesse sentido, os impactos e desvios gerados pela colaboração entre diferentes organizações podem ser minimizados. Desse modo, o número de árvores a serem construídas pode ser significativamente reduzido no método para verificação dos requisitos de serviços. Adicionalmente, os métodos são baseados na Lógica Linear, pois o principal problema de usar puramente redes de Petri ou outro método baseado em autômatos é a explosão combinatorial quando o conjunto de estados é gerado. A Lógica Linear permite a representação e análise dos métodos sem gerar o grafo de alcançabilidade. Além disso, os autômatos não explicam corretamente o paralelismo explícito dos processos. Uma limitação da utilização da Lógica Linear neste trabalho é a falta de representação de possíveis ciclos nos modelos; desse modo, o desempenho pode ser influenciado pela não representação dos ciclos. Adicionalmente, não será possível verificar se um ciclo é livre de *livelock*.

Uma importante observação é que as árvores de prova da Lógica Linear podem ser usadas para verificar tanto os requisitos de comportamento quanto os requisitos de desempenho. Portanto, a mesma árvore de prova pode ser utilizada para análises de tipo qualitativas (requisitos funcionais de comportamento) bem como para análises de tipo quantitativas (requisitos não funcionais de desempenho).

O método para detecção de requisitos negativos do tipo *deadlock* foi definido no Capítulo 4, seção 4.1. Mesmo com uma arquitetura detalhada, não há garantias de que durante o funcionamento normal do sistema, novos comportamentos que não foram previstos surjam para os usuários do sistema. Uma das razões para isso é que cada parte envolvida no processo possui relativa autonomia e poderá alterar o seu funcionamento interno a qualquer momento. Portanto, na prática, tais estados indesejados serão encontrados somente durante a execução do sistema. Desse modo, a partir de uma marcação indesejada, que representa um estado parcial do modelo, são identificadas todas as sequências de ações (cenários) que podem tornar um requisito de serviço em um requisito negativo do tipo *deadlock*. Para a identificação destas sequências de ações, foi utilizado neste trabalho um raciocínio inverso, ou seja, todos os arcos da U(IOWF-net) foram invertidos. Desse modo, a partir de uma marcação indesejada foi possível traçar o caminho percorrido para que tal marcação fosse alcançada. O método para detecção de requisitos negativos do tipo *deadlock* seguiu a mesma ideia do método utilizado na verificação de requisitos funcionais apresentado no Capítulo 3, seção 3.2. Foi considerado o cálculo dos componentes repetitivos estacionários da U(IOWF-net) inversa para encontrar os possíveis cenários que levam o sistema a um requisito negativo; na sequência, estes cenários foram representados em sequentes da Lógica Linear e provados por meio da construção das árvores de prova da Lógica Linear; por fim, os grafos de precedência para os cenários corretamente provados foram gerados. Com as definições e o método apresentados no Capítulo 4, seção 4.1, a questão de pesquisa Q6 (*Considerando o modelo de SOA, representado por WorkFlow nets, como definir um algoritmo para detectar possíveis situações que podem levar uma*

SOA a um estado de deadlock?) foi atendida.

Após a identificação dos cenários que podem tornar um requisito de serviço em um requisito negativo do tipo *deadlock*, foi possível controlar o *deadlock* através da sincronização de partes dos processos locais. O método para a realização deste controle foi definido no Capítulo 4, seção 4.2. A sincronização é realizada através da inserção de um elemento de comunicação síncrona, ou seja, transições que são sincronizadas e forçadas a serem disparadas ao mesmo tempo. A sincronização força os processos de *workflow* locais a executarem simultaneamente certas atividades, removendo, desse modo, a situação de *deadlock* do modelo. Com as definições e o método apresentados Capítulo 4, seção 4.2, a questão de pesquisa Q7 (*Ao detectar que uma SOA não é livre de deadlock, como alterar a arquitetura para transformar os processos não sound em processos seguros e confiáveis, mantendo os requisitos funcionais e não alterando de forma significativa o desempenho?*) foi atendida.

Os métodos definidos no Capítulo 4 não consideram a construção do grafo de alcançabilidade que explora sistematicamente uma rede de Petri assim como acontece na maioria dos métodos que abordam o problema de *deadlock*. Como já citado anteriormente, uma desvantagem em considerar grafos de alcançabilidade é que pode ocorrer uma explosão do número de estados discretos, pois o processo de modelagem envolve a enumeração de todos os estados possíveis, o que, geralmente, leva a uma complexidade elevada nos algoritmos de verificação utilizados. Por isso, neste trabalho, as situações de *deadlock* são consideradas somente quando, de fato, elas ocorrem. Desse modo, a partir dos estados indesejados, uma análise é realizada para diagnosticar as ações que levaram ao alcance de tais estados; na sequência, o modelo arquitetural é corrigido para impedir ocorrências futuras de tal comportamento.

Um estudo de caso foi apresentado no Capítulo 5, onde, inicialmente, foi apresentada a relação entre a modelagem de um processo de *workflow* interorganizacional e a modelagem de uma composição de serviços *Web*. Na sequência, os métodos definidos nos Capítulos 3 e 4 foram aplicados no contexto de composições de serviços *Web*. Para verificar a efetividade dos métodos propostos para a verificação de requisitos funcionais de serviços e não funcionais de desempenho, e também do método para remoção de requisitos negativos do tipo *deadlock*, um modelo de simulação baseado no simulador CPN Tools foi definido na seção 5.5 e aplicado ao estudo de caso. Com as simulações apresentadas, verificou-se que as abordagens propostas neste trabalho de pesquisa podem ser utilizadas em sistemas reais não necessariamente *sound* com possibilidade de detecção, diagnóstico e remoção de *deadlock* em tempo de execução a partir de modificações localizadas na estrutura dos processos em serviços *Web*.

6.2 Trabalhos Futuros

Neste trabalho de pesquisa não foi abordado como o modelo de requisitos foi obtido, ou seja, foi considerado um modelo de requisitos finalizado. Deste modo, como trabalho futuro, é de grande interesse definir um método que obterá o modelo de requisitos no contexto de SOA. Isso significa encontrar os serviços correspondentes entres as partes públicas do processo. Uma hipótese para este problema é explorar as propriedades estruturais das redes de Petri do modelo público a fim de encontrar uma composição de requisitos funcionais que deverá ser atendida pela SOA candidata a satisfazer os contratos de serviços especificados no modelo público. Uma propriedade estrutural que poderia ser explorada, por exemplo, seria o cálculo dos componentes conservativos (invariantes de lugar).

Um componente conservativo definirá um subconjunto de lugares que isoladamente formará uma subrede de Petri. Os lugares pertencentes a um componente conservativo estarão conectados a um subconjunto de transições. É provável que este subconjunto de transições faça parte de algum dos subcenários (conjunto de ações) da rede de Petri pública correspondente, sendo que estes subcenários podem ser obtidos pelo cálculo dos componentes repetivivos estaciónarios (invariantes de transição). Desse modo, dois ou mais componentes conservativos de uma rede pública poderiam ser combinados para encontrar os subcenários de cada rede de Petri pública; por sua vez, os subcenários pertencentes as diferentes redes de Petri públicas poderiam ser combinados para encontrar os cenários completos, ou seja, os serviços que fariam parte do modelo de requisitos. Portanto, seria possível encontrar subsequentes menores (requisitos mínimos) para posteriormente procurar sequentes de maneira distribuída (serviços), como, por exemplo, em serviços *Web*. A Figura 93 resume a hipótese para a obtenção do modelo de requisitos no contexto de SOA.

Outro trabalho futuro está relacionado com a validação por meio de testes funcionais e não funcionais. A atividade de teste contribui por construir sistemas com maior qualidade e confiabilidade sem, no entanto, representar altos custos para as empresas (DELAMARO; MALDONADO; JINO, 2007). Por isso, a importância de se incluir métodos, ferramentas e técnicas que auxiliem a atividade de teste durante o ciclo de desenvolvimento de software. De acordo com (BEIZER, 1990), testes são procedimentos formais; assim, as entradas devem ser preparadas, as saídas previstas, os testes documentados, os comandos executados, e os resultados observados.

Uma hipótese para a aplicação de testes nos modelos apresentados neste trabalho de pesquisa, primeiramente, é representar os modelos de requisitos (*public* WF-net) e de arquitetura (U(IOWF-net)) com dados. Uma *WorkFlow net* com Dados (WFD-net) é um tipo especial de redes de Petri com anotações relacionadas ao manuseio de dados (TRČKA; AALST; SIDOROVA, 2009). Em (HADDAR; TMAR; GARGOURI, 2012), por exemplo, uma nova abordagem para modelagem de *workflow* com dados foi apresentada; os autores utilizaram uma notação de estruturas de processo dirigidas a dados.

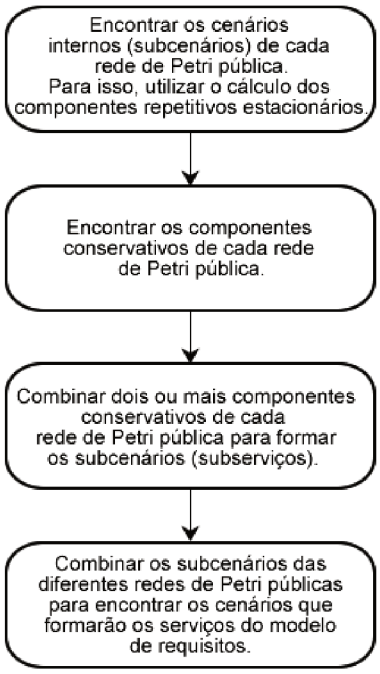


Figura 93 – Hipótese para a obtenção do modelo de requisitos no contexto de SOA.

Alguns trabalhos, como por exemplo (GOTTSCHALK et al., 2008) e (JULIA; VALE; PASSOS, 2016), consideram a realização de testes em processos de negócio representados por redes de Petri. Os dados, poderiam ser, por exemplo, algum tipo de predicado associado as transições. Após considerar os modelos de requisitos e de arquitetura com dados, subconjuntos de dados poderiam ser considerados e testados (subconjunto de testes) para verificar se atendem pelo menos cada sequente (cenário) que foi provado corretamente. O modelo público de requisitos apresentaria dados mínimos a serem testados e o modelo privado de arquitetura possuiria dados adicionais que não apareceriam no modelo público. Os casos de teste poderiam ser simulados no CPN Tools que permite a modelagem de *WorkFlow nets* com Dados, a visualização dos processos em execução e também condições para a realização de análises nos resultados obtidos.

Neste trabalho de pesquisa, foi realizada comparações comportamentais e também comparações em relação ao tempo de finalização dos processos envolvidos no modelo de SOA. Além dessas comparações, seria interessante considerar também como trabalho futuro a comparação dos processos em relação a outras perspectivas. Em (SYAMSIYAH et al., 2017), por exemplo, é apresentada uma metodologia que realiza a comparação de processos, no contexto de mineração de processos, considerando múltiplos aspectos como organizacionais, de dados, desempenho, etc.

Como o modelo arquitetural é privado, podem surgir cenários completos que não foram especificados no modelo de requisitos. Desse modo, seria uma atividade significativa verificar a relevância destes novos cenários e, assim, realizar algum tipo de engenharia reversa (CHIKOFSKY; CROSS, 1990) para que os cenários possam ser inseridos no modelo de

requisitos abstraindo o detalhamento que possuem no modelo arquitetural.

6.3 Contribuições em Produção Bibliográfica

Na sequência são apresentadas as produções bibliográficas resultantes deste trabalho de pesquisa.

- Artigo (OLIVEIRA; OLIVEIRA; JULIA, 2017b), intitulado *Using Linear Logic to Verify Requirement Scenarios in SOA Models based on Interorganizational Work-Flow Nets Relaxed Sound*. Foi publicado na 19º *International Conference on Enterprise Information Systems* em 2017 e refere-se ao método definido no Capítulo 3, seção 3.2, para a verificação de cenários de requisitos comportamentais em SOA. Nesse trabalho foi considerado modelos que não eram necessariamente livres de *deadlock*.

- Artigo (OLIVEIRA; JULIA, 2017), intitulado *Using Linear Logic to Verify Requirement Scenarios in Composite Web Service*. Foi publicado no XX Simpósio Brasileiro de Métodos Formais em 2017 e trata da aplicação do método proposto no Capítulo 3, seção 3.2, em serviços *Web*.

- Artigo (OLIVEIRA; OLIVEIRA; JULIA, 2017a), intitulado *Requirement Verification in SOA Models Based on Interorganizational WorkFlow Nets and Linear Logic*. Foi publicado na 14º *International Conference on Information Technology: New Generations* em 2017 e refere-se ao método definido no Capítulo 3, seção 3.2, para a verificação de cenários de requisitos comportamentais em SOA. Nesse trabalho foi considerado somente modelos livres de *deadlock*.

- Artigo (OLIVEIRA; JULIA, 2019), intitulado *Using Symbolic Dates of the Linear Logic to Verify Performance Requirements in SOA Models*. Foi publicado na 16º *International Conference on Information Technology: New Generations* em 2019 e refere-se ao método definido no Capítulo 3, seção 3.3, para a verificação de requisitos de desempenho em SOA.

- Artigo (OLIVEIRA et al., 2017b), intitulado *A Synchronization Rule Based on Linear Logic for Deadlock Prevention in Interorganizational WorkFlow Nets*. Foi publicado na 14º *International Conference on Information Technology: New Generations* em 2017 e refere-se a ideia inicial de um método para a remoção de requisitos negativos do tipo *deadlock* em SOA.

- Artigo (OLIVEIRA et al., 2017a), intitulado *A Linear Logic based Synchronization Rule for Deadlock Prevention in Web Service Composition*. Foi publicado na 19º *International Conference on Enterprise Information Systems* em 2017 e também

refere-se a ideia inicial de um método para a remoção de requisitos negativos do tipo *deadlock* em SOA.

- Artigo (OLIVEIRA; JULIA, 2020), intitulado *Detection and removal of negative requirements of deadlock-type in Service-Oriented Architectures*. Foi publicado na *International Conference on Computational Science and Computational Intelligence* em 2020 e refere-se aos métodos para detecção e remoção de requisitos negativos do tipo *deadlock* em SOA definidos no Capítulo 4, seções 4.1 e 4.2.

- O artigo intitulado *Behaviour and performance verification of requirement scenarios in Service-Oriented Architecture models based on Linear Logic and Interorganizational WorkFlow nets Relaxed Sound* foi submetido para uma revista.

Referências Bibliográficas

AALST, W. M. P. van der. Petri-net-based workflow management software. In: CITESEER. **Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems**. [S.l.], 1996. p. 114–118.

_____. **Structural Characterizations of Sound Workflow Nets**. [S.l.], 1996.

_____. The application of petri nets to workflow management. **Journal of Circuits, Systems, and Computers**, v. 8, n. 1, p. 21–26, 1998.

_____. Modeling and analyzing interorganizational workflows. In: IEEE COMPUTER SOCIETY PRESS. **International Conference on Application of Concurrency to System Design**. [S.l.], 1998. p. 262–272.

_____. Interorganizational workflows: An approach based on message sequence charts and petri nets. Citeseer, 1999.

_____. Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. **Information and Management**, v. 37, n. 2, p. 67–75, 2000.

_____. Inheritance of interorganizational workflows: How to agree to disagree without loosing control? **Information Technology and Management**, v. 4, n. 4, p. 345–389, Octuber 2003.

AALST, W. M. P. van der; HEE, K. M. van. **Workflow Management: Models, methods and systems**. [S.l.]: The MIT Press, 2004.

AALST, W. M. P. van der et al. Soundness of workflow nets: classification, decidability, and analysis. **Formal Aspects of Computing**, v. 23, n. 3, p. 333–363, 2011.

AALST, W. M. P. van der; HOFSTEDE, A. H. M. ter. Yawl: Yet another workflow language. **Information Systems**, Elsevier Science Ltd., v. 30, n. 4, p. 245–275, jun 2005. https://doi.org/10.1016/j.is.2004.02.002.

AALST, W. M. P. van der; MEDEIROS, A. A. D.; WEIJTERS, A. Process equivalence: Comparing two process models based on observed behavior. In: SPRINGER. **International Conference on Business Process Management**. [S.l.], 2006. p. 129–144. https://doi.org/10.1007/11841760_10.

AALST, W. M. P. van der; WESKE, M. The p2p approach to interorganizational workflows. In: SPRINGER-VERLAG BERLIN HEIDELBERG. **13th International Conference on Advanced Information Systems Engineering**. [S.l.], 2001. p. 140–156.

_____. Reflections on a decade of interorganizational workflow research. In: **Seminal contributions to information systems engineering**. [S.l.]: Springer, 2013. p. 307–313. https://doi.org/10.1007/978-3-642-36926-1_24.

ALONSO, G. et al. **Web Services: Concepts, Architectures and Applications**. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2004. ISBN 3642078885.

AMATO, F.; MOSCATO, F. A modeling profile for availability analysis of composite cloud services. In: IEEE. **P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on**. [S.l.], 2014. p. 406–413. https://doi.org/10.1109/3PGCIC.2014.85.

ARIAS, R.; HIRATA, C. M. Mapping of software model to simulation model for performance requirement verification. In: **Proceedings of the 44th Annual Simulation Symposium**. [S.l.]: Society for Computer Simulation International, 2011. p. 142–150.

ARMAS-CERVANTES, A. et al. On the suitability of generalized behavioral profiles for process model comparison. In: _____. **Web Services, Formal Methods, and Behavioral Types: 11th International Workshop, WS-FM 2014, Eindhoven, The Netherlands, September 11-12, 2014, and 12th International Workshop, WS-FM/BEAT 2015, Madrid, Spain, September 4-5, 2015, Revised Selected Papers**. [S.l.]: Springer International Publishing, 2016. p. 13–28.

BANASZAK, Z. A.; KROGH, B. H. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. **IEEE Transactions on Robotics and Automation**, v. 6, n. 6, p. 724–734, Dec 1990. https://doi.org/10.1109/70.63273.

BARKAOUI, K.; ABDALLAH, I. B. Deadlock avoidance in FMS based on structural theory of Petri nets. In: IEEE. **Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95**. [S.l.], 1995. v. 2, p. 499–510. https://doi.org/10.1109/ETFA.1995.496690.

BASTEN, T. **In Terms of Nets System Design With Petri Nets and Process Algebra**. Tese (Doutorado) — Eindhoven University of Technology, Eindhoven, Netherlands, 1998.

BEIZER, B. **Software Testing Techniques**. International Thomson Computer Press, 1990. ISBN 9781850328803. Disponível em: <https://books.google.com.br/books?id=jjXTHQAACAAJ>.

BENDRISS, S.; BENABDELHAFID, A. Enabling goods traceability through data modeling and semantic web service ontologies. In: IEEE. **2011 4th International Conference on Logistics**. [S.l.], 2011. p. 385–390. https://doi.org/10.1109/LOGISTIQUA.2011.5939431.

BHANDARI, G. P.; GUPTA, R.; UPADHYAY, S. K. Colored Petri nets based fault diagnosis in Service Oriented Architecture. **International Journal of Web Services Research (IJWSR)**, IGI Global, v. 15, n. 4, p. 1–28, 2018. https://doi.org/10.4018/IJWSR.2018100101.

BOUALI, M.; ROCHETEAU, J.; BARGER, P. Application of Linear Logic to backward reachability analysis of Colored Petri nets. In: **The European Safety and Reliability Conference (ESREL'09)**. [S.l.: s.n.], 2009. 10.1201/9780203859759.ch272.

BOUKHEDOUMA, S. et al. Adaptation patterns for service based inter-organizational workflows. In: IEEE. **IEEE 7th International Conference on Research Challenges in Information Science (RCIS)**. [S.l.], 2013. p. 1–10. https://doi.org/10.1109/RCIS.2013.6577722.

_____. Service based cooperation patterns to support flexible inter-organizational workflows. In: **IJITCS**. [S.l.: s.n.], 2014. v. 6, n. 4, p. 1–18. https://doi.org/10.5815/ijitcs.2014.04.01.

BUFFONI-ROGOVCHENKO, L. et al. Requirement verification and dependency tracing during simulation in Modelica. In: IEEE. **Modelling and Simulation (EUROSIM), 2013 8th EUROSIM Congress on**. [S.l.], 2013. p. 561–566. https://doi.org/10.1109/EUROSIM.2013.99.

CHEBBI, I.; DUSTDAR, S.; TATA, S. The view-based approach to dynamic inter-organizational workflow cooperation. **Data & Knowledge Engineering**, Elsevier, v. 56, n. 2, p. 139–173, 2006. https://doi.org/10.1016/j.datak.2005.03.008.

CHENG, A.; ESPARZA, J.; PALSBERG, J. Complexity results for 1-safe nets. In: SPRINGER. **International Conference on Foundations of Software Technology and Theoretical Computer Science**. [S.l.], 1993. p. 326–337. https://doi.org/10.1016/0304-3975(94)00231-7.

CHIKOFSKY, E. J.; CROSS, J. H. Reverse engineering and design recovery: A taxonomy. **IEEE software**, Ieee, v. 7, n. 1, p. 13–17, 1990. https://doi.org/10.1109/52.43044.

CORMEN, T. H. et al. **Introduction to Algorithms, Third Edition**. 3rd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262033844, 9780262033848.

DELAMARO, M.; MALDONADO, J.; JINO, M. **Introdução ao teste de software**. Elsevier, 2007. ISBN 9788535226348. Disponível em: <https://books.google.com.br/books?id=7R6XPgAACAAJ>.

DEMMOU, H. et al. Critical scenarios derivation methodology for mechatronic systems. **Reliability Engineering & System Safety**, Elsevier, v. 84, n. 1, p. 33–44, 2004. https://doi.org/10.1016/j.ress.2003.11.007.

DIAZ, M. **Petri nets: fundamental models, verification and applications**. [S.l.]: John Wiley & Sons, 2013.

DIJKMAN, R. et al. Similarity of business process models: Metrics and evaluation. **Information Systems**, v. 36, n. 2, p. 498 – 516, 2011.

_____. Aligning Business Process Models. In: **Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference (Edoc 2009)**. [S.l.]: IEEE Computer Society, 2009. p. 45–53. https://doi.org/10.1109/EDOC.2009.11.

DILLON, L. K.; SANKAR, S. Introduction to the special issue. **IEEE Transactions on Software Engineering**, v. 23, n. 5, p. 265–266, May 1997. https://doi.org/10.1109/TSE.1997.590647.

DING, J. et al. An approach to modeling software architecture based on SOA. In: WORLD SCIENTIFIC. **Materials, Manufacturing Technology, Electronics and Information Science**. [S.l.], 2016. p. 382–389. https://doi.org/10.1142/9789813109384_0041.

DINGLE, N. J.; KNOTTENBELT, W. J.; SUTO, T. PIPE2: A tool for the performance evaluation of generalised stochastic Petri nets. **SIGMETRICS Perform. Eval. Rev.**, ACM, New York, NY, USA, v. 36, n. 4, p. 34–39, mar. 2009. ISSN 0163-5999. https://doi.org/10.1145/1530873.1530881.

DONGEN, B.; DIJKMAN, R.; MENDLING, J. Measuring similarity between business process models. In: **Proceedings of the 20th International Conference on Advanced Information Systems Engineering**. [S.l.]: Springer-Verlag, 2008. p. 450–464.

DU, Y.; LI, X.; XIONG, P. A Petri net approach to mediation-aided composition of Web services. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 9, n. 2, p. 429–435, 2012. https://doi.org/10.1109/TASE.2012.2188511.

EDER, J. et al. View-based interorganizational workflows. In: **Proceedings of the 12th International Conference on Computer Systems and Technologies**. [S.l.]: ACM, 2011. p. 1–10. https://doi.org/10.1145/2023607.2023609.

EHRIG, M.; KOSCHMIDER, A.; OBERWEIS, A. Measuring similarity between semantic business process models. In: **Proceedings of the Fourth Asia-Pacific Conference on Comceptual Modelling - Volume 67**. [S.l.]: Australian Computer Society, Inc., 2007. p. 71–80. https://doi.org/10.1007/978-3-540-69534-9_34.

ERL, T. **Service-Oriented Architecture: Concepts, Technology, and Design**. [S.l.]: Prentice Hall, 2005. ISBN 0134524454.

_____. **SOA Principles of Service Design**. [S.l.]: Prentice Hall, 2009.

EUZENAT, J.; SHVAIKO, P. **Ontology Matching**. Second. [S.l.]: Springer-Verlag, 2013. https://doi.org/10.1007/978-3-642-38721-0.

FAHLAND, D. et al. Instantaneous soundness checking of industrial business process models. In: **Proceedings of the 7th International Conference on Business Process Management**. Berlin, Heidelberg: Springer-Verlag, 2009. (BPM '09), p. 278–293. https://doi.org/10.1007/978-3-642-03848-8_19.

GIRARD, J.-Y. Linear logic. **Theoretical Computer Science**, v. 50, n. 1, p. 1–102, 1987. https://doi.org/10.1016/0304-3975(87)90045-4.

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

GLABBEEK, R. J. V.; WEIJLAND, W. P. Branching time and abstraction in bisimulation semantics. In: **Information Processing 89, proc. IFIP 11th World Computer Congress (G.X.Ritter, ed.)**. [S.l.: s.n.], 1989. p. 613–618.

GOKNIL, A.; KURTEV, I.; BERG, K. V. D. Generation and validation of traces between requirements and architecture based on formal trace semantics. **Journal of Systems and Software**, v. 88, p. 112 – 137, 2014. https://doi.org/10.1016/j.jss.2013.10.006.

GOTTSCHALK, F. et al. Protos2CPN: Using colored Petri nets for configuring and testing business processes. **International Journal on Software Tools for Technology Transfer**, Springer, v. 10, n. 1, p. 95–110, 2008. https://doi.org/10.1007/s10009-007-0055-9.

HADDAR, N.; TMAR, M.; GARGOURI, F. A data-driven workflow based on structured tokens Petri net. In: **ICSEA 2012**. [S.l.: s.n.], 2012.

HAMADI, R.; BENATALLAH, B. A petri net-based model for web service composition. In: AUSTRALIAN COMPUTER SOCIETY, INC. **Proceedings of the 14th Australasian database conference-Volume 17**. [S.l.], 2003. p. 191–200. https://doi.org/10.13140/2.1.3643.9688.

HAREL, D. Statecharts: A Visual Formalism for Complex Systems. **Science of Computer Programing**, Elsevier North-Holland, Inc., v. 8, n. 3, p. 231–274, jun. 1987. https://doi.org/10.1016/0167-6423(87)90035-9.

HIERONS, R. M.; NEZ, M. N. Implementation relations and probabilistic schedulers in the distributed test architecture. **Journal of Systems and Software**, 2017. https://doi.org/10.1016/j.jss.2017.03.011.

HOFSTEDE, A. H. ter et al. **Modern Business Process Automation: YAWL and its support environment**. [S.l.]: Springer Science & Business Media, 2009.

HOLLINGSWORTH, D. **Workflow Management Coalition - The Workflow Reference Model**. [S.l.], 1994.

HOYOS, H.; CASALLAS, R.; JIMÉNEZ, F. HiLes-T: An ADL for early requirement verification of embedded systems. In: **Proceedings of the 5th International Workshop on Model Based Architecting and Construction of Embedded Systems**. [S.l.]: ACM, 2012. p. 7–12. ISBN 978-1-4503-1800-6. https://doi.org/10.1145/2432631.2432633.

HUANG, G. et al. Rationality of service composition of workflow net in a Service Oriented Architecture. In: SPRINGER. **International Conference on Informatics and Semiotics in Organisations**. [S.l.], 2014. p. 155–165. https://doi.org/10.1007/978-3-642-55355-4_16.

JULIA, S.; VALE, L. do N.; PASSOS, L. Functional testing using object WorkFlow nets. **Comput. Informatics**, v. 35, p. 719–743, 2016.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [S.l.], 2007.

KLAI, K.; OCHI, H.; TATA, S. Formal Abstraction and Compatibility Checking of Web Services. In: **20th International Conference on Web Services (ICWS)**. [S.l.]: IEEE, 2013. p. 163–170. https://doi.org/10.1109/ICWS.2013.31.

KUNZE, M.; WEIDLICH, M.; WESKE, M. Behavioral Similarity: A Proper Metric. In: **Proceedings of the 9th International Conference on Business Process Management**. [S.l.]: Springer-Verlag, 2011. p. 166–181. https://doi.org/10.1007/978-3-642-23059-2_15.

LAZARD, D. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In: SPRINGER. **European Conference on Computer Algebra**. [S.l.], 1983. p. 146–156. https://doi.org/10.1007/3-540-12868-9_99.

LI, J.; FAN, Y.; ZHOU, M. Timing constraint workflow nets for workflow analysis. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 33, n. 2, p. 179–193, March 2003. https://doi.org/10.1109/TSMCA.2003.811771.

LIN, D.; ISHIDA, T. Interorganizational workflow collaboration based on local process views. In: **2008 IEEE Asia-Pacific Services Computing Conference**. [S.l.: s.n.], 2008. p. 789–794. https://doi.org/10.1109/APSCC.2008.171.

LING, S.; SCHMIDT, H. Time Petri nets for workflow modelling and analysis. In: IEEE. **Systems, Man, and Cybernetics, 2000 IEEE International Conference on**. [S.l.], 2000. v. 4, p. 3039–3044. https://doi.org/10.1109/ICSMC.2000.884464.

MANDAL, A. K.; SARKAR, A. Service oriented system design: Domain specific model based approach. In: **2016 3rd International Conference on Computer and Information Sciences (ICCOINS)**. [S.l.: s.n.], 2016. p. 489–494. https://doi.org/10.1109/ICCOINS.2016.7783264.

MARCONI, M. de A.; LAKATOS, E. M. **Metodologia Científica: Ciência e conhecimento científico; Métodos científicos; Teoria, hipóteses e variáveis; Metodologia Jurídica**. 6. ed. [S.l.]: Atlas, 2011. ISBN 8522466254.

MARTENS, A. Analyzing Web service based business processes. In: CERIOLI, M. (Ed.). **Fundamental Approaches to Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 19–33. ISBN 978-3-540-31984-9. https://doi.org/10.1007/978-3-540-31984-9_3.

MATÉ, A.; TRUJILLO, J. Tracing conceptual models' evolution in data warehouses by using the model driven architecture. **Computer Standards & Interfaces**, v. 36, n. 5, p. 831 – 843, 2014. ISSN 0920-5489. https://doi.org/10.1016/j.csi.2014.01.004.

MERLIN, P. M. **A Study of the Recoverability of Computing Systems.** Tese (Doutorado), 1974. AAI7511026.

MPMN. **Business Process Model And Notation (BPMN) - Version 2.0.2**. http://www.omg.org/spec/BPMN/2.0.2/, 2013.

MURATA, T. Petri nets: Properties, analysis and applications. In: **Proceedings of the IEEE**. [S.l.: s.n.], 1989. v. 77, n. 4, p. 541–580. https://doi.org/10.1109/5.24143.

NEJATI, S. et al. Matching and merging of statecharts specifications. In: IEEE. **29th International Conference on Software Engineering.** [S.l.], 2007. p. 54–64. https://doi.org/10.1109/ICSE.2007.50.

NGHIEM, A. **IT Web Services: A Roadmap for the Enterprise**. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130097195.

NICOLA, R. D. Behavioral equivalences. In: _____. **Encyclopedia of Parallel Computing**. Boston, MA: Springer US, 2011. p. 120–127. https://doi.org/10.1007/978-0-387-09766-4.

OLIVEIRA, K. S.; JULIA, S. Using Linear Logic to verify requirement scenarios in composite Web service. In: SPRINGER. **Brazilian Symposium on Formal Methods**. [S.l.], 2017. p. 215–232. https://doi.org/10.1007/978-3-319-70848-5_14.

_____. Using formal methods to performance verification of requirement scenarios in SOA models. In: **Information Technology-New Generations**. [S.l.]: Springer, 2019.

OLIVEIRA, K. S.; OLIVEIRA, V. F.; JULIA, S. Requirement verification in SOA models based on Interorganizational WorkFlow nets and Linear Logic. In: **Information Technology-New Generations**. [S.l.]: Springer, 2017. p. 579–587. https://doi.org/10.1007/978-3-319-54978-1_73.

_____. Using linear logic to verify requirement scenarios in SOA models based on Interorganizational WorkFlow nets relaxed sound. In: **ICEIS (2)**. [S.l.: s.n.], 2017. p. 254–262. https://doi.org/10.5220/0006290202540262.

OLIVEIRA, K. S. de; JULIA, S. Detection and removal of negative requirements of deadlock-type in Service-Oriented Architectures. In: **International Conference on Computational Science and Computational Intelligence (CSCI'20)**. [S.l.]: Conference Publishing Services, 2020.

OLIVEIRA, K. S. de; SOARES, M. S. Modeling aspects in requirements using SysML extensions. In: **Proceedings of the 15th International Conference on Enterprise Information Systems - Volume 2: ICEIS,**. [S.l.: s.n.], 2013. p. 126–133. ISBN 978-989-8565-60-0. https://doi.org/10.5220/0004419601260133.

OLIVEIRA, V. F. et al. A Linear Logic based synchronization rule for deadlock prevention in Web service composition. In: **ICEIS (2)**. [S.l.: s.n.], 2017. p. 316–323. https://doi.org/10.5220/0006308303160323.

_____. A synchronization rule based on Linear Logic for deadlock prevention in Interorganizational WorkFlow nets. In: **Information Technology-New Generations**. [S.l.]: Springer, 2017. p. 929–934. https://doi.org/10.1007/978-3-319-54978-1_119.

PAPAZOGLOU, M. P. Service-oriented computing: Concepts, characteristics and directions. In: IEEE COMPUTER SOCIETY PRESS. **Fourth International Conference on Web Information Systems Engineering**. [S.l.], 2003. p. 03–12.

PARDAL, M. L. et al. Performance assessment of XACML authorizations for supply chain traceability Web services. In: **2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)**. [S.l.: s.n.], 2012. p. 378–383. https://doi.org/10.1109/CASoN.2012.6412432.

PARK, D. Concurrency and automata on infinite sequences. In: SPRINGER-VERLAG, BERLIN, GERMANY. **5th GI-Conference on Theoretical Computer Science**. [S.l.], 1981. p. 167–183.

PASSOS, L. M. S. **A Metodology based on Linear Logic for Interorganizational Workflow Processes Analysis**. Tese (Doutorado) — Federal Univerty of Uberlândia, 2016.

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

_____. Qualitative analysis of interorganizational workflow nets using linear logic: Soundness verification. In: **2013 IEEE 25th International Conference on Tools with Artificial Intelligence**. [S.l.: s.n.], 2013. p. 667–673.

_____. Linear logic as a tool for deadlock-freeness scenarios detection in interorganizational workflow processes. In: **2014 IEEE 26th International Conference on Tools with Artificial Intelligence**. [S.l.: s.n.], 2014. p. 316–320.

_____. Deadlock-freeness scenarios detection in web service composition. In: **12th International Conference on Information Technology - New Generations**. [S.l.: s.n.], 2015. p. 780–783.

_____. Linear Logic as a tool for qualitative and quantitative analysis of Workflow processes. **International Journal on Artificial Intelligence Tools**, p. 1650008(1–25), 2016. https://doi.org/10.1142/S0218213016500081.

PETRI, C. A. **Kommunikation mit Automaten**. Tese (Doutorado) — Institut für instrumentelle Mathematik, Bonn, Germany, 1962.

POLYVYANYY, A. et al. On the Expressive Power of Behavioral Profiles. **Formal Aspects Computing**, v. 28, n. 4, p. 597–613, 2016. https://doi.org/10.1007/s00165-016-0372-4.

POOLEY, R.; ABDULLATIF, A. CPASA: continuous performance assessment of software architecture. In: IEEE. **Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on**. [S.l.], 2010. p. 79–87. https://doi.org/10.1109/ECBS.2010.16.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. [S.l.]: Feevale, 2013.

RAMCHANDANI, C. **Analysis of asynchronous concurrent systems by timed petri nets**. Tese (Doutorado) — Massachusetts Institute of Technology, Cambridge, MA, USA, 1973.

RATZER, A. V. et al. CPN tools for editing, simulating, and analysing Coloured Petri nets. In: **Proceedings of the 24th International Conference on Applications and Theory of Petri Nets**. [S.l.]: Springer-Verlag, 2003. p. 450–462. https://doi.org/10.1007/3-540-44919-1_28.

RENAUX, E.; VANWORMHOUDT, G.; TOMBELLE, C. A multiview framework driven by use cases to support the design of service components. In: **2013 IEEE Seventh International Symposium on Service-Oriented System Engineering**. [S.l.: s.n.], 2013. p. 268–273. https://doi.org/10.1109/SOSE.2013.47.

RIVIERE, N. et al. Reachability and temporal conflicts in t-time Petri nets. In: **Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM'01)**. [S.l.]: IEEE Computer Society, 2001. (PNPM '01), p. 229–238. https://doi.org/10.1109/PNPM.2001.953372.

ROSEN, M. et al. **Applied SOA: service-oriented architecture and design strategies**. [S.l.]: John Wiley & Sons, 2012. ISBN 0470223650.

RUNESON, P.; HöST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering**, Kluwer Academic Publishers, v. 14, n. 2, p. 131–164, April 2009. https://doi.org/10.1007/s10664-008-9102-8.

SAIDA, B.; MAROUA, Z.; ZAIA, A. An approach and a tool for verification of service-based inter-organizational workflows. In: **2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)**. [S.l.: s.n.], 2016. p. 1–6. https://doi.org/10.1109/RCIS.2016.7549308.

SALOMIE, I. et al. Logistic chain generation with traceability features using Web services composition. In: **2008 IEEE International Conference on Automation, Quality and Testing, Robotics**. [S.l.: s.n.], 2008. v. 1, p. 393–397. https://doi.org/10.1109/AQTR.2008.4588774.

SANGIORGI, D. On the origins of bisimulation and coinduction. **ACM Transactions on Programming Languages and Systems (TOPLAS)**, ACM New York, NY, USA, v. 31, n. 4, p. 1–41, 2009. https://doi.org/10.1145/1516507.1516510.

SEEDORF, S.; NORDHEIMER, K.; KRUG, S. Stras: A framework for semantic traceability in enterprise-wide SOA life-cycle management. In: IEEE. **2009 13th Enterprise Distributed Object Computing Conference Workshops**. [S.l.], 2009. p. 212–219. https://doi.org/10.1109/EDOCW.2009.5331994.

SENGUPTA, S.; DASGUPTA, R. Use of semi-formal and formal methods in requirement engineering of ILMS. **SIGSOFT Softw. Eng. Notes**, ACM, v. 40, n. 1, p. 1–13, feb 2015. https://doi.org/10.1145/2693208.2693235.

SYAMSIYAH, A. et al. Business process comparison: A methodology and case study. In: SPRINGER. **International Conference on Business Information Systems**. [S.l.], 2017. p. 253–267. https://doi.org/10.1007/978-3-319-59336-4_18.

ŢIPLEA, F. L.; MARINESCU, D. C. Structural soundness of workflow nets is decidable. **Information Processing Letters**, Elsevier, v. 96, n. 2, p. 54–58, 2005. https://doi.org/10.1016/j.ipl.2005.06.002.

TRČKA, N.; AALST, W. M. P. van der; SIDOROVA, N. Data-flow anti-patterns: Discovering data-flow errors in workflows. In: ECK, P. van; GORDIJN, J.; WIERINGA, R. (Ed.). **Advanced Information Systems Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 425–439. https://doi.org/10.1007/978-3-642-02144-2_34.

TRUBIANI, C.; GHABI, A.; EGYED, A. Exploiting traceability uncertainty between software architectural models and extra-functional results. **Journal of Systems and Software**, v. 125, p. 15 – 34, 2017. ISSN 0164-1212. https://doi.org/10.1016/j.jss.2016.11.032.

TSADIMAS, A. Model-based enterprise information system architectural design with SysML. In: IEEE. **Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on**. [S.l.], 2015. p. 492–497. https://doi.org/10.1109/RCIS.2015.7128911.

TSADIMAS, A.; NIKOLAIDOU, M.; ANAGNOSTOPOULOS, D. Extending SysML to explore non-functional requirements: The case of information system design. In: **Proceedings of the 27th Annual ACM Symposium on Applied Computing**. [S.l.]: ACM, 2012. p. 1057–1062. https://doi.org/10.1145/2245276.2231941.

UML. **Unified Modeling Language Specification - Version 2.5**. http://www.omg.org/spec/UML/2.5/, 2015.

VALERO, V. et al. Transforming Web services choreographies with priorities and time constraints into prioritized-time colored Petri nets. **Science of Computer Programming**, v. 77, n. 3, p. 290 – 313, 2012. ISSN 0167-6423. https://doi.org/10.1016/j.scico.2011.05.002.

VALETTE, R. Analysis of Petri nets by stepwise refinements. **Journal of Computer and System Sciences**, v. 18, n. 1, p. 35 – 46, 1979. https://doi.org/10.1016/0022-0000(79)90050-3.

VLIET, H. van. **Software Engineering: Principles and Practice**. [S.l.: s.n.], 2007. ISBN 0470031468.

WAZLAWICK, R. S. **Metodologia de Pesquisa para Ciência da Computação**. [S.l.]: Elsevier, 2009.

WEIDLICH, M.; DIJKMAN, R.; MENDLING, J. The icop framework: Identification of correspondences between process models. In: **Proceedings of the 22Nd International Conference on Advanced Information Systems Engineering**. [S.l.]: Springer-Verlag, 2010. p. 483–498.

WEIDLICH, M.; MENDLING, J.; WESKE, M. Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. **IEEE Transactions on Software Engineering**, v. 37, n. 3, p. 410–429, 2011. https://doi.org/10.1109/TSE.2010.96.

_____. A foundational approach for managing process variability. In: **Proceedings of the 23rd International Conference on Advanced Information Systems Engineering**. [S.l.]: Springer-Verlag, 2011. p. 267–282. https://doi.org/10.1007/978-3-642-21640-4_21.

WEIDLICH, M. et al. Process compliance measurement based on behavioural profiles. In: **Proceedings of the 22Nd International Conference on Advanced Information Systems Engineering**. [S.l.]: Springer-Verlag, 2010. p. 499–514. https://doi.org/10.1007/978-3-642-13094-6_38.

WEIDLICH, M.; WESKE, M.; MENDLING, J. Change Propagation in Process Models Using Behavioural Profiles. In: **IEEE International Conference on Services Computing**. [S.l.: s.n.], 2009. p. 33–40. https://doi.org/10.1109/SCC.2009.58.

WESTERGAARD, M. CPN tools 4: Multi-formalism and extensibility. In: **Application and Theory of Petri Nets and Concurrency**. [S.l.]: Springer, Berlin, Heidelberg, 2013. v. 7927, p. 400–409. https://doi.org/10.1007/978-3-642-38697-8_22.

XIONG, P.; FAN, Y.; ZHOU, M. A Petri net approach to analysis and composition of Web services. **IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans**, IEEE, v. 40, n. 2, p. 376–387, 2010. https://doi.org/10.1109/TSMCA.2009.2037018.

XIONG, P.; ZHOU, M.; PU, C. A Petri net siphon based solution to protocol-level service composition mismatches. In: **2009 IEEE International Conference on Web Services**. [S.l.: s.n.], 2009. p. 952–958. https://doi.org/10.1109/ICWS.2009.108.

YIN, R. K. **Case Study Research: Design and Methods**. 4th. ed. California, United States of America: SAGE Publications Inc., 2009. ISBN 9781412960991.

ZERNADJI, T. et al. Integrating quality requirements in engineering web service orchestrations. **Journal of Systems and Software**, Elsevier, v. 122, p. 463–483, 2016. https://doi.org/10.1016/j.jss.2015.11.009.

ZHANG, L. J.; MAO, Z. H.; ZHOU, N. Design quality analytics of traceability enablement in service-oriented solution design environment. In: IEEE. **2009 IEEE International Conference on Web Services**. [S.l.], 2009. p. 944–951. https://doi.org/10.1109/ICWS.2009.145.

Apêndice

Apêndice A

Árvores de prova da Lógica Linear geradas para o exemplo abordado nos Capítulos 3 e 4

Neste apêndice são apresentadas as árvores de prova da Lógica Linear construídas nos exemplos que ilustram a aplicação das abordagens propostas. O apêndice A.1 apresenta as árvores de prova da Lógica Linear para os exemplos que ilustram a abordagem para verificação dos cenários de requisitos comportamentais na arquitetura do tipo SOA. O apêndice A.2 apresenta as árvores de prova da Lógica Linear para os exemplos que ilustram a abordagem para verificação dos cenários de requisitos de desempenho na arquitetura do tipo SOA. As tabelas que contém as datas simbólicas de produção e de consumo são apresentadas no apêndice A.3. Já o apêndice A.4, apresenta as árvores de prova da Lógica Linear para a identificação dos dos requisitos negativos na arquitetura do tipo SOA.

A.1 Árvores de prova da Lógica Linear para o exemplo que ilustra o método para a verificação dos cenários de requisitos comportamentais em SOA

Árvore de prova para o cenário Sr2:

$$\underline{\dfrac{P_7\vdash P_7 \quad P_{11}\vdash P_{11}}{P_7,P_{11}\vdash P_7\otimes P_{11}}\otimes_R \qquad o\vdash o}\ \multimap_L$$

$$\underline{P_9\vdash P_9 \qquad P_7,P_{11},P_7\otimes P_{11}\multimap o\vdash o}\ \multimap_L$$

$$\underline{\dfrac{P_6\vdash P_6 \quad P_8\vdash P_8}{P_6,P_8\vdash P_6\otimes P_8}\otimes_R \qquad P_7,P_9,P_9\multimap P_{11},t_{10}\vdash o}\ \multimap_L$$

$$\underline{P_6,P_7,P_8,P_6\otimes P_8\multimap P_9,t_9,t_{10}\vdash o}\ \otimes_L$$

$$\underline{P_5\vdash P_5 \qquad P_6,P_7\otimes P_8,t_6,t_9,t_{10}\vdash o}\ \multimap_L$$

$$\underline{\dfrac{P_1\vdash P_1 \quad P_4\vdash P_4}{P_1,P_4\vdash P_1\otimes P_4}\otimes_R \qquad P_6,P_5,P_5\multimap P_7\otimes P_8,t_6,t_9,t_{10}\vdash o}\ \multimap_L$$

$$\underline{P_1,P_4,P_6,P_1\otimes P_4,\multimap P_5,t_5,t_6,t_9,t_{10}\vdash o}\ \otimes_L$$

$$\underline{P_3\vdash P_3 \qquad P_1,P_4\otimes P_6,P_1\otimes P_4\multimap P_5,t_5,t_6,t_9,t_{10}\vdash o}\ \multimap_L$$

$$\underline{P_1,P_3,P_1\otimes P_4\multimap P_5,P_3\multimap P_4\otimes P_6,t_5,t_6,t_9,t_{10}\vdash o}$$

$$\underline{P_2\vdash P_2 \qquad P_1,P_3,P_1\otimes P_4\multimap P_5,t_4,t_5,t_6,t_9,t_{10}\vdash o}\ \multimap_L$$

$$\underline{P_1,P_2,P_2\multimap P_3,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o}\ \otimes_L$$

$$\underline{i\vdash i \qquad P_1\otimes P_2,t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o}\ \multimap_L$$

$$i,i\multimap P_1\otimes P_2,t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o$$

Árvore de prova rotulada com as transições de disparo para o cenário Sr2:

$$\underline{\dfrac{\overset{t_5}{P_7}\vdash \overset{t_{10}}{P_7} \quad \overset{t_9}{P_{11}}\vdash \overset{t_{10}}{P_{11}}}{\overset{t_5}{P_7},\overset{t_9}{P_{11}}\vdash \overset{t_{10}}{P_7}\otimes \overset{t_{10}}{P_{11}}}\otimes_R \qquad \overset{t_{10}}{o}\vdash \overset{f_1}{o}}\ \underset{t_{10}}{\multimap_L}$$

$$\underline{\overset{t_6}{P_9}\vdash \overset{t_9}{P_9} \qquad \overset{t_5}{P_7},\overset{t_9}{P_{11}},P_7\otimes P_{11}\multimap o\vdash \overset{f_1}{o}}\ \underset{t_9}{\multimap_L}$$

$$\underline{\dfrac{\overset{t_4}{P_6}\vdash \overset{t_6}{P_6} \quad \overset{t_5}{P_8}\vdash \overset{t_6}{P_8}}{\overset{t_4}{P_6},\overset{t_5}{P_8}\vdash \overset{t_6}{P_6}\otimes \overset{t_6}{P_8}}\otimes_R \qquad \overset{t_5}{P_7},\overset{t_6}{P_9},P_9\multimap P_{11},t_{10}\vdash \overset{f_1}{o}}\ \underset{t_6}{\multimap_L}$$

$$\underline{\overset{t_4}{P_6},\overset{t_5}{P_7},\overset{t_5}{P_8},P_6\otimes P_8\multimap P_9,t_9,t_{10}\vdash \overset{f_1}{o}}\ \otimes_L$$

$$\underline{\overset{t_3}{P_5}\vdash \overset{t_5}{P_5} \qquad \overset{t_4}{P_6},\overset{t_5}{P_7}\otimes \overset{t_5}{P_8},t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \underset{t_5}{\multimap_L}$$

$$\underline{\dfrac{\overset{t_1}{P_1}\vdash \overset{t_3}{P_1} \quad \overset{t_4}{P_4}\vdash \overset{t_3}{P_4}}{\overset{t_1}{P_1},\overset{t_4}{P_4}\vdash \overset{t_3}{P_1}\otimes \overset{t_3}{P_4}}\otimes_R \qquad \overset{t_4}{P_6},\overset{t_3}{P_5},P_5\multimap P_7\otimes P_8,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \underset{t_3}{\multimap_L}$$

$$\underline{\overset{t_1}{P_1},\overset{t_4}{P_4},\overset{t_4}{P_6},P_1\otimes P_4,\multimap P_5,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \otimes_L$$

$$\underline{\overset{t_2}{P_3}\vdash \overset{t_4}{P_3} \qquad \overset{t_1}{P_1},\overset{t_4}{P_4}\otimes \overset{t_4}{P_6},P_1\otimes P_4\multimap P_5,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \underset{t_4}{\multimap_L}$$

$$\underline{\overset{t_1}{P_1},\overset{t_2}{P_3},P_1\otimes P_4\multimap P_5,P_3\multimap P_4\otimes P_6,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}$$

$$\underline{\overset{t_1}{P_2}\vdash \overset{t_2}{P_2} \qquad \overset{t_1}{P_1},\overset{t_2}{P_3},P_1\otimes P_4\multimap P_5,t_4,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \underset{t_2}{\multimap_L}$$

$$\underline{\overset{t_1}{P_1},\overset{t_1}{P_2},P_2\multimap P_3,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \otimes_L$$

$$\underline{\overset{i_1}{i}\vdash \overset{t_1}{i} \qquad \overset{t_1}{P_1}\otimes \overset{t_1}{P_2},t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}}\ \underset{t_1}{\multimap_L}$$

$$\overset{i_1}{i},i\multimap P_1\otimes P_2,t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash \overset{f_1}{o}$$

Árvore de prova para o cenário Sa1:

$$\underline{\dfrac{C_{10}\vdash C_{10} \quad S_{10}\vdash S_{10}}{C_{10},S_{10}\vdash C_{10}\otimes S_{10}}\otimes_R \quad o\vdash o}\ \multimap_L$$

$$\underline{\dfrac{CP_4\vdash CP_4 \quad S_9\vdash S_9}{CP_4,S_9\vdash CP_4\otimes S_9}\otimes_R \quad C_{10},C_{10},C_{10}\otimes C_{10}\multimap o\vdash o}\ \multimap_L$$

$$\underline{S_5\vdash S_5 \quad CP_4,C_{10},S_9,S_9\otimes CP_4\multimap S_{10},t_o\vdash o}\ \multimap_L$$

$$\underline{S_5,CP_4,C_{10},S_5\multimap S_9,t_8,t_o\vdash o}\ \otimes_L$$

$$\underline{\dfrac{C_6\vdash C_6 \quad C_7\vdash C_7}{C_6,C_7\vdash C_6\otimes C_7}\otimes_R \quad S_5,CP_4\otimes C_{10},t_{17},t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{C'_4\vdash C'_4 \quad S_5,C_6,C_7,C_6\otimes C_7\multimap CP_4\otimes C_{10},t_{17},t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{C_5\vdash C_5 \quad C'_4,S_5,C_6,C'_4\multimap C_7,t_{17},t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{\dfrac{C_2\vdash C_2 \quad CP_2\vdash CP_2}{C_2,CP_2\vdash C_2\otimes CP_2}\otimes_R \quad C'_4,S_5,C_5,C_5\multimap C_6,t_{17},t_{20},t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{C_2,C'_4,CP_2,S_5,C_2\otimes CP_2\multimap C_5,t_{13},t_{17},t_{20},t_7,t_8,t_o\vdash o}\ \otimes_L$$

$$\underline{S_4\vdash S_4 \quad C_2,C'_4,CP_2\otimes S_5,t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{S_3\vdash S_3 \quad C_2,C'_4,S_4,S_4\multimap CP_2\otimes S_5,t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{S_2\vdash S_2 \quad C_2,C'_4,S_3,S_3\multimap S_4,t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{C'_3\vdash C'_3 \quad C_2,S_2,C'_4,S_2\multimap S_3,t_{12},t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{\dfrac{S_1\vdash S_1 \quad CP_1\vdash CP_1}{S_1,CP_1\vdash S_1\otimes CP_1}\otimes_R \quad C_2,C'_3,S_2,C'_3\multimap C'_4,t_{11},t_{12},t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{S_1,CP_1,C_2,C'_3,S_1\otimes CP_1\multimap S_2,t_{11},t_{12},t_{13},t_{17},t'_{18},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \otimes_L$$

$$\underline{C_1\vdash C_1 \quad S_1,CP_1\otimes C_2\otimes C'_3,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$\underline{C_1,S_1,C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \otimes_L$$

$$\underline{i\vdash i \quad C_1\otimes S_1,t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\ \multimap_L$$

$$i,i\multimap C_1\otimes S_1,t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o$$

Árvore de prova rotulada com as transições de disparo para o cenário Sa1:

Árvore de prova para o cenário Sa2:

$$\dfrac{\dfrac{S_{10}\vdash S_{10} \quad C_{10}\vdash C_{10}}{S_{10},C_{10}\vdash C_{10}\otimes S_{10}}\otimes_R \quad o\vdash o}{S_{10},C_{10},C_{10}\otimes S_{10}\multimap o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_7\vdash C_7 \quad C_9\vdash C_9 \quad CP_5\vdash CP_5}{C_7,C_9,CP_5\vdash C_9\otimes CP_5\otimes C_7}\otimes_R \quad S_{10},C_{10},C_{10}\otimes S_{10}\multimap o\vdash o}{C_7,C_9,CP_5,S_{10},C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o\vdash o}\multimap_L$$

$$\dfrac{C_7,C_9,CP_5,S_{10},C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o\vdash o}{S_8\vdash S_8 \quad C_7,C_9,CP_5\otimes S_{10},C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o,\vdash o}\otimes_L$$

$$\dfrac{S_8\vdash S_8 \quad C_7,C_9,CP_5\otimes S_{10},C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o,\vdash o}{C_8\vdash C_8 \quad C_7,S_8,C_9,S_8\multimap CP_5\otimes S_{10},t_{10},t_o,\vdash o}\multimap_L$$

$$\dfrac{C_8\vdash C_8 \quad C_7,S_8,C_9,S_8\multimap CP_5\otimes S_{10},t_{10},t_o,\vdash o}{S_7\vdash S_7 \quad C_7,C_8,S_8,C_8\multimap C_9,t_{10},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{S_7\vdash S_7 \quad C_7,C_8,S_8,C_8\multimap C_9,t_{10},t_9,t_o,\vdash o}{\dfrac{S_6\vdash S_6 \quad CP_3\vdash CP_3}{S_6,CP_3\vdash CP_3\otimes S_6}\otimes_R \quad C_7,C_8,S_7,S_7\multimap S_8,t_{10},t_{14},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_6\vdash S_6 \quad CP_3\vdash CP_3}{S_6,CP_3\vdash CP_3\otimes S_6}\otimes_R \quad C_7,C_8,S_7,S_7\multimap S_8,t_{10},t_{14},t_9,t_o,\vdash o}{S_5\vdash S_5 \quad C_7,CP_3,C_8,S_6,CP_3\otimes S_6\multimap S_7,t_{10},t_{14},t_{16},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{S_5\vdash S_5 \quad C_7,CP_3,C_8,S_6,CP_3\otimes S_6\multimap S_7,t_{10},t_{14},t_{16},t_9,t_o,\vdash o}{S_5,C_7,CP_3,C_8,S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}\otimes_L$$

$$\dfrac{S_5,C_7,CP_3,C_8,S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}{C_6\vdash C_6 \quad S_5,C_7,CP_3\otimes C_8,S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{C_6\vdash C_6 \quad S_5,C_7,CP_3\otimes C_8,S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}{C'_4\vdash C'_4 \quad S_5,C_6,C_7,C_6\multimap CP_3\otimes C_8,t_{10},t_{14},t_{16},t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C'_4\vdash C'_4 \quad S_5,C_6,C_7,C_6\multimap CP_3\otimes C_8,t_{10},t_{14},t_{16},t_6,t_9,t_o,t_{15}\vdash o}{C_5\vdash C_5 \quad C'_4,S_5,C_6,C'_4\multimap C_7,t_{10},t_{14},t_{16},t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_5\vdash C_5 \quad C'_4,S_5,C_6,C'_4\multimap C_7,t_{10},t_{14},t_{16},t_5,t_6,t_9,t_o,t_{15}\vdash o}{\dfrac{C_2\vdash C_2 \quad CP_2\vdash CP_2}{C_2,CP_2\vdash C_2\otimes CP_2}\otimes_R \quad C'_4,S_5,C_5,C_5\multimap C_6,t_{10},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_2\vdash C_2 \quad CP_2\vdash CP_2}{C_2,CP_2\vdash C_2\otimes CP_2}\otimes_R \quad C'_4,S_5,C_5,C_5\multimap C_6,t_{10},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2,C'_4,CP_2,S_5,C_2\otimes CP_2\multimap C_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_2,C'_4,CP_2,S_5,C_2\otimes CP_2\multimap C_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_4\vdash S_4 \quad C_2,C'_4,CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{S_4\vdash S_4 \quad C_2,C'_4,CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_3\vdash S_3 \quad C_2,C'_4,S_4,S_4\multimap CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{S_3\vdash S_3 \quad C_2,C'_4,S_4,S_4\multimap CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_2\vdash S_2 \quad C_2,C'_4,S_3,S_3\multimap S_4,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{S_2\vdash S_2 \quad C_2,C'_4,S_3,S_3\multimap S_4,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C'_3\vdash C'_3 \quad C_2,S_2,C'_4,S_2\multimap S_3,t_{10},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C'_3\vdash C'_3 \quad C_2,S_2,C'_4,S_2\multimap S_3,t_{10},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{\dfrac{S_1\vdash S_1 \quad CP_1\vdash CP_1}{S_1,CP_1\vdash S_1\otimes CP_1}\otimes_R \quad C_2,C'_3,S_2,C'_3\multimap C'_4,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_1\vdash S_1 \quad CP_1\vdash CP_1}{S_1,CP_1\vdash S_1\otimes CP_1}\otimes_R \quad C_2,C'_3,S_2,C'_3\multimap C'_4,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_1,CP_1,C_2,C'_3,S_1\otimes CP_1\multimap S_2,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{S_1,CP_1,C_2,C'_3,S_1\otimes CP_1\multimap S_2,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_1\vdash C_1 \quad S_1,CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_1\vdash C_1 \quad S_1,CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_1,S_1,C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{C_1,S_1,C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{i\vdash i \quad C_1\otimes S_1,t_1,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$i,i\multimap C_1\otimes S_1,t_1,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o$$

Árvore de prova rotulada com as transições de disparo para o cenário Sa2:

Árvore de prova para o cenário Sa3:

$$\underline{C_8 \vdash C_8 \quad C_7, CP_3, S_9, C_9, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{S_5 \vdash S_5 \quad C_7, CP_3, C_8, S_9, C_8 \multimap C_9, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{S_5, C_7, CP_3, C_8, S_5 \multimap S_9, t_{14}, t_{10}, t_8, t_o \vdash o}\ \otimes_L$$

$$\underline{C_6 \vdash C_6 \quad S_5, C_7, CP_3 \otimes C_8, S_5 \multimap S_9, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{C'_4 \vdash C'_4 \quad S_5, C_6, C_7, C_6 \multimap CP_3 \otimes C_8, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{C_5 \vdash C_5 \quad C'_4, S_5, C_6, C'_4 \multimap C_7, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{\dfrac{C_2 \vdash C_2 \quad CP_2 \vdash CP_2}{C_2, CP_2 \vdash C_2 \otimes CP_2}\ \otimes_R \quad C'_4, S_5, C_5, C_5 \multimap C_6, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{C_2, C'_4, CP_2, S_5, C_2 \otimes CP_2 \multimap C_5, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \otimes_L$$

$$\underline{S_4 \vdash S_4 \quad C_2, C'_4, CP_2 \otimes S_5, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{S_3 \vdash S_3 \quad C_2, C'_4, S_4, S_4 \multimap CP_2 \otimes S_5, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{S_2 \vdash S_2 \quad C_2, C'_4, S_3, S_3 \multimap S_4, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{C'_3 \vdash C'_3 \quad C_2, S_2, C'_4, S_2 \multimap S_3, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{\dfrac{S_1 \vdash S_1 \quad CP_1 \vdash CP_1}{S_1, CP_1 \vdash S_1 \otimes CP_1}\ \otimes_R \quad C_2, C'_3, S_2, C'_3 \multimap C'_4, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{S_1, CP_1, C_2, C'_3, S_1 \otimes CP_1 \multimap S_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \otimes_L$$

$$\underline{C_1 \vdash C_1 \quad S_1, CP_1 \otimes C_2 \otimes C'_3, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$\underline{C_1, S_1, C_1 \multimap CP_1 \otimes C_2 \otimes C'_3, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \otimes_L$$

$$\underline{i \vdash i \quad C_1 \otimes S_1, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o}\ \multimap_L$$

$$i, i \multimap C_1 \otimes S_1, t_1, t_2, t'_{18}, t_{11}, t_{12}, t_4, t_3, t_{13}, t_{20}, t_5, t_{17}, t_{14}, t_{10}, t_8, t_o \vdash o$$

Árvore de prova para o cenário Sa4:

$$\frac{S_5 \vdash S_5 \quad CP_4,C_{10},S_6,t_6,t_{16},t_9,t_o \vdash o}{S_5,CP_4,C_{10},S_5 \multimap S_6,t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{S_5,CP_4,C_{10},S_5 \multimap S_6,t_6,t_{16},t_9,t_o \vdash o}{S_5,CP_4 \otimes C_{10},S_5 \multimap S_6,t_6,t_{16},t_9,t_o \vdash o}\otimes_L$$

$$\frac{\dfrac{C_6 \vdash C_6 \quad C_7 \vdash C_7}{C_6,C_7 \vdash C_6 \otimes C_7}\otimes_R \quad S_5,CP_4 \otimes C_{10},S_5 \multimap S_6,t_6,t_{16},t_9,t_o \vdash o}{S_5,C_6,C_7,C_6 \otimes C_7 \multimap CP_4 \otimes C_{10},t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{C'_4 \vdash C'_4 \quad S_5,C_6,C_7,C_6 \otimes C_7 \multimap CP_4 \otimes C_{10},t_{15},t_6,t_{16},t_9,t_o \vdash o}{C'_4,S_5,C_6,C'_4 \multimap C_7,t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{C_5 \vdash C_5 \quad C'_4,S_5,C_6,C'_4 \multimap C_7,t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C'_4,S_5,C_5,C_5 \multimap C_6,t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{\dfrac{C_2 \vdash C_2 \quad CP_2 \vdash CP_2}{C_2,CP_2 \vdash C_2 \otimes CP_2}\otimes_R \quad C'_4,S_5,C_5,C_5 \multimap C_6,t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,C'_4,CP_2,S_5,C_2 \otimes CP_2 \multimap C_5,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{C_2,C'_4,CP_2,S_5,C_2 \otimes CP_2 \multimap C_5,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,C'_4,CP_2 \otimes S_5,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\otimes_L$$

$$\frac{S_4 \vdash S_4 \quad C_2,C'_4,CP_2 \otimes S_5,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,C'_4,S_4,S_4 \multimap CP_2 \otimes S_5,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{S_3 \vdash S_3 \quad C_2,C'_4,S_4,S_4 \multimap CP_2 \otimes S_5,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,C'_4,S_3,S_3 \multimap S_4,t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{S_2 \vdash S_2 \quad C_2,C'_4,S_3,S_3 \multimap S_4,t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,S_2,C'_4,S_2 \multimap S_3,t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{C'_3 \vdash C'_3 \quad C_2,S_2,C'_4,S_2 \multimap S_3,t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_2,C'_3,S_2,C'_3 \multimap C'_4,t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{\dfrac{S_1 \vdash S_1 \quad CP_1 \vdash CP_1}{S_1,CP_1 \vdash S_1 \otimes CP_1}\otimes_R \quad C_2,C'_3,S_2,C'_3 \multimap C'_4,t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{S_1,CP_1,C_2,C'_3,S_1 \otimes CP_1 \multimap S_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{S_1,CP_1,C_2,C'_3,S_1 \otimes CP_1 \multimap S_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{S_1,CP_1 \otimes C_2 \otimes C'_3,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\otimes_L$$

$$\frac{C_1 \vdash C_1 \quad S_1,CP_1 \otimes C_2 \otimes C'_3,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_1,S_1,C_1 \multimap CP_1 \otimes C_2 \otimes C'_3,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

$$\frac{C_1,S_1,C_1 \multimap CP_1 \otimes C_2 \otimes C'_3,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{C_1 \otimes S_1,t_1,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\otimes_L$$

$$\frac{i \vdash i \quad C_1 \otimes S_1,t_1,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}{i,i \multimap C_1 \otimes S_1,t_1,t_2,t'_{18},t_{11},t_{12},t_4,t_3,t_{13},t_{20},t_7,t_{15},t_6,t_{16},t_9,t_o \vdash o}\multimap_L$$

A.2 Árvores de prova da Lógica Linear para o exemplo que ilustra o método para a verificação dos cenários de requisitos de desempenho em SOA

Por razões de espaço, algumas partes das datas simbólicas serão representadas por sequências. Considerando $Seq_2 = D_i+d_1+d_2+d_4+d_3$, a árvore de prova com datas simbólicas para o cenário Sr2 é:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P_7(Seq_2+d_5,Seq_2+d_5+d_6+d_9+d_{10})\vdash P_7 \quad P_{11}(Seq_2+d_5+d_6+d_9,Seq_2+d_5+d_6+d_9+d_{10})\vdash P_{11}}{P_7(Seq_2+d_5,Seq_2+d_5+d_6+d_9+d_{10}),P_{11}(Seq_2+d_5+d_6+d_9,Seq_2+d_5+d_6+d_9+d_{10})\vdash P_7\otimes P_{11}}\otimes_R \quad o(Seq_2+d_5+d_6+d_9+d_{10},.)\vdash o}{P_9(Seq_2+d_5+d_6,Seq_2+d_5+d_6+d_9)\vdash P_9 \quad P_7(Seq_2+d_5,.),P_{11}(Seq_2+d_5+d_6+d_9,.),P_7\otimes P_{11}\multimap o\vdash o}\multimap_L}{\dfrac{P_6(D_i+d_1+d_2+d_4,Seq_2+d_5+d_6)\vdash P_6 \quad P_8(Seq_2+d_5,Seq_2+d_5+d_6)\vdash P_8}{P_6(D_i+d_1+d_2+d_4,Seq_2+d_5+d_6),P_8(Seq_2+d_5,Seq_2+d_5+d_6)\vdash P_6\otimes P_8}\otimes_R \quad P_7(Seq_2+d_5,.),P_9(Seq_2+d_5+d_6,.),P_9\multimap P_{11},t_{10}\vdash o}\multimap_L}{P_6(D_i+d_1+d_2+d_4,.),P_7(Seq_2+d_5,.),P_8(Seq_2+d_5,.),P_6\otimes P_8\multimap P_9,t_9,t_{10}\vdash o}\otimes_L}{P_5(Seq_2,Seq_2+d_5)\vdash P_5 \quad P_6(D_i+d_1+d_2+d_4,.),P_7(Seq_2+d_5,.)\otimes P_8(Seq_2+d_5,.),t_6,t_9,t_{10}\vdash o}\multimap_L}{\dfrac{P_1(D_i+d_1,D_i+d_1+d_2+d_4+d_3)\vdash P_1 \quad P_4(D_i+d_1+d_2+d_4,Seq_2)\vdash P_4}{P_1(D_i+d_1,D_i+d_1+d_2+d_4+d_3),P_4(D_i+d_1+d_2+d_4,Seq_2)\vdash P_1\otimes P_4}\otimes_R \quad P_6(D_i+d_1+d_2+d_4,.),P_5(Seq_2,.),P_5\multimap P_7\otimes P_8,t_6,t_9,t_{10}\vdash o}\multimap_L}{P_1(D_i+d_1,.),P_4(D_i+d_1+d_2+d_4,.),P_6(D_i+d_1+d_2+d_4,.),P_1\otimes P_4,\multimap P_5,t_5,t_6,t_9,t_{10}\vdash o}\otimes_L}{P_3(D_i+d_1+d_2,D_i+d_1+d_2+d_4)\vdash P_3 \quad P_1(D_i+d_1,.),P_4(D_i+d_1+d_2+d_4,.)\otimes P_6(D_i+d_1+d_2+d_4,.),P_1\otimes P_4\multimap P_5,t_5,t_6,t_9,t_{10}\vdash o}\multimap_L}{P_1(D_i+d_1,.),P_3(D_i+d_1+d_2,.),P_1\otimes P_4\multimap P_5,P_3\multimap P_4\otimes P_6,t_5,t_6,t_9,t_{10}\vdash o}$$

$$\dfrac{\dfrac{P_2(D_i+d_1,D_i+d_1+d_2)\vdash P_2 \quad P_1(D_i+d_1,.),P_3(D_i+d_1+d_2,.),P_1\otimes P_4\multimap P_5,t_4,t_5,t_6,t_9,t_{10}\vdash o}{P_1(D_i+d_1,.),P_2(D_i+d_1,.),P_2\multimap P_3,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o}\otimes_L}{\dfrac{i(D_i,D_i+d_1)\vdash i \quad P_1(D_i+d_1,.)\otimes P_2(D_i+d_1,.),t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o}{i(D_i,.),i\multimap P_1\otimes P_2,t_2,t_3,t_4,t_5,t_6,t_9,t_{10}\vdash o}\multimap_L}\multimap_L$$

Considerando $Seq_3 = D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$, $Seq_4 = D_i + d_i + d_1 + d_{18}$, $Seq_5 = D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7$ e $Seq_6 = D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17}, max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7) + d_8$, a árvore de prova com datas simbólicas para o cenário Sa1 é:

$$\dfrac{\dfrac{C_{10}(Seq_5,Seq_6+d_o)\vdash C_{10} \quad S_{10}(Seq_6,Seq_6+d_o)\vdash S_{10}}{C_{10}(Seq_5,Seq_6+d_o),S_{10}(Seq_6,Seq_6+d_o)\vdash C_{10}\otimes S_{10}}\otimes_R \quad o(Seq_6+d_o,.)\vdash o}{C_{10}(Seq_5,.),S_{10}(Seq_6,.),C_{10}\otimes S_{10}\multimap o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{CP_4(Seq_5,Seq_6)\vdash CP_4 \quad S_9(Seq_3+d_{17},Seq_6)\vdash S_9}{CP_4(Seq_5,Seq_6),S_9(Seq_3+d_{17},Seq_6)\vdash CP_4\otimes S_9}\otimes_R \quad C_{10}(Seq_5,.),S_{10}(Seq_6,.),C_{10}\otimes S_{10}\multimap o\vdash o}{S_5(Seq_3,Seq_3+d_{17})\vdash S_5 \quad CP_4(Seq_5,.),C_{10}(Seq_5,.),S_9(Seq_3+d_{17},.),S_9\otimes CP_4\multimap S_{10},t_o\vdash o}\multimap_L$$

$$\dfrac{S_5(Seq_3,.),CP_4(Seq_5,.),C_{10}(Seq_5,.),S_5\multimap S_9,t_8,t_o\vdash o}{S_5(Seq_3,.),CP_4(Seq_5,.),C_{10}(Seq_5,.),S_5\multimap S_9,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{\dfrac{C_6(Seq_3+d_3+d_{13},Seq_5)\vdash C_6 \quad C_7(Seq_4+d_{20},Seq_5)\vdash C_7}{C_6(Seq_3+d_3+d_{13},Seq_5),C_7(Seq_4+d_{20},Seq_5)\vdash C_6\otimes C_7}\otimes_R \quad S_5(Seq_3,.),CP_4(Seq_5,.)\otimes C_{10}(Seq_5,.),t_{17},t_8,t_o\vdash o}{C'_4(Seq_4,Seq_4+d_{20})\vdash C'_4 \quad S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C_7(Seq_4+d_{20},.),C_6\otimes C_7\multimap CP_4\otimes C_{10},t_{17},t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{C_5(Seq_3+d_3,Seq_3+d_3+d_{13})\vdash C_5 \quad C'_4(Seq_4,.),S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C'_4\multimap C_7,t_{17},t_7,t_8,t_o\vdash o}{C'_4(Seq_4,Seq_4+d_{20})\vdash C'_4 \quad S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C_7(Seq_4+d_{20},.),C_6\otimes C_7\multimap CP_4\otimes C_{10},t_{17},t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_2(D_i+d_i+d_1,Seq_3+d_3)\vdash C_2 \quad CP_2(Seq_3,Seq_3+d_3)\vdash CP_2}{C_2(D_i+d_i+d_1,Seq_3+d_3),CP_2(Seq_3,Seq_3+d_3)\vdash C_2\otimes CP_2}\otimes_R \quad C'_4(Seq_4,.),S_5(Seq_3,.),C_5(Seq_3+d_3,.),C_5\multimap C_6,t_{17},t_{20},t_7,t_8,t_o\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.),S_5(Seq_3,.),C_2\otimes CP_2\multimap C_5,t_{13},t_{17},t_{20},t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},Seq_3)\vdash S_4 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.)\otimes S_5(Seq_3,.),t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.),S_5(Seq_3,.),C_2\otimes CP_2\multimap C_5,t_{13},t_{17},t_{20},t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{S_3(D_i+d_i+d_1+d_2+d_{11},D_i+d_i+d_1+d_2+d_{11}+d_{12})\vdash S_3 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},.),S_4\multimap CP_2\otimes S_5,t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}{S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},Seq_3)\vdash S_4 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.)\otimes S_5(Seq_3,.),t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{S_2(D_i+d_i+d_1+d_2,D_i+d_i+d_1+d_2+d_{11})\vdash S_2 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_3(D_i+d_i+d_1+d_2+d_{11},.),S_3\multimap S_4,t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{S_3(D_i+d_i+d_1+d_2+d_{11},D_i+d_i+d_1+d_2+d_{11}+d_{12})\vdash S_3 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},.),S_4\multimap CP_2\otimes S_5,t_{13},t_{17},t_{20},t_3,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{C'_3(D_i+d_i+d_1,Seq_4)\vdash C'_3 \quad C_2(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_4(Seq_4,.),S_2\multimap S_3,t_{12},t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{S_2(D_i+d_i+d_1+d_2,D_i+d_i+d_1+d_2+d_{11})\vdash S_2 \quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_3(D_i+d_i+d_1+d_2+d_{11},.),S_3\multimap S_4,t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_1(D_i+d_i,D_i+d_i+d_1+d_2)\vdash S_1 \quad CP_1(D_i+d_i+d_1,D_i+d_i+d_1+d_2)\vdash CP_1}{S_1(D_i+d_i,D_i+d_i+d_1+d_2),CP_1(D_i+d_i+d_1,D_i+d_i+d_1+d_2)\vdash S_1\otimes CP_1}\otimes_R \quad C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_3\multimap C'_4,t_{11},t_{12},t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{C'_3(D_i+d_i+d_1,Seq_4)\vdash C'_3 \quad C_2(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_4(Seq_4,.),S_2\multimap S_3,t_{12},t_{13},t_{17},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.),C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_1\otimes CP_1\multimap S_2,t_{11},t_{12},t_{13},t_{17},t'_{18},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.),C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_1\otimes CP_1\multimap S_2,t_{11},t_{12},t_{13},t_{17},t'_{18},t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{C_1(D_i+d_i,D_i+d_i+d_1)\vdash C_1 \quad S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.)\otimes C_2(D_i+d_i+d_1,.)\otimes C'_3(D_i+d_i+d_1,.),t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{C_1(D_i+d_i,.),S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.)\otimes C_2(D_i+d_i+d_1,.)\otimes C'_3(D_i+d_i+d_1,.),t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{C_1(D_i+d_i,.),S_1(D_i+d_i,.),C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{C_1(D_i+d_i,.),S_1(D_i+d_i,.),C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{i(D_i,D_i+d_i)\vdash i \quad C_1(D_i+d_i,.)\otimes S_1(D_i+d_i,.),t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}{i(D_i,.),i\multimap C_1\otimes S_1,t_1,t_{11},t_{12},t_{13},t_{17},t'_{18},t_2,t_{20},t_3,t_4,t_7,t_8,t_o\vdash o}\multimap_L$$

Considerando Seq_3 e Seq_4 já definidas anteriormente e também as sequências $Seq_7 = D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} +$

$d_5, d_{15}) + d_6$ e $Seq_8 = D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14},$ a árvore de prova com datas simbólicas para o cenário Sa2 é:

$$\dfrac{\dfrac{S_{10}(Seq_7+d_{16}+d_9,Seq_8+d_o)\vdash S_{10}\quad C_{10}(Seq_8,Seq_8+d_o)\vdash C_{10}}{S_{10}(Seq_7+d_{16}+d_9,Seq_8+d_o),C_{10}(Seq_8,Seq_8 d_o)\vdash C_{10}\otimes S_{10}}\otimes_R \quad o(Seq_8,.)\vdash o}{S_{10}(Seq_7+d_{16}+d_9,.),C_{10}(Seq_8,.),C_{10}\otimes S_{10}\multimap o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_7(Seq_4+d_{20},Seq_8)\vdash C_7\quad C_9(Seq_3+d_3+d_{13}+d_5+d_{14},Seq_8)\vdash C_9\quad CP_5(Seq_7+d_{16}+d_9,Seq_8)\vdash CP_5}{C_7(Seq_4+d_{20},Seq_8),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},Seq_8),CP_5(Seq_7+d_{16}+d_9,Seq_8)\vdash C_9\otimes CP_5\otimes C_7}\otimes_R \quad S_{10}(Seq_7+d_{16}+d_9,.),C_{10}(Seq_8,.),C_{10}\otimes S_{10}\multimap o\vdash o}{C_7(Seq_4+d_{20},.),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},.),CP_5(Seq_7+d_{16}+d_9,.),S_{10}(Seq_7+d_{16}+d_9,.),C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o\vdash o}\otimes_L$$

$$\dfrac{S_8(Seq_7+d_{16},Seq_7+d_{16}+d_9)\vdash S_8\quad C_7(Seq_4+d_{20},.),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},.),CP_5(Seq_7+d_{16}+d_9,.)\otimes S_{10}(Seq_7+d_{16}+d_9,.),C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o,\vdash o}{C_7(Seq_4+d_{20},.),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},.),CP_5(Seq_7+d_{16}+d_9,.),S_{10}(Seq_7+d_{16}+d_9,.),C_9\otimes CP_5\otimes C_7\multimap C_{10},t_o\vdash o}\multimap_L$$

$$\dfrac{C_8(Seq_3+d_3+d_{13}+d_5,Seq_3+d_3+d_{13}+d_5+d_{14})\vdash C_8\quad C_7(Seq_4+d_{20},.),S_8(Seq_7+d_{16},.),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},.),S_8\multimap CP_5\otimes S_{10},t_{10},t_o,\vdash o}{C_7(Seq_4+d_{20},.),S_8(Seq_7+d_{16},.),C_9(Seq_3+d_3+d_{13}+d_5+d_{14},.),S_8\multimap CP_5\otimes S_{10},t_{10},t_o,\vdash o}\multimap_L$$

$$\dfrac{S_7(Seq_7,Seq_7+d_{16})\vdash S_7\quad C_7(Seq_4+d_{20},.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_8(Seq_7+d_{16},.),C_8\multimap C_9,t_{10},t_9,t_o,\vdash o}{C_7(Seq_4+d_{20},.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_8(Seq_7+d_{16},.),C_8\multimap C_9,t_{10},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_6(Seq_3+d_{15},Seq_7)\vdash S_6\quad CP_3(Seq_3+d_3+d_{13}+d_5,Seq_7)\vdash CP_3}{S_6(Seq_3+d_{15},Seq_7),CP_3(Seq_3+d_3+d_{13}+d_5,Seq_7)\vdash CP_3\otimes S_6}\otimes_R \quad C_7(Seq_4+d_{20},.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_7(Seq_7,.),S_7\multimap S_8,t_{10},t_{14},t_9,t_o,\vdash o}{C_7(Seq_4+d_{20},.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_7(Seq_7,.),S_7\multimap S_8,t_{10},t_{14},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{S_5(Seq_3,Seq_3+d_{15})\vdash S_5\quad C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_6(Seq_3+d_{15},.),CP_3\otimes S_6\multimap S_7,t_{10},t_{14},t_{16},t_9,t_o,\vdash o}{C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_6(Seq_3+d_{15},.),CP_3\otimes S_6\multimap S_7,t_{10},t_{14},t_{16},t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{S_5(Seq_3,.),C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}{C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.),C_8(Seq_3+d_3+d_{13}+d_5,.),S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}\otimes_L$$

$$\dfrac{C_6(Seq_3+d_3+d_{13},Seq_3+d_3+d_{13}+d_5)\vdash C_6\quad S_5(Seq_3,.),C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.)\otimes C_8(Seq_3+d_3+d_{13}+d_5,.),S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}{S_5(Seq_3,.),C_7(Seq_4+d_{20},.),CP_3(Seq_3+d_3+d_{13}+d_5,.)\otimes C_8(Seq_3+d_3+d_{13}+d_5,.),S_5\multimap S_6,t_{10},t_{14},t_{16},t_6,t_9,t_o,\vdash o}\multimap_L$$

$$\dfrac{C'_4(Seq_4,Seq_4+d_{20})\vdash C'_4\quad S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C_7(Seq_4+d_{20},.),C_6\multimap CP_3\otimes C_8,t_{10},t_{14},t_{16},t_6,t_9,t_o,t_{15}\vdash o}{S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C_7(Seq_4+d_{20},.),C_6\multimap CP_3\otimes C_8,t_{10},t_{14},t_{16},t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_5(Seq_3+d_3,Seq_3+d_3+d_{13})\vdash C_5\quad C'_4(Seq_4,.),S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C'_4\multimap C_7,t_{10},t_{14},t_{16},t_5,t_6,t_9,t_o,t_{15}\vdash o}{C'_4(Seq_4,.),S_5(Seq_3,.),C_6(Seq_3+d_3+d_{13},.),C'_4\multimap C_7,t_{10},t_{14},t_{16},t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_2(D_i+d_i+d_1,Seq_3+d_3)\vdash C_2\quad CP_2(Seq_3,Seq_3+d_3)\vdash CP_2}{C_2(D_i+d_i+d_1,Seq_3+d_3),CP_2(Seq_3,Seq_3+d_3)\vdash C_2\otimes CP_2}\otimes_R \quad C'_4(Seq_4,.),S_5(Seq_3,.),C_5(Seq_3+d_3,.),C_5\multimap C_6,t_{10},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}{C'_4(Seq_4,.),S_5(Seq_3,.),C_5(Seq_3+d_3,.),C_5\multimap C_6,t_{10},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.),S_5(Seq_3,.),C_2\otimes CP_2\multimap C_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.),S_5(Seq_3,.),C_2\otimes CP_2\multimap C_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},Seq_3)\vdash S_4\quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.)\otimes S_5(Seq_3,.),t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),CP_2(Seq_3,.)\otimes S_5(Seq_3,.),t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{S_3(D_i+d_i+d_1+d_2+d_{11},D_i+d_i+d_1+d_2+d_{11}+d_{12})\vdash S_3\quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},.),S_4\multimap CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_4(D_i+d_i+d_1+d_2+d_{11}+d_{12},.),S_4\multimap CP_2\otimes S_5,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{S_2(D_i+d_i+d_1+d_2,D_i+d_i+d_1+d_2+d_{11})\vdash S_2\quad C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_3(D_i+d_i+d_1+d_2+d_{11},.),S_3\multimap S_4,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),C'_4(Seq_4,.),S_3(D_i+d_i+d_1+d_2+d_{11},.),S_3\multimap S_4,t_{10},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C'_3(D_i+d_i+d_1,Seq_4)\vdash C'_3\quad C_2(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_4(Seq_4,.),S_2\multimap S_3,t_{10},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_4(Seq_4,.),S_2\multimap S_3,t_{10},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_1(D_i+d_i,D_i+d_i+d_1+d_2)\vdash S_1\quad CP_1(D_i+d_i+d_1,D_i+d_i+d_1+d_2)\vdash CP_1}{S_1(D_i+d_i,D_i+d_i+d_1+d_2),CP_1(D_i+d_i+d_1,D_i+d_i+d_1+d_2)\vdash S_1\otimes CP_1}\otimes_R \quad C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_3\multimap C'_4,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_2(D_i+d_i+d_1+d_2,.),C'_3\multimap C'_4,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.),C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_1\otimes CP_1\multimap S_2,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.),C_2(D_i+d_i+d_1,.),C'_3(D_i+d_i+d_1,.),S_1\otimes CP_1\multimap S_2,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{C_1(D_i+d_i,D_i+d_i+d_1)\vdash C_1\quad S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.)\otimes C_2(D_i+d_i+d_1,.)\otimes C'_3(D_i+d_i+d_1,.),t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{S_1(D_i+d_i,.),CP_1(D_i+d_i+d_1,.)\otimes C_2(D_i+d_i+d_1,.)\otimes C'_3(D_i+d_i+d_1,.),t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

$$\dfrac{C_1(D_i+d_i,.),S_1(D_i+d_i,.),C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{C_1(D_i+d_i,.),S_1(D_i+d_i,.),C_1\multimap CP_1\otimes C_2\otimes C'_3,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\otimes_L$$

$$\dfrac{i(D_i,D_i+d_i)\vdash i\quad C_1(D_i+d_i,.)\otimes S_1(D_i+d_i,.),t_1,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}{i(D_i,.),i\multimap C_1\otimes S_1,t_1,t_{10},t_{11},t_{12},t_{13},t_{14},t_{16},t'_{18},t_2,t_{20},t_3,t_4,t_5,t_6,t_9,t_o,t_{15}\vdash o}\multimap_L$$

A.3 Representação das Datas Simbólicas em Tabelas

A Tabela 11 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sr2.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_1$
P_1	$D_i + d_1$	$D_i + d_1 + d_2 + d_4 + d_3$
P_2	$D_i + d_1$	$D_i + d_1 + d_2$
P_3	$D_i + d_1 + d_2$	$D_i + d_1 + d_2 + d_4$
P_4	$D_i + d_1 + d_2 + d_4$	$D_i + d_1 + d_2 + d_4 + d_3$
P_5	$D_i + d_1 + d_2 + d_4 + d_3$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5$
P_6	$D_i + d_1 + d_2 + d_4$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6$
P_7	$D_i + d_1 + d_2 + d_4 + d_3 + d_5$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6 + d_9 + d_{10}$
P_8	$D_i + d_1 + d_2 + d_4 + d_3 + d_5$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6$
P_9	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6 + d_9$
P_{11}	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6 + d_9$	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6 +$ $d_9 + d_{10}$
o	$D_i + d_1 + d_2 + d_4 + d_3 + d_5 + d_6 +$ $d_9 + d_{10}$	Desconhecido

Tabela 11 – Datas simbólicas de produção e de consumo para o cenário Sr2.

A Tabela 12 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sa1.

A Tabela 13 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sa2.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
C_1	$D_i + d_i$	$D_i + d_i + d_1$
S_1	$D_i + d_i$	$D_i + d_i + d_1 + d_2$
CP_1	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_2$
S_2	$D_i + d_i + d_1 + d_2$	$D_i + d_i + d_1 + d_2 + d_{11}$
C'_3	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_{18}$
C_2	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$
S_3	$D_i + d_i + d_1 + d_2 + d_{11}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12}$
C_4	$D_i + d_i + d_1 + d_{18}$	$D_i + d_i + d_1 + d_{18} + d_{20}$
S_4	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$
CP_2	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$
C_5	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}$
S_5	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_{17}$
S_9	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_{17}$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7) + d_8$
C_6	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7$
C_7	$D_i + d_i + d_1 + d_{18} + d_{20}$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7$
CP_4	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13},$ $d_{18} + d_{20}) + d_7$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7) + d_8$
C_{10}	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13},$ $d_{18} + d_{20}) + d_7$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7) + d_8 + d_o$
S_{10}	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) +$ $d_7) + d_8$	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20}) + d_7) + d_8 + d_o$
o	$D_i + d_i + d_1 + max(d_2 + d_{11} + d_{12} + d_4 + d_{17},$ $max(d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}, d_{18} + d_{20})$ $+d_7) + d_8 + d_o$	Desconhecido

Tabela 12 – Datas simbólicas de produção e de consumo para o cenário Sa1.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
C_1	$D_i + d_i$	$D_i + d_i + d_1$
S_1	$D_i + d_i$	$D_i + d_i + d_1 + d_2$
CP_1	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_2$
S_2	$D_i + d_i + d_1 + d_2$	$D_i + d_i + d_1 + d_2 + d_{11}$
C_2	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$
S_3	$D_i + d_i + d_1 + d_2 + d_{11}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12}$
C_3	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_{18}$
C_4	$D_i + d_i + d_1 + d_{18}$	$D_i + d_i + d_1 + d_{18} + d_{20}$
S_4	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$
CP_2	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$
C_5	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}$
S_5	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_{15}$
S_6	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_{15}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6$
C_6	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5$
CP_3	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6$
C_7	$D_i + d_i + d_1 + d_{18} + d_{20}$	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15})$ $+d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10}$
C_8	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}$
S_7	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 +$ $max(d_3 + d_{13} + d_5, d_{15}) + d_6$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15})$ $+d_6 + d_{16}$
C_9	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} +$ $d_5 + d_{14}$	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15})$ $+d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10}$
S_8	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 +$ $max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16}$	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15})$ $+d_6 + d_{16} + d_9$
CP_5	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 +$ $d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9$	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15})$ $+d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10}$

C_{10}	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10}$	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10} + d_o$
S_{10}	$D_i + d_i + d_1 + d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9$	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10} + d_o$
o	$D_i + d_i + d_1 + max(d_{18} + d_{20}, d_2 + d_{11} + d_{12} + d_4 + max(d_3 + d_{13} + d_5, d_{15}) + d_6 + d_{16} + d_9, d_2 + d_{11} + d_{12} + d_4 + d_3 + d_{13} + d_5 + d_{14}) + d_{10} + d_o$	Desconhecido

Tabela 13 – Datas simbólicas de produção e de consumo para o cenário Sa2.

A.4 Árvores de prova da Lógica Linear para o exemplo que ilustra o método para a identificação dos dos requisitos negativos

Árvore de prova para o cenário Snr1:

$$\underline{\dfrac{S_1 \vdash S_1 \quad C_1 \vdash C_1}{S_1, C_1 \vdash S_1 \otimes C_1}\otimes_R \qquad i \vdash i}\ \multimap_L$$

$$\underline{\dfrac{C'_3 \vdash C'_3 \quad C_2 \vdash C_2 \quad CP_1 \vdash CP_1}{C'_3, C_2, CP_1 \vdash C'_3 \otimes C_2 \otimes CP_1}\otimes_R \qquad S_1, C_1, C_1 \otimes S_1 \multimap i \vdash i}\ \multimap_L$$

$$\underline{C'_3, C_2, S_1, CP_1, C_3 \otimes C_2 \otimes CP_1 \multimap C_1, t_i \vdash i}\ \otimes_L$$

$$\underline{S_2 \vdash S_2 \qquad C'_3, C_2, S_1 \otimes CP_1, t_1, t_i \vdash i}\ \multimap_L$$

$$\underline{S_3 \vdash S_3 \qquad C'_3, C_2, S_2, S_2 \multimap S_1 \otimes CP_1, t_1, t_i \vdash i}\ \multimap_L$$

$$\underline{S_4 \vdash S_4 \qquad C'_3, C_2, S_3, S_3 \multimap S_2, t_1, t_2, t_i \vdash i}\ \multimap_L$$

$$\underline{\dfrac{CP_2 \vdash CP_2 \quad S_5 \vdash S_5}{CP_2, S_5 \vdash CP_2 \otimes S_5}\otimes_R \qquad C'_3, C_2, S_4, S_4 \multimap S_3, t_1, t_{11}, t_2, t_i \vdash i}\ \multimap_L$$

$$\underline{C'_3, S_5, C_2, CP_2, CP_2 \otimes S_5 \multimap S_4, t_1, t_{11}, t_{12}, t_2, t_i \vdash i}\ \otimes_L$$

$$\underline{C_5 \vdash C_5, C'_3, S_5, C_2 \otimes CP_2, t_1, t_{11}, t_{12}, t_2, t_4, t_i \vdash i}\ \multimap_L$$

$$\underline{S_9 \vdash S_9, C'_3, C_5, S_5, C_5 \multimap C_2 \otimes CP_2, t_1, t_{11}, t_{12}, t_2, t_4, t_i \vdash i}\ \multimap_L$$

$$\underline{C_6 \vdash C_6, S_9, C'_3, C_5, S_9 \multimap S_5, t_1, t_{11}, t_{12}, t_2, t_3, t_4, t_i \vdash i}\ \multimap_L$$

$$\underline{C_4 \vdash C_4, S_9, C_6, C'_3, C_6 \multimap C_5, t_1, t_{11}, t_{12}, t_{17}, t_2, t_3, t_4, t_i \vdash i}\ \multimap_L$$

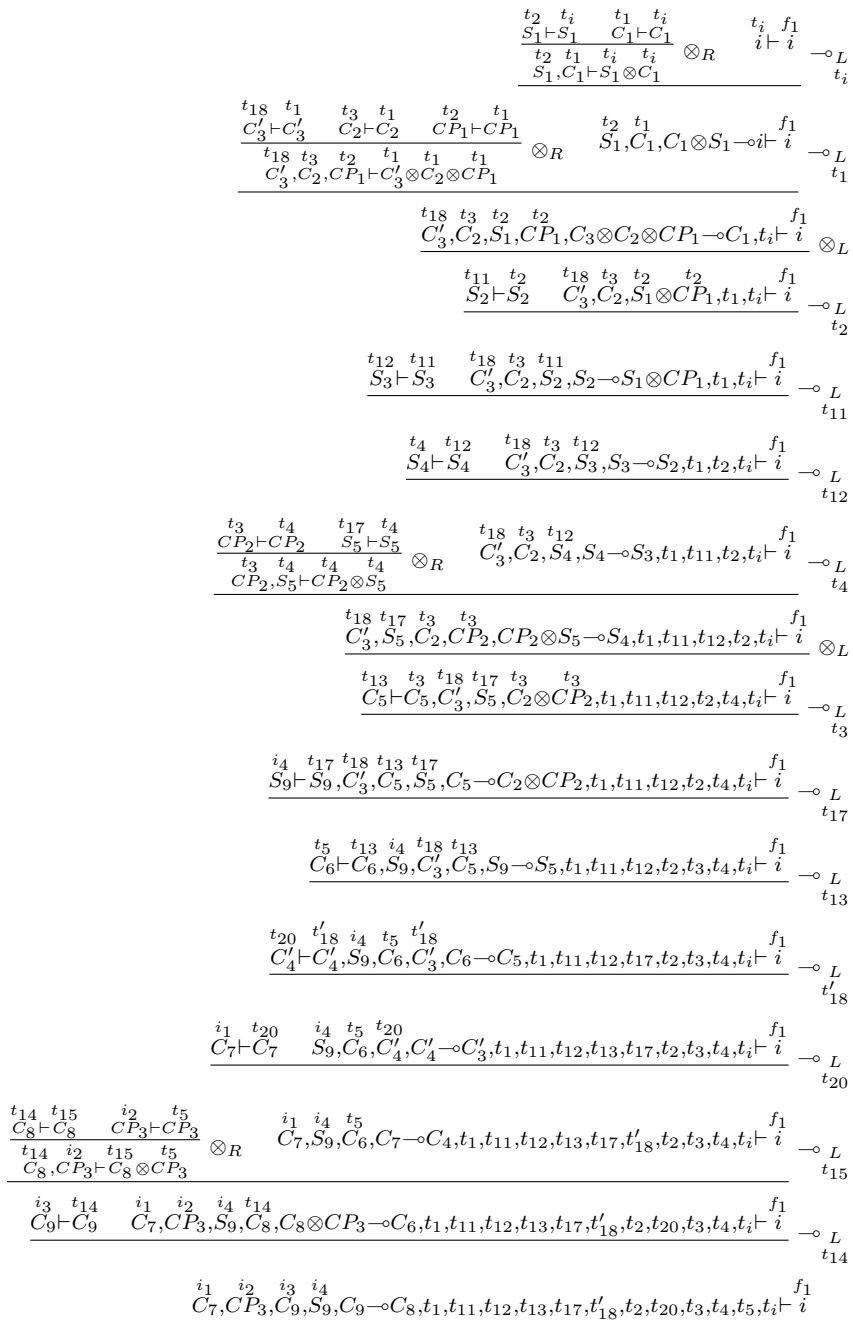
$$\underline{C_7 \vdash C_7 \qquad S_9, C_6, C_4, C'_4 \multimap C'_3, t_1, t_{11}, t_{12}, t_{13}, t_{17}, t_2, t_3, t_4, t_i \vdash i}\ \multimap_L$$

$$\underline{\dfrac{C_8 \vdash C_8 \quad CP_3 \vdash CP_3}{C_8, CP_3 \vdash C_8 \otimes CP3}\otimes_R \qquad C_7, S_9, C_6, C_7 \multimap C_4, t_1, t_{11}, t_{12}, t_{13}, t_{17}, t'_{18}, t_2, t_3, t_4, t_i \vdash i}\ \multimap_L$$

$$\underline{C_9 \vdash C_9 \qquad C_7, CP_3, S_9, C_8, C_8 \otimes CP_3 \multimap C_6, t_1, t_{11}, t_{12}, t_{13}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_i \vdash i}\ \multimap_L$$

$$C_7, CP_3, C_9, S_9, C_9 \multimap C_8, t_1, t_{11}, t_{12}, t_{13}, t_{17}, t'_{18}, t_2, t_{20}, t_3, t_4, t_5, t_i \vdash i$$

Árvore de prova rotulada com as transições de disparo para o cenário Snr1:



Apêndice B

Árvores de prova da Lógica Linear geradas para o estudo de caso

Neste apêndice são apresentadas as árvores de prova da Lógica Linear construídas para o exemplo que ilustra a aplicação dos métodos propostos no estudo de caso. O apêndice B.1 apresenta as árvores de prova da Lógica Linear para o exemplo que ilustra o método para verificação dos cenários de requisitos comportamentais em serviços Web compostos. O apêndice B.2 apresenta as árvores de prova da Lógica Linear para o exemplo que ilustra o método proposto para a verificação dos cenários de requisitos de desempenho em serviços Web compostos. As tabelas que contém as datas simbólicas de produção e de consumo sÃčo apresentadas no apêndice B.3. Já o apêndice B.4, apresenta as árvores de prova da Lógica Linear para a identificação dos dos requisitos negativos nos serviços Web compostos.

B.1 Árvores de prova da Lógica Linear para o exemplo que ilustra o método para a verificação dos cenários de requisitos comportamentais em serviços Web compostos

Árvore de prova para o cenário Sr3:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\frac{oC\vdash oC \quad oS\vdash oS}{oC,oS\vdash oC\otimes oS}\otimes_R \quad o\vdash o}\multimap_L}}{\frac{S_1\vdash S_1 \quad P_3\vdash P_3}{S_1,P_3\vdash P_3\otimes S_1}\otimes_R \quad oC,oS,oC\otimes oS\multimap o\vdash o}\multimap_L}}{S_1,P_3,oC,P_3\otimes S_1\multimap oS,t_o\vdash o}\otimes_L}}{C_1\vdash C_1 \quad S_1,P_3\otimes oC,P_3\otimes S_1\multimap oSt_o\vdash o}\multimap_L}}{\frac{iS\vdash iS \quad P_1\vdash P_1}{iS,P_1\vdash P_1\otimes iS}\otimes_R \quad C_1,S_1,C_1\multimap P_3\otimes oC,t_6,t_o\vdash o}\multimap_L}}{iS,P_1,C_1,iS\otimes P_1\multimap S_1,t_4,t_6,t_o\vdash o}\otimes_L}}{iC\vdash iC \quad iS,P_1\otimes C_1,t_2,t_4,t_6,t_o\vdash o}\multimap_L}}{iC,iS,iC\multimap P_1\otimes C_1,t_2,t_4,t_6,t_o\vdash o}\otimes_L}}{i\vdash i \quad iC\otimes iS,t_1,t_2,t_4,t_6,t_o\vdash o}\multimap_L}$$

$$i,i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_o\vdash o$$

Árvore de prova rotulada com as transições de disparo para o cenário Sr3:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\frac{\overset{t_4}{oC}\vdash\overset{t_o}{oC} \quad \overset{t_6}{oS}\vdash\overset{t_o}{oS}}{\overset{t_4\ t_6}{oC,oS}\vdash\overset{t_o\ t_o}{oC\otimes oS}}\otimes_R \quad \overset{t_o}{o}\vdash\overset{f_1}{o}}\multimap_{L\ t_o}}}{\frac{\overset{t_2}{S_1}\vdash\overset{t_6}{S_1} \quad \overset{t_4}{P_3}\vdash\overset{t_6}{P_3}}{\overset{t_2\ t_4}{S_1,P_3}\vdash\overset{t_6\ t_6}{P_3\otimes S_1}}\otimes_R \quad \overset{t_4\ t_6}{oC,oS,oC\otimes oS}\multimap o\vdash\overset{f_1}{o}}\multimap_{L\ t_6}}}{\overset{t_2\ t_4\ t_4}{S_1,P_3,oC,P_3\otimes S_1}\multimap oS,t_o\vdash\overset{f_1}{o}}\otimes_L}}{\overset{t_1}{C_1}\vdash\overset{t_4}{C_1} \quad \overset{t_2\ t_4\ t_4}{S_1,P_3\otimes oC,P_3\otimes S_1}\multimap oSt_o\vdash\overset{f_1}{o}}\multimap_{L\ t_4}}}{\frac{\overset{t_i}{iS}\vdash\overset{t_2}{iS} \quad \overset{t_1}{P_1}\vdash\overset{t_2}{P_1}}{\overset{t_i\ t_1}{iS,P_1}\vdash\overset{t_2\ t_2}{P_1\otimes iS}}\otimes_R \quad \overset{t_1\ t_2}{C_1,S_1,C_1}\multimap P_3\otimes oC,t_6,t_o\vdash\overset{f_1}{o}}\multimap_{L\ t_2}}}{\overset{t_i\ t_1\ t_1}{iS,P_1,C_1,iS\otimes P_1}\multimap S_1,t_4,t_6,t_o\vdash\overset{f_1}{o}}\otimes_L}}{\overset{t_i}{iC}\vdash\overset{t_1}{iC} \quad \overset{t_i\ t_1\ t_1}{iS,P_1\otimes C_1},t_2,t_4,t_6,t_o\vdash\overset{f_1}{o}}\multimap_{L\ t_1}}}{\overset{t_i\ t_i}{iC,iS,iC}\multimap P_1\otimes C_1,t_2,t_4,t_6,t_o\vdash\overset{f_1}{o}}\otimes_L}}{\overset{i_1\ t_i}{i}\vdash\overset{t_i}{i} \quad \overset{t_i\ t_i}{iC\otimes iS},t_1,t_2,t_4,t_6,t_o\vdash\overset{f_1}{o}}\multimap_{L\ t_i}}$$

$$\overset{i_1}{i},i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_o\vdash\overset{f_1}{o}$$

Árvore de prova para o cenário Sr4:

$$\dfrac{\dfrac{oC\vdash oC \quad oS\vdash oS}{oC,oS\vdash oC\otimes oS}\otimes_R \quad o\vdash o}{oC,oS,oC\otimes oS\multimap o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{C_2\vdash C_2 \quad P_4\vdash P_4}{C_2,P_4\vdash C_2\otimes P_4}\otimes_R \quad oC,oS,oC\otimes oS\multimap o\vdash o}{C_2,P_4,oS,C_2\otimes P_4\multimap oC,t_o\vdash o}\multimap_L$$

$$\dfrac{C_2,P_4,oS,C_2\otimes P_4\multimap oC,t_o\vdash o}{C_2,P_4\otimes oS,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{S_2\vdash S_2 \quad C_2,P_4\otimes oS,t_8,t_o\vdash o}{C_2,S_2,S_2\multimap P_4\otimes oS,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_1\vdash S_1 \quad P_2\vdash P_2}{S_1,P_2\vdash P_2\otimes S_1}\otimes_R \quad C_2,S_2,S_2\multimap P_4\otimes oS,t_8,t_o\vdash o}{S_1,P_2,C_2,S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{S_1,P_2,C_2,S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash o}{S_1,P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{C_1\vdash C_1 \quad S_1,P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}{C_1,S_1,C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{\dfrac{iS\vdash iS \quad P_1\vdash P_1}{iS,P_1\vdash P_1\otimes iS}\otimes_R \quad C_1,S_1,C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}{iS,P_1,C_1,iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{iS,P_1,C_1,iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash o}{iS,P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{iC\vdash iC \quad iS,P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{iC,iS,iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\dfrac{iC,iS,iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\dfrac{i\vdash i \quad iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{i,i\multimap iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

Árvore de prova rotulada com as transições de disparo para o cenário Sr4:

$$\dfrac{\dfrac{\overset{t_8}{oC}\vdash\overset{t_o}{oC} \quad \overset{t_7}{oS}\vdash\overset{t_o}{oS}}{\overset{t_8}{oC},\overset{t_7}{oS}\vdash\overset{t_o}{oC}\otimes\overset{t_o}{oS}}\otimes_R \quad \overset{t_o}{o}\vdash\overset{f_1}{o}}{oC,oS,oC\otimes oS\multimap o\vdash\overset{f_1}{o}}\multimap_{L\,t_o}$$

$$\dfrac{\dfrac{\overset{t_3}{C_2}\vdash\overset{t_8}{C_2} \quad \overset{t_7}{P_4}\vdash\overset{t_8}{P_4}}{\overset{t_3}{C_2},\overset{t_7}{P_4}\vdash\overset{t_8}{C_2}\otimes\overset{t_8}{P_4}}\otimes_R \quad oC,oS,oC\otimes oS\multimap o\vdash\overset{f_1}{o}}{\overset{t_3}{C_2},\overset{t_7}{P_4},\overset{t_7}{oS},C_2\otimes P_4\multimap oC,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_8}$$

$$\dfrac{\overset{t_3}{C_2},\overset{t_7}{P_4},\overset{t_7}{oS},C_2\otimes P_4\multimap oC,t_o\vdash\overset{f_1}{o}}{\overset{t_3}{C_2},\overset{t_7}{P_4}\otimes\overset{t_7}{oS},t_8,t_o\vdash\overset{f_1}{o}}\otimes_L$$

$$\dfrac{\overset{t_5}{S_2}\vdash\overset{t_7}{S_2} \quad \overset{t_3}{C_2},\overset{t_7}{P_4}\otimes\overset{t_7}{oS},t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_3}{C_2},\overset{t_5}{S_2},S_2\multimap P_4\otimes oS,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_7}$$

$$\dfrac{\dfrac{\overset{t_2}{S_1}\vdash\overset{t_5}{S_1} \quad \overset{t_3}{P_2}\vdash\overset{t_5}{P_2}}{\overset{t_2}{S_1},\overset{t_3}{P_2}\vdash\overset{t_5}{P_2}\otimes\overset{t_5}{S_1}}\otimes_R \quad \overset{t_3}{C_2},\overset{t_5}{S_2},S_2\multimap P_4\otimes oS,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_2}{S_1},\overset{t_3}{P_2},\overset{t_3}{C_2},S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_5}$$

$$\dfrac{\overset{t_2}{S_1},\overset{t_3}{P_2},\overset{t_3}{C_2},S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_2}{S_1},\overset{t_3}{P_2}\otimes\overset{t_3}{C_2},t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\otimes_L$$

$$\dfrac{\overset{t_1}{C_1}\vdash\overset{t_3}{C_1} \quad \overset{t_2}{S_1},\overset{t_3}{P_2}\otimes\overset{t_3}{C_2},t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_1}{C_1},\overset{t_2}{S_1},C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_3}$$

$$\dfrac{\dfrac{\overset{t_i}{iS}\vdash\overset{t_2}{iS} \quad \overset{t_1}{P_1}\vdash\overset{t_2}{P_1}}{\overset{t_i}{iS},\overset{t_1}{P_1}\vdash\overset{t_2}{P_1}\otimes\overset{t_2}{iS}}\otimes_R \quad \overset{t_1}{C_1},\overset{t_2}{S_1},C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_i}{iS},\overset{t_1}{P_1},\overset{t_1}{C_1},iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_2}$$

$$\dfrac{\overset{t_i}{iS},\overset{t_1}{P_1},\overset{t_1}{C_1},iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_i}{iS},\overset{t_1}{P_1}\otimes\overset{t_1}{C_1},t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\otimes_L$$

$$\dfrac{\overset{t_i}{iC}\vdash\overset{t_1}{iC} \quad \overset{t_i}{iS},\overset{t_1}{P_1}\otimes\overset{t_1}{C_1},t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_i}{iC},\overset{t_i}{iS},iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_1}$$

$$\dfrac{\overset{t_i}{iC},\overset{t_i}{iS},iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{t_i}{iC}\otimes\overset{t_i}{iS},t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\otimes_L$$

$$\dfrac{\overset{i_1}{i}\vdash\overset{t_i}{i} \quad \overset{t_i}{iC}\otimes\overset{t_i}{iS},t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}{\overset{i_1}{i},i\multimap iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash\overset{f_1}{o}}\multimap_{L\,t_i}$$

Árvore de prova para o cenário Sa5:

$$\frac{oC\vdash oC \quad oS\vdash oS}{oC,oS\vdash oC\otimes oS}\otimes_R \qquad o\vdash o \quad \multimap_L$$

$$S_7\vdash S_7 \qquad oC,oS,oC\otimes oS\multimap o\vdash o \quad \multimap_L$$

$$\frac{S_1\vdash S_1 \quad P_3\vdash P_3}{S_1,P_3\vdash S_1\otimes P_3}\otimes_R \qquad S_7,oC,S_7\multimap oS,t_o,\vdash o \quad \multimap_L$$

$$S_1,P_3,oC,P_3\otimes S_1\multimap S_7,t_{14},t_o\vdash o \quad \otimes_L$$

$$C_1\vdash C_1 \qquad S_1,P_3\otimes oC,t_6,t_{14},t_o\vdash o \quad \multimap_L$$

$$\frac{iS\vdash iS \quad P_1\vdash P_1}{iS,P_1\vdash iS\otimes P_1}\otimes_R \qquad C_1,S_1,C_1\multimap P_3\otimes oC,t_6,t_{14},t_o\vdash o \quad \multimap_L$$

$$iS,C_1,P_1,P_1\otimes iS\multimap S_1,t_4,t_6,t_{14},t_o\vdash o \quad \otimes_L$$

$$C_3\vdash C_3 \qquad iS,C_1\otimes P_1,t_2,t_4,t_6,t_{14},t_o\vdash o \quad \multimap_L$$

$$iC\vdash iC \qquad iS,C_3,C_3\multimap C_1\otimes P_1,t_2,t_4,t_6,t_{14},t_o\vdash o \quad \multimap_L$$

$$iC,iS,iC\multimap C_3,t_1,t_2,t_4,t_6,t_{14},t_o\vdash o \quad \otimes_L$$

$$i\vdash i \qquad iC\otimes iS,t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash o \quad \multimap_L$$

$$i,i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash o$$

Árvore de prova rotulada com as transições de disparo para o cenário Sa5:

$$\frac{\overset{t_4}{oC}\vdash\overset{t_o}{oC} \quad \overset{t_6}{oS}\vdash\overset{t_o}{oS}}{\overset{t_4}{oC},\overset{t_6}{oS}\vdash\overset{t_o}{oC}\otimes\overset{t_o}{oS}}\otimes_R \qquad \overset{t_o}{o}\vdash\overset{f_1}{o} \quad \underset{t_o}{\multimap_L}$$

$$\overset{t_6}{S_7}\vdash\overset{t_{14}}{S_7} \qquad \overset{t_4}{oC},\overset{t_{14}}{oS},oC\otimes oS\multimap o\vdash\overset{f_1}{o} \quad \multimap_L$$

$$\frac{\overset{t_2}{S_1}\vdash\overset{t_6}{S_1} \quad \overset{t_4}{P_3}\vdash\overset{t_6}{P_3}}{\overset{t_2}{S_1},\overset{t_4}{P_3}\vdash\overset{t_6}{S_1}\otimes\overset{t_6}{P_3}}\otimes_R \qquad \overset{t_4}{oC},S_7\multimap oS,t_o\vdash\overset{f_1}{o} \quad \underset{t_6}{\multimap_L}$$

$$\overset{t_2}{S_1},\overset{t_4}{P_3},\overset{t_4}{oC},P_3\otimes S_1\multimap S_7,t_{14},t_o\vdash\overset{f_1}{o} \quad \otimes_L$$

$$\overset{t_1}{C_1}\vdash\overset{t_4}{C_1} \qquad \overset{t_2}{S_1},\overset{t_4}{P_3}\otimes\overset{t_4}{oC},t_6,t_{14},t_o\vdash\overset{f_1}{o} \quad \underset{t_4}{\multimap_L}$$

$$\frac{\overset{t_i}{iS}\vdash\overset{t_2}{iS} \quad \overset{t_1}{P_1}\vdash\overset{t_2}{P_1}}{\overset{t_i}{iS},\overset{t_1}{P_1}\vdash\overset{t_2}{iS}\otimes\overset{t_2}{P_1}}\otimes_R \qquad \overset{t_1}{C_1},\overset{t_2}{S_1},C_1\multimap P_3\otimes oC,t_6,t_{14},t_o\vdash\overset{f_1}{o} \quad \underset{t_2}{\multimap_L}$$

$$\overset{t_i}{iS},\overset{t_1}{C_1},\overset{t_1}{P_1},P_1\otimes iS\multimap S_1,t_4,t_6,t_{14},t_o\vdash\overset{f_1}{o} \quad \otimes_L$$

$$\overset{t_9}{C_3}\vdash\overset{t_1}{C_3} \qquad \overset{t_i}{iS},\overset{t_1}{C_1}\otimes\overset{t_1}{P_1},t_2,t_4,t_6,t_{14},t_o\vdash\overset{f_1}{o} \quad \underset{t_1}{\multimap_L}$$

$$\overset{t_i}{iC}\vdash\overset{t_9}{iC} \qquad \overset{t_i}{iS},\overset{t_9}{C_3},C_3\multimap C_1\otimes P_1,t_2,t_4,t_6,t_o\vdash\overset{f_1}{o} \quad \underset{t_9}{\multimap_L}$$

$$\overset{t_i}{iC},\overset{t_i}{iS},iC\multimap C_3,t_1,t_2,t_4,t_6,t_{14},t_o\vdash\overset{f_1}{o} \quad \otimes_L$$

$$\overset{i_1}{i}\vdash\overset{t_i}{i} \qquad \overset{t_i}{iC}\otimes\overset{t_i}{iS},t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash\overset{f_1}{o} \quad \underset{t_i}{\multimap_L}$$

$$\overset{i_1}{i},i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash\overset{f_1}{o}$$

Árvore de prova para o cenário Sa6:

$$\dfrac{\dfrac{oC \vdash oC \quad oS \vdash oS}{oC,oS \vdash oC \otimes oS}\otimes_R \quad o \vdash o}{oS,oC,oC \otimes oS \multimap o \vdash o}\multimap_L$$

$$\dfrac{\dfrac{P_4 \vdash P_4 \quad C_2 \vdash C_2}{P_4,C_2 \vdash C_2 \otimes P_4}\otimes_R \quad oS,oC,oC \otimes oS \multimap o \vdash o}{P_4,oS,C_2,C_2 \otimes P_4 \multimap oC,t_o \vdash o}\multimap_L$$

$$\dfrac{C_4 \vdash C_4 \quad P_4,oS,C_2,C_2 \otimes P_4 \multimap oC,t_o \vdash o}{C_4,P_4,oS,C_4 \multimap C_2,t_8,t_o \vdash o}\multimap_L$$

$$\dfrac{C_4,P_4,oS,C_4 \multimap C_2,t_8,t_o \vdash o}{S_2 \vdash S_2 \quad C_4,P_4 \otimes oS,t_8,t_{12},t_o \vdash o}\multimap_L$$

$$\dfrac{S_2 \vdash S_2 \quad C_4,P_4 \otimes oS,t_8,t_{12},t_o \vdash o}{S_4 \vdash S_4 \quad C_4,S_5,S_6,S_5 \otimes S_6 \multimap S_2,t_7,t_8,t_{12},t_o \vdash o}\multimap_L$$

$$\dfrac{\dfrac{S_5 \vdash S_5 \quad S_6 \vdash S_6}{S_5,S_6 \vdash S_5 \otimes S_6}\otimes_R \quad C_4,S_2,S_2 \multimap P_4 \otimes oS,S_5,t_8,t_{12},t_o \vdash o}{}\multimap_L$$

$$\dfrac{S_4 \vdash S_4 \quad C_4,S_5,S_6,S_5 \otimes S_6 \multimap S_2,t_7,t_8,t_{12},t_o \vdash o}{\dfrac{P_2 \vdash P_2 \quad S_3 \vdash S_3}{P_2,S_3 \vdash S_3 \otimes P_2}\otimes_R \quad C_4,S_4,S_5,S_4 \multimap S_6,S_5,t_7,t_8,t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{\dfrac{P_2 \vdash P_2 \quad S_3 \vdash S_3}{P_2,S_3 \vdash S_3 \otimes P_2}\otimes_R \quad C_4,S_4,S_5,S_4 \multimap S_6,S_5,t_7,t_8,t_{12},t_{13},t_o \vdash o}{C_4,P_2,S_3,S_4,S_3 \otimes P_2 \multimap S_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{C_4,P_2,S_3,S_4,S_3 \otimes P_2 \multimap S_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}{S_1 \vdash S_1 \quad C_4,P_2,S_3 \otimes S_4,t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{S_1 \vdash S_1 \quad C_4,P_2,S_3 \otimes S_4,t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}{S_1,C_4,P_2,S_1 \multimap S_3 \otimes S_4,t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{S_1,C_4,P_2,S_1 \multimap S_3 \otimes S_4,t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o \vdash o}{C_1 \vdash C_1 \quad S_1,C_4 \otimes P_2,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\otimes_L$$

$$\dfrac{C_1 \vdash C_1 \quad S_1,C_4 \otimes P_2,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{\dfrac{iS \vdash iS \quad P_1 \vdash P_1}{iS,P_1 \vdash iS \otimes P_1}\otimes_R \quad C_1,S_1,C_1 \multimap C_4 \otimes P_2,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{\dfrac{iS \vdash iS \quad P_1 \vdash P_1}{iS,P_1 \vdash iS \otimes P_1}\otimes_R \quad C_1,S_1,C_1 \multimap C_4 \otimes P_2,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{iS,C_1,P_1,P_1 \otimes iS \multimap S_1,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{iS,C_1,P_1,P_1 \otimes iS \multimap S_1,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{C_3 \vdash C_3 \quad iS,C_1 \otimes P_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\otimes_L$$

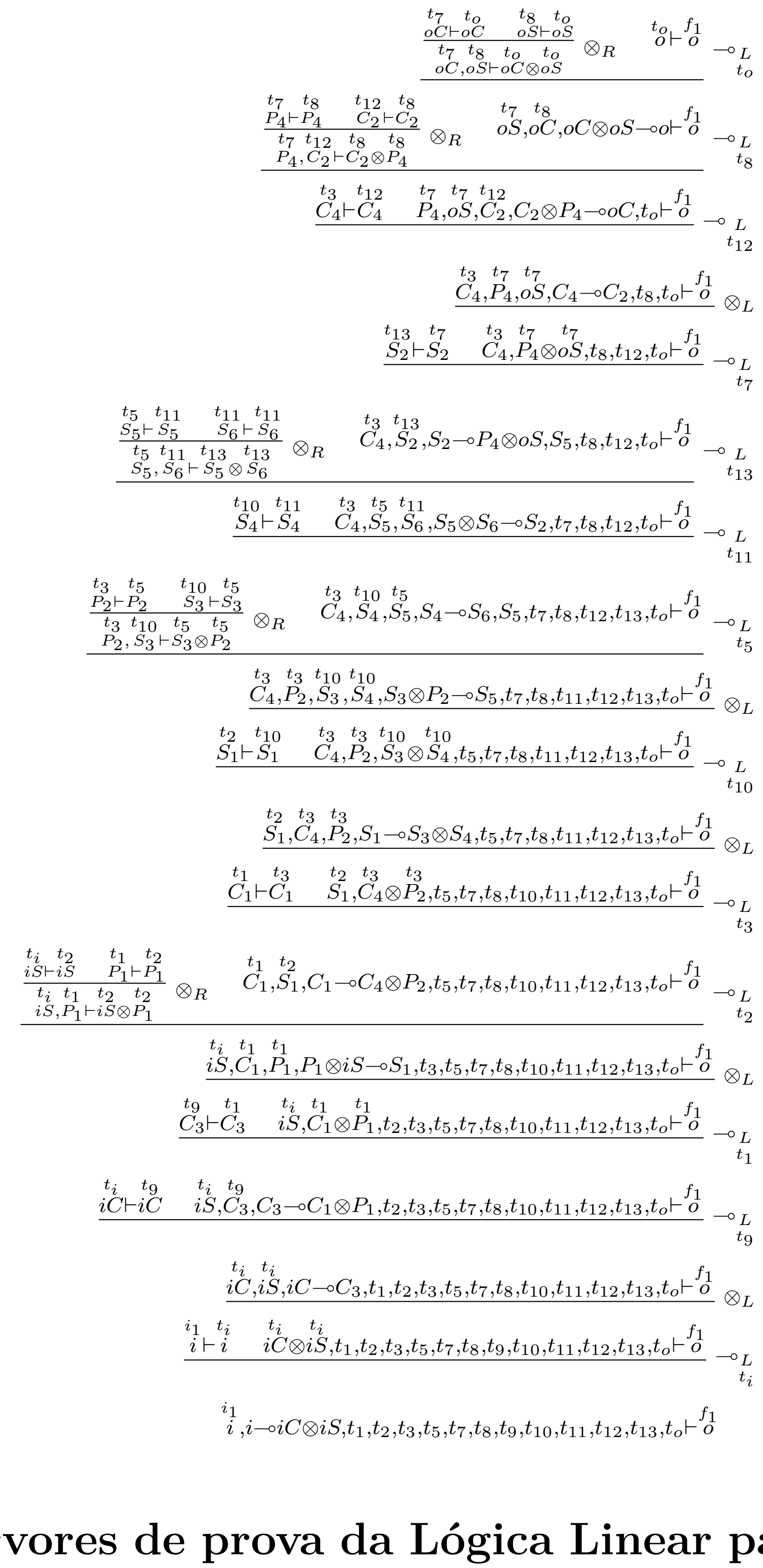
$$\dfrac{C_3 \vdash C_3 \quad iS,C_1 \otimes P_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{iC \vdash iC \quad iS,C_3,C_3 \multimap C_1 \otimes P_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{iC \vdash iC \quad iS,C_3,C_3 \multimap C_1 \otimes P_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{iC,iS,iC \multimap C_3,t_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

$$\dfrac{iC,iS,iC \multimap C_3,t_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{i \vdash i \quad iC \otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\otimes_L$$

$$\dfrac{i \vdash i \quad iC \otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}{i,i \multimap iC \otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_o \vdash o}\multimap_L$$

Árvore de prova rotulada com as transições de disparo para o cenário Sa6:



B.2 Árvores de prova da Lógica Linear para o exemplo que ilustra o método para a verificação dos cenários de requisitos de desempenho em serviços Web compostos

Por razões de espaço, algumas partes das datas simbólicas serão representadas por sequências. Considerando $Seq_9 = D_i + d_i + d_1$ e $Seq_{10} = max(d_2, d_4) + d_6$, a árvore de prova com datas simbólicas para o cenário Sr3 é:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{oC(Seq11+d_4,Seq9+max(d_4,d_6)+d_o)\vdash oC}\quad \dfrac{}{oS(Seq9+Seq_{10},Seq9+Seq_{10}+d_o)\vdash oS}\otimes_R}{oCoC(Seq11+d_4,Seq9+max(d_4,d_6)+d_o),oS(Seq9+Seq_{10},Seq9+Seq_{10}+d_o)\vdash oC\otimes oS}\otimes_R \quad \dfrac{}{o(Seq9+Seq12+d_o,.)\vdash o}\multimap_L}{S_1(Seq9+d_2,Seq9+Seq_{10})\vdash S_1 \quad P_3(Seq9+d_4,Seq9+Seq_{10})\vdash P_3}\otimes_R}{S_1(Seq9+d_2,Seq9+Seq_{10}),P_3(Seq9+d_4,Seq9+Seq_{10})\vdash P_3\otimes S_1}\otimes_R \quad \dfrac{}{oC(Seq9+d_4,.),oS(Seq9+Seq_{10},.),oC\otimes oS\multimap o\vdash o}\multimap_L}{S_1(Seq9+d_2,.),P_3(Seq9+d_4,.),oC(Seq9+d_4,.),P_3\otimes S_1\multimap oS,t_o\vdash o}\otimes_L}{C_1(Seq9,Seq9+d_4)\vdash C_1 \quad S_1(Seq9+d_2,.),P_3(Seq9+d_4,.)\otimes oC(Seq9+d_4,.),P_3\otimes S_1\multimap oSt_o\vdash o}\multimap_L}{\dfrac{\dfrac{}{iS(D_i+d_i,Seq9+d_2)\vdash iS}\quad \dfrac{}{P_1(Seq9,Seq9+d_2)\vdash P_1}\otimes_R}{iS(D_i+d_i,Seq9+d_2),P_1(Seq9,Seq9+d_2)\vdash P_1\otimes iS}\otimes_R \quad \dfrac{}{C_1(Seq9,.),S_1(Seq9+d_2,.),C_1\multimap P_3\otimes oC,t_6,t_o\vdash o}\multimap_L}{iS(D_i+d_i,.),P_1(Seq9,.),C_1(Seq9,.),iS\otimes P_1\multimap S_1,t_4,t_6,t_o\vdash o}\otimes_L}{iC(D_i+d_i,Seq9)\vdash iC \quad iS(D_i+d_i,.),P_1(Seq9,.)\otimes C_1(Seq9,.),t_2,t_4,t_6,t_o\vdash o}\multimap_L}{iC(D_i+d_i,.),iS(D_i+d_i,.),iC\multimap P_1\otimes C_1,t_2,t_4,t_6,t_o\vdash o}\otimes_L}{i(D_i,D_i+d_i)\vdash i \quad iC(D_i+d_i,.)\otimes iS(D_i+d_i,.),t_1,t_2,t_4,t_6,t_o\vdash o}\multimap_L$$

$$i(D_i,.),i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_o\vdash o$$

Condiserando $Seq_{11} = max(d_2, d_3) + d_5$, a árvore de prova com datas simbólicas para o cenário Sr4 é:

$$\cfrac{\cfrac{oC(Seq_9{+}Seq_{11}{+}d_7{+}d_8,Seq_9{+}Seq_{11}{+}d_7{+}d_8{+}d_o)\vdash oC \qquad oS(Seq_9{+}Seq_{11}{+}d_7,Seq_9{+}Seq_{11}{+}d_7{+}d_8{+}d_o)\vdash oS}{oC(Seq_9{+}Seq_{11}{+}d_7{+}d_8,Seq_9{+}Seq_{11}{+}d_7{+}d_8{+}d_o),oS(Seq_9{+}Seq_{11}{+}d_7,Seq_9{+}Seq_{11}{+}d_7{+}d_8{+}d_o)\vdash oC\otimes oS}\otimes_R \qquad o(Seq_9{+}Seq_{11}{+}d_7{+}d_8{+}d_o,.))\vdash o}{oC(Seq_9{+}Seq_{11}{+}d_7{+}d_8,.),oS(Seq_9{+}Seq_{11}{+}d_7,.),oC\otimes oS\multimap o\vdash o}\multimap_L$$

$$\cfrac{\cfrac{C_2(Seq_9{+}d_3,Seq_9{+}Seq_{11}{+}d_7{+}d_8)\vdash C_2 \qquad P_4(Seq_9{+}Seq_{11}{+}d_7,Seq_9{+}Seq_{11}{+}d_7{+}d_8)\vdash P_4}{C_2(Seq_9{+}d_3,Seq_9{+}Seq_{11}{+}d_7{+}d_8),P_4(Seq_9{+}Seq_{11}{+}d_7,Seq_9{+}Seq_{11}{+}d_7{+}d_8)\vdash C_2\otimes P_4}\otimes_R \qquad oC(Seq_9{+}Seq_{11}{+}d_7{+}d_8,.),oS(Seq_9{+}Seq_{11}{+}d_7,.),oC\otimes oS\multimap o\vdash o}{C_2(Seq_9{+}d_3,.),P_4(Seq_9{+}Seq_{11}{+}d_7,.),oS(Seq_9{+}Seq_{11}{+}d_7,.),C_2\otimes P_4\multimap oC,t_o\vdash o}\multimap_L$$

$$\cfrac{C_2(Seq_9{+}d_3,.),P_4(Seq_9{+}Seq_{11}{+}d_7,.),oS(Seq_9{+}Seq_{11}{+}d_7,.),C_2\otimes P_4\multimap oC,t_o\vdash o}{C_2(Seq_9{+}d_3,.),P_4(Seq_9{+}Seq_{11}{+}d_7,.)\otimes oS(Seq_9{+}Seq_{11}{+}d_7),t_8,t_o\vdash o}\otimes_L$$

$$\cfrac{S_2(Seq_9{+}Seq_{11},Seq_9{+}Seq_{11}{+}d_7)\vdash S_2 \qquad C_2(Seq_9{+}d_3,.),P_4(Seq_9{+}Seq_{11}{+}d_7,.)\otimes oS(Seq_9{+}Seq_{11}{+}d_7),t_8,t_o\vdash o}{C_2(Seq_9{+}d_3,.),S_2(Seq_9{+}Seq_{11},.),S_2\multimap P_4\otimes oS,t_8,t_o\vdash o}\multimap_L$$

$$\cfrac{\cfrac{S_1(Seq_9{+}d_2,Seq_9{+}Seq_{11})\vdash S_1 \qquad P_2(Seq_9{+}d_3,Seq_9{+}Seq_{11})\vdash P_2}{S_1(Seq_9{+}d_2,Seq_9{+}Seq_{11}),P_2(Seq_9{+}d_3,Seq_9{+}Seq_{11})\vdash P_2\otimes S_1}\otimes_R \qquad C_2(Seq_9{+}d_3,.),S_2(Seq_9{+}Seq_{11},.),S_2\multimap P_4\otimes oS,t_8,t_o\vdash o}{S_1(Seq_9{+}d_2,.),P_2(Seq_9{+}d_3,.),C_2(Seq_9{+}d_3,.),S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\cfrac{S_1(Seq_9{+}d_2,.),P_2(Seq_9{+}d_3,.),C_2(Seq_9{+}d_3,.),S_1\otimes P_2\multimap S_2,t_7,t_8,t_o\vdash o}{S_1(Seq_9{+}d_2,.),P_2(Seq_9{+}d_3,.)\otimes C_2(Seq_9{+}d_3,.),t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\cfrac{C_1(Seq_9,Seq_9{+}d_3))\vdash C_1 \qquad S_1(Seq_9{+}d_2,.),P_2(Seq_9{+}d_3,.)\otimes C_2(Seq_9{+}d_3,.),t_5,t_7,t_8,t_o\vdash o}{C_1(Seq_9,.),S_1(Seq_9{+}d_2,.),C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\cfrac{\cfrac{iS(D_i{+}d_i,Seq_9{+}d_2)\vdash iS \qquad P_1(Seq_9,Seq_9{+}d_2)\vdash P_1}{iS(D_i{+}d_i,Seq_9{+}d_2),P_1(Seq_9,Seq_9{+}d_2)\vdash P_1\otimes iS}\otimes_R \qquad C_1(Seq_9,.),S_1(Seq_9{+}d_2,.),C_1\multimap P_2\otimes C_2,t_5,t_7,t_8,t_o\vdash o}{iS(D_i{+}d_i,.),P_1(Seq_9,.),C_1(Seq_9,.),iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\cfrac{iS(D_i{+}d_i,.),P_1(Seq_9,.),C_1(Seq_9,.),iS\otimes P_1\multimap S_1,t_3,t_5,t_7,t_8,t_o\vdash o}{iS(D_i{+}d_i,.),P_1(Seq_9,.)\otimes C_1(Seq_9,.),t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\cfrac{iC(D_i{+}d_i,Seq_9)\vdash iC \qquad iS(D_i{+}d_i,.),P_1(Seq_9,.)\otimes C_1(Seq_9,.),t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{iC(D_i{+}d_i,.),iS(D_i{+}d_i,.),iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

$$\cfrac{iC(D_i{+}d_i,.),iS(D_i{+}d_i,.),iC\multimap P_1\otimes C_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{iC(D_i{+}d_i,.)\otimes iS(D_i{+}d_i,.),t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\otimes_L$$

$$\cfrac{i(D_i,D_i{+}d_i)\vdash i \qquad iC(D_i{+}d_i,.)\otimes iS(D_i{+}d_i,.),t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}{i(D_i,.),i\multimap iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_o\vdash o}\multimap_L$$

Condiserando $Seq_{12} = D_i + d_i + d_9 + d_1$, a árvore de prova com datas simbólicas para o cenário Sa5 é:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{oC(Seq_{12}+d_4,Seq_{12}+Seq_{10}+d_{14}+d_o)\vdash oC \quad oS(Seq_{12}+Seq_{10}+d_{14},Seq_{12}+Seq_{10}+d_{14}+d_o)\vdash oS}{oC(Seq_{12}+d_4,Seq_{12}+Seq_{10}+d_{14}+d_o),oS(Seq_{12}+Seq_{10}+d_{14},Seq_{12}+Seq_{10}+d_{14}+d_o)\vdash oC\otimes oS}\otimes_R \quad o(Seq_{12}+Seq_{10}+d_{14}+d_o,.)\vdash o}{S_7(Seq_{12}+Seq_{10},Seq_{12}+Seq_{10}+d_{14})\vdash S_7 \quad oC(Seq_{12}+d_4,.),oS(Seq_{12}+Seq_{10}+d_{14},.),oC\otimes oS\multimap o\vdash o}\multimap_L}{\dfrac{S_1(Seq_{12}+d_2,Seq_{12}+Seq_{10})\vdash S_1 \quad P_3(Seq_{12}+d_4,Seq_{12}+Seq_{10})\vdash P_3}{S_1(Seq_{12}+d_2,Seq_{12}+Seq_{10}),P_3(Seq_{12}+d_4,Seq_{12}+Seq_{10})\vdash S_1\otimes P_3}\otimes_R \quad oC(Seq_{12}+d_4,.),S_7(Seq_{12}+Seq_{10},.),S_7\multimap oS,t_o\vdash o}\multimap_L}{S_1(Seq_{12}+d_2,.),P_3(Seq_{12}+d_4,.),oC(Seq_{12}+d_4,.),P_3\otimes S_1\multimap S7,t_{14},t_o\vdash o}\otimes_L}{C_1(Seq_{12},Seq_{12}+d_4)\vdash C_1 \quad S_1(Seq_{12}+d_2,.),P_3(Seq_{12}+d_4,.)\otimes oC(Seq_{12}+d_4,.),t_6,t_{14},t_o\vdash o}\multimap_L}{\dfrac{iS(D_i+d_i,Seq_{12}+d_2)\vdash iS \quad P_1(Seq_{12},Seq_{12}+d_2)\vdash P_1}{iS(D_i+d_i,Seq_{12}+d_2),P_1(Seq_{12},Seq_{12}+d_2)\vdash iS\otimes P_1}\otimes_R \quad C_1(Seq_{12},.),S_1(Seq_{12}+d_2,.),C_1\multimap P_3\otimes oC,t_6,t_{14},t_o\vdash o}\multimap_L}{iS(D_i+d_i,.),C_1(Seq_{12},.),P_1(Seq_{12},.),P_1\otimes iS\multimap S_1,t_4,t_6,t_{14},t_o\vdash o}\otimes_L}{C_3(D_i+d_i+d_9,Seq_{12})\vdash C_3 \quad iS(D_i+d_i,.),C_1(Seq_{12},.)\otimes P_1(Seq_{12},.),t_2,t_4,t_6,t_{14},t_o\vdash o}\multimap_L}{iC(D_i+d_i,D_i+d_i+d_9)\vdash iC \quad iS(D_i+d_i,.),C_3(D_i+d_i+d_9,.),C_3\multimap C_1\otimes P_1,t_2,t_4,t_6,t_{14},t_o\vdash o}\multimap_L}{iC(D_i+d_i,.),iS(D_i+d_i,.),iC\multimap C_3,t_1,t_2,t_4,t_6,t_{14},t_o\vdash o}\otimes_L}{i(D_i,D_i+d_i)\vdash i \quad iC(D_i+d_i,.)\otimes iS(D_i+d_i,.),t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash o}\multimap_L$$

$$i(D_i,.),i\multimap iC\otimes iS,t_1,t_2,t_4,t_6,t_{14},t_9,t_o\vdash o$$

Condiserando $Seq_{13} = D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) + d_{13}$, $Seq_{14} = D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_{12}, max(d_5, d_{11}) + d_3 + d_7) + d_8$ e $Seq_{15} = D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + d_5$, a árvore de prova com datas simbólicas para o cenário Sa6 é:

$$\dfrac{\dfrac{oC(Seq_{14},Seq_{14}+d_o)\vdash oC \quad oS(Seq_{13}+d_7,Seq_{14}+d_o)\vdash oS}{oC(Seq_{14},Seq_{14}+d_o),oS(Seq_{13}+d_7,Seq_{14}+d_o)\vdash oC\otimes oS}\otimes_R \quad o(Seq_{14}+d_o,.)\vdash o}{}\multimap_L$$

$$\dfrac{\dfrac{P_4(Seq_{13}+d_7,Seq_{14})\vdash P_4 \quad C_2(Seq_{12}+d_3+d_{12},Seq_{14})\vdash C_2}{P_4(Seq_{13}+d_7,Seq_{14}),C_2(Seq_{12}+d_3+d_{12},Seq_{14})\vdash C_2\otimes P_4}\otimes_R \quad oS(Seq_{13}+d_7,.),oC(Seq_{14},.),oC\otimes oS\multimap o\vdash o}{}\multimap_L$$

$$\dfrac{C_4(Seq_{12}+d_3,Seq_{12}+d_3+d_{12})\vdash C_4 \quad P_4(Seq_{13}+d_7,.),oS(Seq_{13}+d_7,.),C_2(Seq_{12}+d_3+d_{12},.),C_2\otimes P_4\multimap oC,t_o\vdash o}{}\multimap_L$$

$$\dfrac{C_4(Seq_{12}+d_3,.),P_4(Seq_{13}+d_7,.),oS(Seq_{13}+d_7,.),C_4\multimap C_2,t_8,t_o\vdash o}{}\multimap_L$$

$$\dfrac{S_2(Seq_{13},Seq_{13}+d_7)\vdash S_2 \quad C_4(Seq_{12}+d_3,.),P_4(Seq_{13}+d_7,.)\otimes oS(Seq_{13}+d_7,.),t_8,t_{12},t_o\vdash o}{}\multimap_L$$

$$\dfrac{\dfrac{S_5(Seq_{15},Seq_{13})\vdash S_5 \quad S_6(Seq_{12}+d_2+d_{10}+d_{11},Seq_{13})\vdash S_6}{S_5(Seq_{15},Seq_{13}),S_6(Seq_{12}+d_2+d_{10}+d_{11},Seq_{13})\vdash S_5\otimes S_6}\otimes_R \quad C_4(Seq_{12}+d_3,.),S_2(Seq_{13},.),S_2\multimap P_4\otimes oS,S_5,t_8,t_{12},t_o\vdash o}{}\multimap_L$$

$$\dfrac{S_4(Seq_{12}+d_2+d_{10},Seq_{12}+d_2+d_{10}+d_{11})\vdash S_4 \quad C_4(Seq_{12}+d_3,Seq_{12}+d_3+d_{12}),S_5(Seq_{15},.),S_6(Seq_{12}+d_2+d_{10}+d_{11},.),S_5\otimes S_6\multimap S_2,t_7,t_8,t_{12},t_o\vdash o}{}\multimap_L$$

$$\dfrac{\dfrac{P_2(Seq_{12}+d_3,Seq_{15})\vdash P_2 \quad S_3(Seq_{12}+d_2+d_{10},Seq_{15})\vdash S_3}{P_2(Seq_{12}+d_3,Seq_{15}),S_3(Seq_{12}+d_2+d_{10},Seq_{15})\vdash S_3\otimes P_2}\otimes_R \quad C_4(Seq_{12}+d_3,.),S_4(Seq_{12}+d_2+d_{10},.),S_5(Seq_{15},.),S_4\multimap S_6,t_7,t_8,t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{C_4(Seq_{12}+d_3,.),P_2(Seq_{12}+d_3,.),S_3(Seq_{12}+d_2+d_{10},.),S_4(Seq_{12}+d_2+d_{10},.),S_3\otimes P_2\multimap S_5,t_7,t_8,t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{S_1(Seq_{12}+d_2,Seq_{12}+d_2+d_{10})\vdash S_1 \quad C_4(Seq_{12}+d_3,.),P_2(Seq_{12}+d_3,.),S_3(Seq_{12}+d_2+d_{10},.)\otimes S_4(Seq_{12}+d_2+d_{10},.),t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{S_1(Seq_{12}+d_2,.),C_4(Seq_{12}+d_3,.),P_2(Seq_{12}+d_3,.),S_1\multimap S_3\otimes S_4,t_5,t_7,t_8,t_{11},t_{12},t_{13},t_o\vdash o}{}\otimes_L$$

$$\dfrac{C_1(Seq_{12},Seq_{12}+d_3)\vdash C_1 \quad S_1(Seq_{12}+d_2,.),C_4(Seq_{12}+d_3,.)\otimes P_2(Seq_{12}+d_3,.),t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{\dfrac{iS(D_i+d_i,Seq_{12}+d_2)\vdash iS \quad P_1(Seq_{12},Seq_{12}+d_2)\vdash P_1}{iS(D_i+d_i,Seq_{12}+d_2),P_1(Seq_{12},Seq_{12}+d_2)\vdash iS\otimes P_1}\otimes_R \quad C_1(Seq_{12},.),S_1(Seq_{12}+d_2,.),C_1\multimap C_4\otimes P_2,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{iS(D_i+d_i,.),C_1(Seq_{12},.),P_1(Seq_{12},.),P_1\otimes iS\multimap S_1,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\otimes_L$$

$$\dfrac{C_3(D_i+d_i+d_9,Seq_{12})\vdash C_3 \quad iS(D_i+d_i,.),C_1(Seq_{12},.)\otimes P_1(Seq_{12},.),t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{iC(D_i+d_i,D_i+d_i+d_9)\vdash iC \quad iS(D_i+d_i,.),C_3(D_i+d_i+d_9,.),C_3\multimap C_1\otimes P_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$\dfrac{iC(D_i+d_i,.),iS(D_i+d_i,.),iC\multimap C_3,t_1,t_2,t_3,t_5,t_7,t_8,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\otimes_L$$

$$\dfrac{i(D_i,D_i+d_i)\vdash i \quad iC(D_i+d_i,.)\otimes iS(D_i+d_i,.),t_1,t_2,t_3,t_5,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_o\vdash o}{}\multimap_L$$

$$i(D_i,.),i\multimap iC\otimes iS,t_1,t_2,t_3,t_5,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_o\vdash o$$

B.3 Representação das Datas Simbólicas em Tabelas

A Tabela 14 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sr3.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
iC	$D_i + d_i$	$D_i + d_i + d_1$
P_1	$D_i + d_i + d_1$	$D_i + max(d_i, d_1) + d_2$
iS	$D_i + d_i$	$D_i + d_i + d_1 + d_2$
C_1	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_4$
S_1	$D_i + d_i + d_1 + d_2$	$D_i + d_i + d_1 + max(d_2, d_4) + d_6$
oC	$D_i + d_i + d_1 + d_4$	$D_i + d_i + d_1 + max(d_4, d_6) + d_o$
P_3	$D_i + d_i + d_1 + d_4$	$D_i + d_i + d_1 + max(d_2, d_4) + d_6$
oS	$D_i + d_i + d_1 + max(d_2, d_4) + d_6$	$D_i + d_i + d_1 + max(d_2, d_4) + d_6 + d_o$
o	$D_i + d_i + d_1 + max(d_2, d_4) + d_6 + d_o$	$Desconhecido$

Tabela 14 – Datas simbólicas de produção e de consumo para o cenário Sr2.

A Tabela 15 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sr4.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
iC	$D_i + d_i$	$D_i + d_i + d_1$
iS	$D_i + d_i$	$D_i + d_i + d_1 + d_2$
C_1	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_3$
S_1	$D_i + d_i + d_1 + d_2$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5$
P_1	$D_i + d_i + d_1$	$D_i + d_i + d_1 + d_2$
P_2	$D_i + d_i + d_1 + d_3$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5$
C_2	$D_i + d_i + d_1 + d_3$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7 + d_8$
S_2	$D_i + d_i + d_1 + max(d_2, d_3) + d_5$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7$
P_4	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7 + d_8$
oC	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7 + d_8$	$D_i + d_i + d_1 + max(d_2, d_3) + d_7 + d_8 + d_o$
oS	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7$	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7 + d_8 + d_o$
o	$D_i + d_i + d_1 + max(d_2, d_3) + d_5 + d_7 + d_8 + d_o$	$Desconhecido$

Tabela 15 – Datas simbólicas de produção e de consumo para o cenário Sr4.

A Tabela 16 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sa5.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
iC	$D_i + d_i$	$D_i + d_i + d_9$
iS	$D_i + d_i$	$D_i + d_i + d_9 + d_1 + d_2$
C_3	$D_i + d_i + d_9$	$D_i + d_i + d_9 + d_1$
P_1	$D_i + d_i + d_9 + d_1$	$D_i + d_i + d_1 + d_9 + d_2$
C_1	$D_i + d_i + d_9 + d_1$	$D_i + d_i + d_9 + d_1 + d_4$
S_1	$D_i + d_i + d_9 + d_1 + d_2$	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6$
P_3	$D_i + d_i + d_9 + d_1 + d_4$	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6$
S_7	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6$	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6 + d_{14}$
oC	$D_i + d_i + d_9 + d_1 + d_4$	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6 + d_{14} + d_o$
oS	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6 + d_{14}$	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6 + d_{14} + d_o$
o	$D_i + d_i + d_9 + d_1 + max(d_2, d_4) + d_6 + d_{14} + d_o$	*Desconhecido*

Tabela 16 – Datas simbólicas de produção e de consumo para o cenário Sa5.

A Tabela 17 mostra as datas simbólicas de produção e de consumo dos átomos do cenário Sa6.

Átomos	Datas Simbólicas de Produção	Datas Simbólicas de Consumo
i	D_i	$D_i + d_i$
iC	$D_i + d_i$	$D_i + d_i + d_9$
iS	$D_i + d_i$	$D_i + d_i + d_9 + d_1 + d_2$
C_3	$D_i + d_i + d_9$	$D_i + d_i + d_9 + d_1$
P_1	$D_i + d_i + d_9 + d_1$	$D_i + d_i + d_9 + d_1 + d_2$
S_1	$D_i + d_i + d_9 + d_1 + d_2$	$D_i + d_i + d_9 + d_1 + d_2 + d_{10}$
C_1	$D_i + d_i + d_9 + d_1$	$D_i + d_i + d_9 + d_1 + d_3$
S_3	$D_i + d_i + d_9 + d_1 + d_2 + d_{10}$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + d_5$
P_2	$D_i + d_i + d_9 + d_1 + d_3$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + d_5$
S_4	$D_i + d_i + d_9 + d_1 + d_2 + d_{10}$	$D_i + d_i + d_9 + d_1 + d_2 + d_{10} + d_{11}$
S_6	$D_i + d_i + d_9 + d_1 + d_2 + d_{10} + d_{11}$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) + d_{13}$
C_4	$D_i + d_i + d_9 + d_1 + d_3$	$D_i + d_i + d_9 + d_1 + d_3 + d_{12}$
C_2	$D_i + d_i + d_9 + d_1 + d_3 + d_{12}$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8$
S_5	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + d_5$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) + d_{13}$
S_2	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) +$ d_{13}	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) +$ $d_{13} + d_7$
P_4	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) +$ $d_{13} + d_7$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8$
oS	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) + max(d_5, d_{11}) +$ $d_{13} + d_7$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8 + d_o$
oC	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8$	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8 + d_o$
o	$D_i + d_i + d_9 + d_1 + max(d_2 + d_{10}, d_3) +$ $max(d_{12}, max(d_5, d_{11}) + d_{13} + d_7) + d_8 + d_o$	*Desconhecido*

Tabela 17 – Datas simbólicas de produção e de consumo para o cenário Sa6.

B.4 Árvores de prova da Lógica Linear para a identificação dos dos requisitos negativos nos serviços Web compostos

Árvore de prova para o cenário Snr2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{iS\vdash iS}\quad\dfrac{}{iC\vdash iC}}{iS,iC\vdash iS\otimes iC}\otimes_R\quad i\vdash i}{C_3\vdash C_3\quad iS,iC,iC\otimes iS\multimap i\vdash i}\multimap_L}{\dfrac{C_1\vdash C_1\quad P_1\vdash P_1}{C_1,P_1\vdash C_1\otimes P_1}\otimes_R\quad iS,C_3,C_3\multimap iC,t_i\vdash i}\multimap_L}{C_1,iS,P_1,C_1\otimes P_1\multimap C_3,t_9,t_i\vdash i}\otimes_L}{S_1\vdash S_1\quad C_1,iS\otimes P_1,t_1,t_9,t_i\vdash i}\multimap_L}{\dfrac{S_3\vdash S_3\quad S_4\vdash S_4}{S_3,S_4\vdash S_3\otimes S_4}\otimes_R\quad C_1,S_1,S_1\multimap iS\otimes P_1,t_1,t_9,t_i\vdash i}\multimap_L}{\dfrac{oC\vdash oC\quad P_3\vdash P_3}{oC,P_3\vdash oC\otimes P_3}\otimes_R\quad S_3,S_4,C_1,S_3\otimes S_4\multimap S_1,t_1,t_2,t_9,t_i\vdash i}\multimap_L}{S_6\vdash S_6\quad oC,P_3,S_3,S_4,P_3\otimes oC\multimap C_1,t_1,t_{10},t_2,t_9,t_i\vdash i}\multimap_L}{oC,P_3,S_3,S_6,S_6\multimap S_4,t_1,t_{10},t_2,t_4,t_9,t_i\vdash i}\multimap_L$$

Árvore de prova rotulada com as transições de disparo para o cenário Snr2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{iS \vdash iS}^{t_2\ t_i} \quad \dfrac{}{iC \vdash iC}^{t_9\ t_i}}{iS, iC \vdash iS \otimes iC}^{t_2\ t_9} \otimes_R \quad \dfrac{}{i \vdash i}^{t_i\ f_1}}{C_3 \vdash C_3}^{t_1\ t_9} \quad iS, iC, iC \otimes iS \multimap i \vdash i}^{f_1} \multimap_L^{t_i}}{C_1 \vdash C_1}^{t_4\ t_1} \quad \dfrac{}{P_1 \vdash P_1}^{t_2\ t_1}}{C_1, P_1 \vdash C_1 \otimes P_1}^{t_4\ t_2} \otimes_R \quad iS, C_3, C_3 \multimap iC, t_i \vdash i}^{f_1} \multimap_L^{t_9}}{C_1, iS, P_1, C_1 \otimes P_1 \multimap C_3, t_9, t_i \vdash i}^{f_1} \otimes_L^{t_1}}{S_1 \vdash S_1}^{t_{10}\ t_2} \quad C_1, iS \otimes P_1, t_1, t_9, t_i \vdash i}^{f_1} \multimap_L^{t_2}}{S_3 \vdash S_3}^{i_3\ t_{10}} \quad \dfrac{}{S_4 \vdash S_4}^{t_{11}\ t_{10}}}{S_3, S_4 \vdash S_3 \otimes S_4}^{i_3\ t_{11}} \otimes_R \quad C_1, S_1, S_1 \multimap iS \otimes P_1, t_1, t_9, t_i \vdash i}^{f_1} \multimap_L^{t_{10}}}{oC \vdash oC}^{i_1\ t_4} \quad \dfrac{}{P_3 \vdash P_3}^{i_2\ t_4}}{oC, P_3 \vdash oC \otimes P_3}^{i_1\ i_2} \otimes_R \quad S_3, S_4, C_1, S_3 \otimes S_4 \multimap S_1, t_1, t_2, t_9, t_i \vdash i}^{f_1} \multimap_L^{t_4}}{S_6 \vdash S_6}^{i_4\ t_{11}} \quad oC, P_3, S_3, S_4, P_3 \otimes oC \multimap C_1, t_1, t_{10}, t_2, t_9, t_i \vdash i}^{f_1} \multimap_L^{t_{11}}$$

$$oC, P_3, S_3, S_6, S_6 \multimap S_4, t_1, t_{10}, t_2, t_4, t_9, t_i \vdash i^{f_1}$$