A Study on Petri Net Based Refactoring of Workflows and Its Applications

(ペトリネットに基づくワークフローのリファクタリングと応用に関する研究)

September 2015

Ichiro Toyoshima

Graduate School of Science and Engineering,
Yamaguchi University

Abstract

Business Process Management (BPM) is very important to many organizations. BPM allows organizations to improve their consistency and efficiency. Nowadays, the importance of workflow management has been increasing since business models are changing rapidly to adapt to market changes and law revision.

Petri net is a graphical and mathematical model for concurrent and discrete event systems. Actual business workflows in the real world can be represented with a restricted subclass of Petri nets called as Workflow net (WF-net). We can check the consistency of a workflow by analysing its corresponding WF-net. For example, "Will this workflow terminate correctly?", "How many states does this workflow have?". In case of dealing with huge and complex workflows, automatic checking is becoming more important. Petri net ' s theory and applications allow us to analyse actual workflows.

There is a subclass of WF-net, called Extended Free Choice WF-net (EFC WF-net for short). Many actual workflows can be modeled as EFC WF-nets. On the other hand, there is another subclass of WF-net, called Well-structured WF-net (WS WF-net for short). WS WF-net has more checking algorithms than EFC WF-net. It is natural for us to try to transform EFC WF-net to WS WF-net to check some properties.

Yamaguchi has defined a problem, called *acyclic EFC WF-net refactoring problem*: given an acyclic EFC WF-net, transform it to an acyclic WS WF-net without changing its observable behavior. If the acyclic EFC WF-net is refactored to an acyclic WS WF-net, we can use the analysis methods of WS WF-nets to analyze the EFC WF-net.

Not all EFC WF-nets are refactored to WS WF-nets. So that, we judge to refactor a target EFC WF-net. It is important to know sufficient conditions of refactorizability. If some conditions are "true", we apply refactoring algorithms to the EFC WF-net, and obtain the output WS WF-net to check the consistency of the original EFC WF-net.

In this thesis, we focus sufficient conditions of refactorizabilty of WF-net and refactoring algorithms to study refactoring actual workflows. This thesis is organized as follows :

Chapter 1 presents background and overview of this thesis.

Chapter 2 presents the definitions of Petri net, and WF-net. Next, we show mainly two concepts called soundness and branching bisimilarity. Soundness corresponds to workflows' logical correctness. Branching bisimilarity corresponds to equivalence between workflows' behavior. Furthermore, we propose WF-net refactoring problem and refactorizability problem.

Chapter 3 presents three sufficient conditions of refactorizability. The first two conditions are based on paths in WF-net. The last one is based on places in WF-net.

Chapter 4 presents refactoring algorithm by using the sufficient conditions proposed in Chapter 3. The algorithm enables us to refactor actual WF-nets automatically.

Chapter 5 presents three applications of refactoring. The first is *Reachability Checking*. The second is *State Number Calculation*. The third is *Response Property Checking*.

Chapter 6 introduces an advanced concept "Timed Branching Bisimulation (TBB for short)". TBB is an extension of branching bisimilarity. Next, we propose new equlity criterion *Timed projection Inheritance*. Furthermore, we propose new refactoring methods by reduction for timed system based on the above equality.

Chapter 7 gives the conclusion of this thesis and presents the future works.

The results presented in this thesis reveal the sufficient conditions of EFC WF-net refactoring and related refactoring algorithms. It enables us to analyze EFC WF-net to wider properties more easily. In the future works, we will try to apply our method to upstream process in workflow system design. It may reduce the costs in the process.

要旨

題目： ペトリネットに基づくワークフローのリファクタリングと応用に関する研究

ワークフローの管理は，官公庁や企業のシステムを円滑に運営する上で必須の活動である．管理内容は多岐にわたるが，その正しさを検証することは重要な活動の一つである．企業の分割・合併や，インターネットの普及によるビジネス環境の多様化等によりワークフローの管理は重要度を増しつつある．

ワークフローネットはペトリネットのサブクラスであり，ワークフローに対応したモデルである．ワークフローネットのモデルを活用することにより，ワークフローに関する様々な正しさを計算機で解析することが可能になる．例えば到達可能性の解析は，ワークフローが所望の状態に到達し得るか否かを設計段階で知るための重要な検証である．

残念ながら，各種の解析技法は全てのワークフローに適用できる訳ではない．例えば Well-Structured ワークフローネット (以下，WS WF-net) と呼ばれるワークフローネットのサブクラスは，上位クラスである拡張自由選択ワークフローネット (Extend Free Choice WF-net, 以下 EFC WF-net) よりも相対的により多くの解析技法を有する．しかし現実のワークフローをモデル化するには，EFC WF-net の自由度が必要であると考えられており，実用上これを解析できることが望まれる．

ここで，EFC WF-net の振舞いを維持したまま，WS WF-net に変換することができれば，各種の解析技法を適用して元のネットの性質を解析することが可能になる．このような変換はワークフローネットのリファクタリングと呼ばれている．どのようなネットがリファクタリング可能なのか，その十分条件を明らかにすることは，応用面からの重要な要請である．

本論文は，ワークフローネットのリファクタリングと，その応用に関するものである．１章では問題の背景と，本論文が扱う範囲の全体像を示した．２章では離散数学の基本的な概念の導入から始め，グラフ，ペトリネットに関する数学的諸定義を導入した．次に先行研究におけるワークフローネット及びそのリファクタリングに関する定義と，それらの定義から導かれる重要な諸性質を示した．本論文で特に重要な役割を果たす「健全性」と

「分岐双模倣性」の２つの概念には特に説明を加えた．健全性はワークフローネットの論理的正しさを表す性質である．分岐双模倣性はシステムの等価性を表す概念の一種であり，２個のワークフローについて，各々外部アクションが付与されたトランジションの発火にのみ注目して構成される遷移グラフ間に対応関係が成立する場合に，両者は分岐双模倣関係と呼ばれる．分岐双模倣関係にある２つのワークフローは，外部からは区別できない．分岐双模倣関係を保持したままネットの構造を簡素化することがワークフローネットのリファクタリングである．最後に同章では，リファクタリング問題とリファクタリング可能性問題の定義を与えた．

３章は本論文の主要結果である．Handle, Bridge と呼ばれるパス構造の存在有無が，リファクタリング可能性の十分条件となっている事実を以下の定理として示した．

【定理】健全な非巡回 EFC WF-net N がいずれかの条件を満たすとき，N は WS WF-net N' へリファクタリング可能である

- N の短絡ネットに PT-handle が存在しない
- N の短絡ネットに TP-handle が存在せず，かつ存在する PT-handle は各々長さ 1 の TP-bridge を持つ

加えて同章では，implicit place と呼ばれる，一種の冗長性に相当するプレースの定義を導入した．更に次章において特定の前提の下で implicit place を除去することが，リファクタリングに相当することを定理として示し，implicit place を発見するための必要十分条件とそれを用いた多項式時間アルゴリズムを定義した．

４章では, ３章で証明した定理相互の関係を比較検討した．これらの定理はそれぞれ独立しており，各々の事前条件と事後条件を突き合わせて包含関係をチェックした．その結果をもとにこれらを組み合わせることで，効果的にリファクタリングを行う統合アルゴリズムを提案した．

５章ではリファクタリングの応用例として, システムの上流設計段階で効果を発揮するであろう３つの事例を提示した．

- 到達可能性判定
- 状態数計算
- アクション間の因果関係チェック

６章では発展的な話題として，リファクタリングの概念の時間付きシステムへの拡張を論じた．最初に時間付きペトリネットと呼ばれる時間制約を持つシステムを導入した．次

に時間ペトリネット間で定義される新たな分岐双模倣の概念を提案し，時間付き振舞いの等価性を保持したまま構造を変換する一種のリファクタリングアルゴリズムを提案した．最後に 7 章では結果と今後の展望について総括した．

本論文により，ワークフローのリファクタリングに関する十分条件の一端と，その応用方法が明らかにされた．これらの成果により，実際的なワークフローに対して設計段階で各種の検証を適用する可能性が向上した．

Acknowledgements

I would like to express my deep appreciation to those who have supported me throughout this work.

I am very grateful to Associate Professor Shingo Yamaguchi for invaluable guidance. He has been a constant source of continuous support and encouragement throughout this work.

I also thank the other members of the examination committee for this thesis: Professors Shinya Matsufuji, Masaaki Ishikawa, Masanao Obayashi, Qi-Wei Ge, for their careful reading and precious comments on this thesis.

I would like to express appreciation to Professor Naoshi Uchihira (JAIST), and TOSHIBA corporations' members, Dr.Takashi Kamitake, Dr.Kondo Koichi, Dr.Minoru Yonezawa, Mr.Yoshiyuki Sakamoto, Mr.Naoki Imasaki, Mr.Chiharu Kakita, Mr.Shinji Yurino, Dr.Hiromasa Shin, Dr.Mikito Iwamasa and Dr.Keiichi Handa for their valuable advice and encouragement.

I also want to thank students of Associate Professor Yamaguchi's laboratory for their discussion and many contribution to this work.

And finally, I express my private but sincere gratitude to my wife, Naoko, my daughter Honoka, and my parents Tomoharu and Masako, for their invaluable support and encouragement.

List of Symbols

$x \in X$:	x is an element of set X.
$X \subset Y$:	Set X is contained in set Y.
$X \cup Y$:	Union of sets X and Y.
$X \cap Y$:	Intersection of sets X and Y.
$X \times Y$:	Products of sets X and Y.
$X - Y$:	Difference of sets X and Y.
$\emptyset$:	Empty set.
$\mathbb{N}$:	The set of natural numbers.
$\lvert X \rvert$:	The cardinality of set X.
$\lfloor x \rfloor$:	Floor function – the maximum integer not grater than x.
$\lceil x \rceil$:	Ceiling function – the minimum integer not less than x.

Contents

Chapter 1

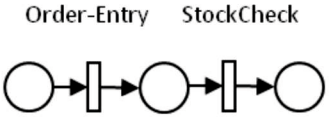
Introduction

Business Process Management (BPM) is very important to many organizations. BPM allows organizations to improve their consistency and efficiency. Nowadays, the importance of workflow management has been increasing since business models are changing rapidly to adapt to market changes and law revision [1][2].

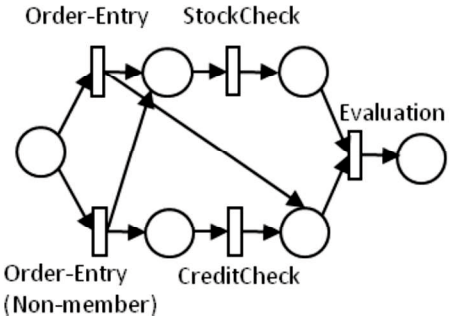
Petri net is a graphical and mathematical model for concurrent and discrete event systems [3][4][5]. Actual business workflows in the real world can be represented with a subclass of Petri nets called as Workflow net (WF-net) [6][7]. We can check the consistency of a workflow by analyzing its corresponding WF-net. For example, "Will this workflow terminate correctly?", "How many states does this workflow have?". In case of dealing with huge and complex workflows, automatic checking is becoming more important. Petri net ' s theory and applications allow us to analyze actual workflows.

There is a subclass of workflow net, called Extended Free Choice WF-net (EFC WF-net for short). EFC WF-nets is known to be able to represent most real world workflows [8][9]. Unfortunately, it is difficult to solve analysis problem of EFC WF-nets using existing checking algorithm. On the other hand, there is another subclass of WF-net, called Well-Structured WF-net (WS WF-net for short). There exist more analysis methods for WS WF-net.

It is true that we can design workflows as WS WF-nets in the beginning of development. However, in the latter stage, the workflows are often forced to change [10][11][12]. For example, adding a new credit check process which is located beside existing processes concurrently. We show a sample of workflow changing in Fig. 1.1. (a)A virtual e-commerce company operate exclusive service for registered members, initially. (b)One day, the service disclosed for non-members. Non-member needs credit check before contract. New WF-net is expected to have a new credit check process and concurrent communication with existing processes. We can see that the complexity of the net increases. As a result, workflows that correspond to EFC WF-net



(a) A virtual e-commerce company's WF-net $N_{enclose}$ before changing.



(b) A new WF-net $N_{disclosed}$ has a new concurrent process to $N_{enclose}$.

Figure 1.1: An Example of WF-net changing.

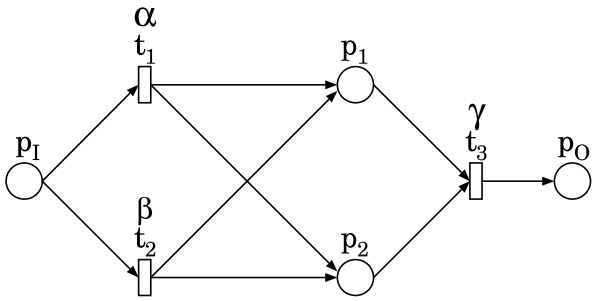
are generated involuntarily. It is natural for us to try to transform EFC WF-net to WS WF-net to check some properties.

For many years, reduction techniques have been studied in Petri net research field. Reduction of Petri nets involves simplifying a complex Petri net to a simpler one, and at the same time preserves important properties concerned by designers.

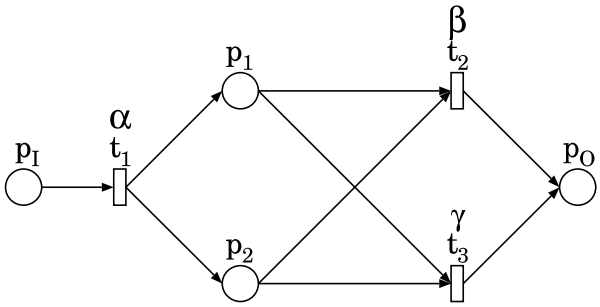
Best [13][14] proposed reduction rule which removes EFC-structure from the original net, it also preserves *liveness* [16] and *boundness* [16] property. Desel [15] proposed another rule which removes dual of EFC-structure, it preserves *well-formedness*. Another six reduction rules were studied and summarized by Murata [16]. Murata's rules preserve liveness, safeness and boundness.

In recent years, there are some remarkable progress about reduction of WF-nets. Polyvyanyy [17] proposed a WF-net reduction algorithm. It searches for concurrent elements (*modular decomposition tree* in Ref. [17]) in ordering relations graph corresponding to the original WF-net. It outputs WS WF-net by reconstructing the concurrent elements of the net. The study differs in using ordering relation graph (a kind of state graph) from us. Our approach is more comprehensive and intuitive, since we transform the paths in the WF-net, directly.

Yamaguchi [18] has defined a problem, called *acyclic EFC WF-net refactoring problem* : given an acyclic EFC transform it to an acyclic WS WF-net without changing its observable behavior. That is an analogy of *Refactoring* in software engineering [20].



(a) A CB WF-net with TP-cross structure.



(b) A CB WF-net with PT-cross structure.

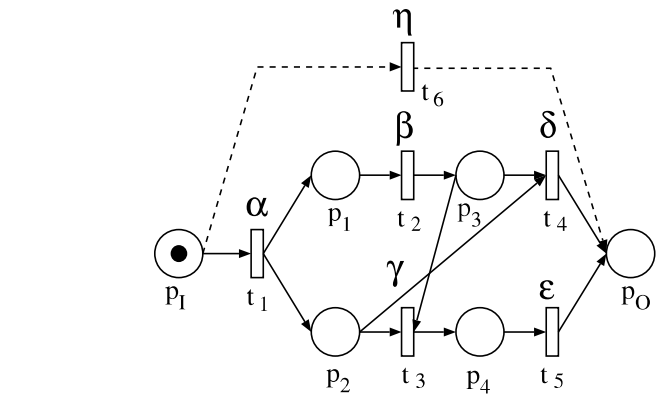
Figure 1.2: Examples of CB WF-net.

Not all EFC WF-nets are refactored to WS WF-nets. So that, we can decide to refactor the target EFC WF-net. It is necessary to know sufficient conditions of refactorizability. If some conditions are "true", we can apply refactoring algorithms to the EFC-net, and obtain the output WS WF-net to check the consistency of the original EFC WF-net. Yamaguchi [18] has proposed one sufficient condition of refactorizability : Let N_X be an acyclic EFC but non-WS WF-net whose every label in N_X except τ is unique. If N_X is CB then there is an acyclic WS WF-net N_Y such that $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$ and every label in N_Y except τ is unique.

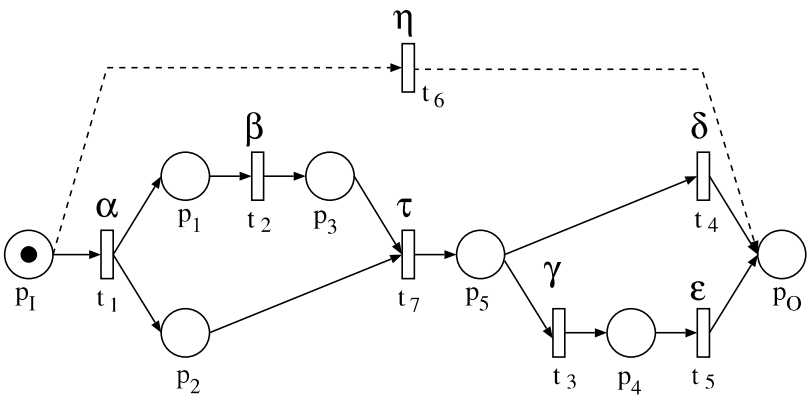
CB means that the net has PT or TP cross structure. Figure 1.2 (a) and (b) shows each case of CB WF-net. Let us consider a sound acyclic EFC WF-net N_1 shown in Fig. 1.3 (a). N_1 has a CB (PT) structure, i.e. places p_2, p_3 and transitions t_3, t_4, and the structure is a cut-set of N_1.

Since N_1 satisfies the above sufficient condition, it can be refactored to an acyclic WS WF-net N_1' shown in Fig. 1.3 (b). Next, let us extend N_1 by adding a path $p_I t_6 p_O$. The extended WF-net does not any longer satisfy the sufficient condition, because the CB (PT) structure is not a cut-set of the net. The complex structure often may reveal in actual workflows. We have to generalize the sufficient condition to cope with real world's problem.

By a new sufficient conditions proposed in this study, we can say that N_1' is refactorizable, and we can refactor it to WS WF-net shown in Fig. 1.3 (b) automatically.



(a) Acyclic EFC WF-net N_1



(b) Acyclic WS WF-net N_1'

Figure 1.3: An example refactoring which can be refactored for the first time by this study

In this thesis, we study about more general sufficient conditions of refactorizabilty of WF-net and develop refactoring algorithm and its applications. This thesis is organized as follows :

In Chapter 2 presents the definitions of Petri net, and WF-net. Next, we introduce mainly two concepts called soundness and branching bisimilarity. Soundness corresponds to workflows' logical correctness. Branching bisimilarity corresponds to equivalence between workflows' behavior. Furthermore, we propose WF-net refactoring problem and refactorizability problem. We will explain its definition with own mathematical background elaborately.

Chapter 3 presents three sufficient conditions of refactorizability [21]. The first two conditions are based on paths called "PT-handle" and "TP-handle" in EFC WF-net. The last one is based on places called "implicit place" in EFC WF-net. We prove that the paths' non-existence and places existence are pre-conditions of refactorizability. And we give refactoring rules based on above sufficient conditions.

Chapter 4 presents an refactoring algorithm by using partial rules proposed in Chap.3 [22].

The algorithm composes from the rules. The rules are applied in sequence. The sequence is optimized to deal with a large subclass of EFC WF-net.

Chapter 5 presents three applications of refactoring. The fist two cases are regarding reachability analysis. It is common analysis to check correctness of workflow instances. The third case is about state number calculation. It enables model checking to EFC WF-net properties. Estimating state number is needed by developers for model checking.

Chapter 6 introduces an advanced concept "Timed Branching Bisimulation (TBB for short)". TBB is an extension of branching bisimilarity. Next, we propose a new equality criterion *Timed Projection Inheritance*. Furthermore, we propose new reduction methods for timed system based on the above equality [23].

Chapter 7 gives the conclusion of this thesis and gives future works.

Chapter 2

Preliminaries

This chapter presents definitions and basic properties about WF-net and its refactoring problem.

2.1 Petri net and Workflow net

Petri net [16][5] is the basis of WF-net. A (labeled) Petri net PN is a four tuple $PN = (P, T, A, \ell)$, where P and T are disjoint finite sets of *places*, *transitions*, respectively, $A \subseteq (P \times T) \times (T \times P)$ is a set of arcs, and $\ell : T \to \mathcal{A} \cup \{\tau\}$ is a *labeling function* of transitions, where $\mathcal{A}$ denotes the set of all possible labels except a designated label τ The label of transitions may be used to abstract the transition from the action. Let x be a place or a transition, $\bullet x$ and $x\bullet$ are defined as $\bullet x = \{y | (y, x) \in A\}$ and $x\bullet = \{y | (x, y) \in A\}$, respectively.

A (labeled) WF-net is a labeled Petri net which represents a workflow.

Definition 2.1 (WF-net [6]) A labeled Petri net $N=(P, T, A, \ell)$ is a (labeled) WF-net iff (i) N has a single source place p_I ($\overset{N}{\bullet}p_I=\emptyset$ and $\forall p \in (P-\{p_I\}) : \overset{N}{\bullet}p \neq \emptyset$) and a single sink place p_O ($p_O\overset{N}{\bullet}=\emptyset$ and $\forall p \in (P-\{p_O\}) : p\overset{N}{\bullet} \neq \emptyset$), and (ii) Every place and transition is on a path from p_I to p_O. (iii) Let t be a transition, if $\ell(t) = \tau$, the firing of t is *unobservable*. Otherwise, the firing of t is *observable*. □

Every label except τ is unique. (i.e. for any pair t_i, t_j of transitions, if $\ell(t_i) \neq \tau$ and $\ell(t_j) \neq \tau$ then $\ell(t_i) \neq \ell(t_j)$).

Let $N=(P, T, A, \ell)$ be a WF-net. We represent a marking of N as a bag over P. A marking is denoted by $M=[p^{M(p)} | p \in P, M(p)>0]$. $M_X=M_Y$ denotes that $\forall p \in P : M_X(p)=M_Y(p)$. $M_X \geq M_Y$ denotes that $\forall p \in P : M_X(p) \geq M_Y(p)$. M_X+M_Y denotes $[p^n | p \in P, n = M_X(p)+M_Y(p)]$. M_X-M_Y denotes $[p^n | p \in P, n = max(M_X(p)-M_Y(p), 0)]$. A transition t is said to be firable in a marking M if $M \geq \overset{N}{\bullet}t$. Firing t in M results in a new marking M'. M' is defined $M' = M + t\overset{N}{\bullet} - \overset{N}{\bullet}t$. This

is denoted by $M[N,t\rangle M'$. M' is said to be reachable from M if there exists a firing sequence of transitions transforming M to M'. The set of all possible markings reachable from M is denoted by $R(N,M)$.

There are two important subclasses of WF-nets: *WS* and *EFC*. A structural characterization of good workflows is that two paths initiated by a transition/place should not be joined by a place/transition. WS is derived from this structural characterization. To give the formal definition of WS, we introduce some notations. The WF-net obtained by connecting p_O with p_I via an additional transition t^* is called the *short-circuited net* of N, denoted by $\overline{N}$ $(=(P, T\cup\{t^*\}, A\cup\{(p_O,t^*),(t^*,p_I)\}, \ell\cup\{(t^*,\tau)\}))$. A path $\rho = n_1n_2\cdots n_k$ is said to be *elementary* iff, for any two nodes n_i and n_j on ρ, $i\neq j\Rightarrow n_i\neq n_j$. Let c be an elementary circuit in $\overline{N}$, and ν an elementary path from a node x to another node y in $\overline{N}$. ν is called a *handle* [16] of c if ν shares exactly two nodes, x and y, with c. Let c be an elementary circuit in $\overline{N}$, and d a handle of c, ξ is an elementary path from a node x to another node y in $\overline{N}$. ξ is called a *bridge* [24] between c and d iff c shares exactly x and d shares exactly y, with ξ. A handle (a bridge) from a node x to another node y is called an *XY-handle* (an *XY-bridge*), where if $x\in P$ then X is P, otherwise X is T; if $y\in P$ then Y is P, otherwise Y is T. For example, a handle from a place to a transition is a PT-handle.

Next, we gives the three subclasses of WF-net. They play a central role in this thesis.

Definition 2.2 (Well-structured(WS) WF-net) A WF-net N is WS, if there are neither TP-handles nor PT-handles in $\overline{N}$. □

Definition 2.3 (Free Choice(FC) WF-net) A WF-net N is FC, if $\forall p_1, p_2\in P$: ${}^{N}p_1^{\bullet}\cap {}^{N}p_2^{\bullet}\neq\emptyset \Rightarrow |{}^{N}p_1^{\bullet}|=|{}^{N}p_2^{\bullet}|=1$. □

Definition 2.4 (Extended Free Choice(EFC) WF-net) A WF-net N is EFC, if $\forall p_1, p_2\in P$: ${}^{N}p_1^{\bullet}\cap {}^{N}p_2^{\bullet}\neq\emptyset \Rightarrow {}^{N}p_1^{\bullet}={}^{N}p_2^{\bullet}$. □

Note that those subclasses are defined independent of labeling.

Acyclic FC WF-nets is a proper subclass of acyclic EFC WF-nets. And acyclic WS WF-nets is a proper subclass of acyclic FC WF-nets. We can see the relations between the subclasses and the images of their own structures in Fig. 2.1.

2.2 Soundness and Branching Bisimilarity

We introduce two important concepts in WF-nets: *soundness* and *branching bisimilarity*. Soundness is a criterion of logical correctness [25][27][28], i.e. soundness is independent of labeling.

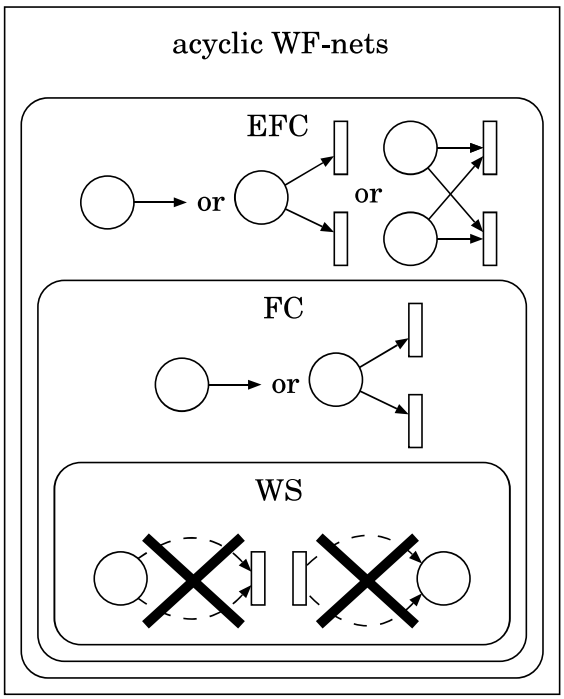


Figure 2.1: The Relations between the subclasses of WF-net.

Definition 2.5 (*soundness* [6]) A WF-net N is sound iff
(i) $\forall M{\in}R(N,[p_I])$: $\exists M'{\in}R(N,M)$: $M'{\geq}[p_O]$; (ii) $\forall M{\in}R(N,[p_I])$:$M{\geq}[p_O]{\Rightarrow}M{=}[p_O]$; and (iii) there is no dead transition in $(N,[p_I])$. □

Soundness of EFC WF-nets can be checked in polynomial time on the basis of Rank Theorem [15]. Any sound EFC WF-net is safe (Lemma 3 in Appendix A of Ref. [29]). Any acyclic WS WF-net is sound [18]. As for acyclic FC WF-nets, we have the following necessary and sufficient condition on soundness.

Property 2.1 An acyclic FC WF-net N is sound iff (i) No circuit of $\overline{N}$ has TP-handles; and (ii) If $\overline{N}$ has PT-handles, then each PT-handle has a TP-bridge from the handle to the circuit [24]. □

Unfortunately, for acyclic EFC WF-nets, such a structural necessary and sufficient condition is unclear.

Branching bisimilarity is widely used as an equivalence relation on WF-nets. Branching bisimilarity intuitively equates WF-nets which have the same external behavior. The behavior of a WF-net N $(=(P,T,A,\ell))$ is captured by the reachability graph of $(N,[p_I])$. It is denoted by $G{=}(V,E)$, where $V{=}R(N,[p_I])$, $E{=}\{(M,\ell(t),M')|M,M'{\in}V,t{\in}T,M[N,t\rangle M'\}$, where $(M,\ell(t),M')$ means that an edge (M,M') has a label $\ell(t)$. Let $M,M'{\in}V$, $\alpha{\in}\ell(T)$. We write $M[N,\alpha\rangle M'$ if M'

is reachable from M by following an edge labeled as α. We write $M[N, \tau^*\rangle M'$ if M' is reachable from M by following any number of edges labeled as τ. We write $M[N, (\alpha)\rangle M'$ if either (i) $\alpha{=}\tau$ and $M{=}M'$; or (ii) $M[N, \alpha\rangle M'$.

Definition 2.6 (*branching bisimilarity* [30][31]) Let G_X and G_Y be the reachability graphs of a WF-net $(N_X, [p_I^X])$ and another WF-net $(N_Y, [p_I^Y])$, respectively. A binary relation $\mathcal{R}$ ($\subseteq R(N_X, [p_I^X]) \times R(N_Y, [p_I^Y])$) is branching bisimulation iff (i) if $M_X \mathcal{R} M_Y$ and $M_X[N_X, \alpha\rangle M'_X$, then $\exists M'_Y, M''_Y \in R(N_Y, [p_I^Y])$: $M_Y[N_Y, \tau^*\rangle M''_Y$, $M''_Y[N_Y, (\alpha)\rangle M'_Y$, $M_X \mathcal{R} M''_Y$, and $M'_X \mathcal{R} M'_Y$; (ii) if $M_X \mathcal{R} M_Y$ and $M_Y[N_Y, \alpha\rangle M'_Y$, then $\exists M'_X, M''_X \in R(N_X, [p_I^X])$: $M_X[N_X, \tau^*\rangle M''_X$, $M''_X[N_X, (\alpha)\rangle M'_X$, $M''_X \mathcal{R} M_Y$, $M'_X \mathcal{R} M'_Y$; and (iii) if $M_X \mathcal{R} M_Y$ then $(M_X{=}[p_O^X] \Rightarrow M_Y[N_Y, \tau^*\rangle [p_O^Y])$ and $(M_Y{=}[p_O^Y] \Rightarrow M_X[N_X, \tau^*\rangle [p_O^X])$. $(N_X, [p_I^X])$ and $(N_Y, [p_I^Y])$ are called *branching bisimilar*, denoted by $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$, iff there exists a branching bisimulation $\mathcal{R}$ between G_X and G_Y. □

2.3 Refactoring and Refactorizability Problem

The formal definition of refactoring and refactorizability problem is given as follows.

Definition 2.7 (refactoring problem [18])

Input: Acyclic EFC WF-net $N_X{=}(P_X, T_X, A_X, \ell_X)$, where every label in N_X except τ is unique, i.e. for any pair t_i, t_j of transitions, if $\ell(t_i){\neq}\tau$ and $\ell(t_j){\neq}\tau$ then $\ell(t_i){\neq}\ell(t_j)$.

Output: Acyclic WS WF-net $N_Y{=}(P_Y, T_Y, A_Y, \ell_Y)$ such that (i) $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$; and (ii) Every label in N_Y except τ is unique. □

Definition 2.8 (refactorizability problem [18])

Instance: Acyclic EFC WF-net $N_X{=}(P_X, T_X, A_X, \ell_X)$, where every label in N_X except τ is unique, i.e. for any pair t_i, t_j of transitions, if $\ell(t_i){\neq}\tau$ and $\ell(t_j){\neq}\tau$ then $\ell(t_i){\neq}\ell(t_j)$.

Question: Is there an acyclic WS WF-net $N_Y{=}(P_Y, T_Y,$
$A_Y, \ell_Y)$ such that (i) $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$; and (ii) Every label in N_Y except τ is unique? □

Constraint (ii) prohibits duplication of any transition with an observable label. A transition firing is performed by resources (workers and/or machines). If the transition is duplicated, it would share the resources with its duplicate. This makes it difficult that those resources are scheduled.

Yamaguchi has given a necessary condition on the problem.

Property 2.2 Let N_X be an acyclic EFC but non-WS WF-net whose every label in N_X except τ is unique. If N_X is not sound then there is no acyclic WS WF-net N_Y such that $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$ and every label in N_Y except τ is unique. □

This property implies that if a given acyclic EFC WF-net is not sound, then we cannot refactor it to an acyclic WS WF-net. Since soundness of EFC WF-nets can be checked in polynomial time, the above necessary condition can also be checked in polynomial time.

Yamaguchi has also given a sufficient condition on the problem. He defined a subclass of acyclic EFC but non-WS, named cross-bridged (CB for short). A CB WF-net is intuitively only one of a key structure of EFC and its dual structure as a cut-set of the net. For the detail of CB, refer to Ref. [18].

Property 2.3 Let N_X be an acyclic EFC but non-WS WF-net whose every label in N_X except τ is unique. If N_X is CB then there is an acyclic WS WF-net N_Y such that $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$ and every label in N_Y except τ is unique. □

2.4 Implicit Place

Definition 2.9 (Implicit place [8]) In a Petri net $N = ((P, T, A, \ell), M_0)$, a place $p \in P$ is called implicit in (N, M_0) iff $\forall t \in p^{\overset{N}{\bullet}}, \forall M \in R(N, M_0) : M \geq {}^{\overset{N}{\bullet}}t \setminus \{p\} \Rightarrow M \geq {}^{\overset{N}{\bullet}}t$. □

2.5 Remarks

This chapter presents mathematical definitions and properties about WF-net. In the next chapters, we use them to explain new theorems and other results.

Chapter 3

Sufficient Conditions of Refactorizability and Refactoring Rules

3.1 Refactorizability Problem

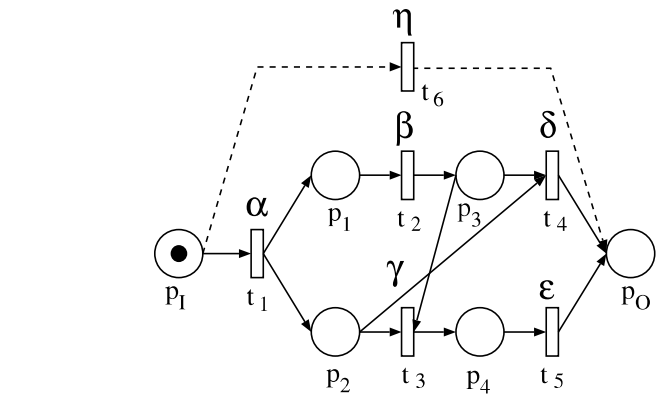
A WF-net [6] is a Petri net [16] which represents a workflow. There are two important subclasses of WF-nets: extended free choice (EFC for short) and well-structured (WS for short). It is known that most actual workflows can be modeled as EFC WF-nets; acyclic WS is a subclass of acyclic EFC but has more analysis methods, e.g. polynomial time algorithm on the reachability problem [32].

Yamaguchi [18] has defined a problem, called *acyclic EFC WF-net refactorizability problem*, that decides whether a given acyclic EFC WF-net can be transformed to an acyclic WS WF-net without changing its observable behavior. If the acyclic EFC WF-net is refactored to an acyclic WS WF-net, we can use the analysis methods of WS WF-nets to analyze the EFC WF-net.

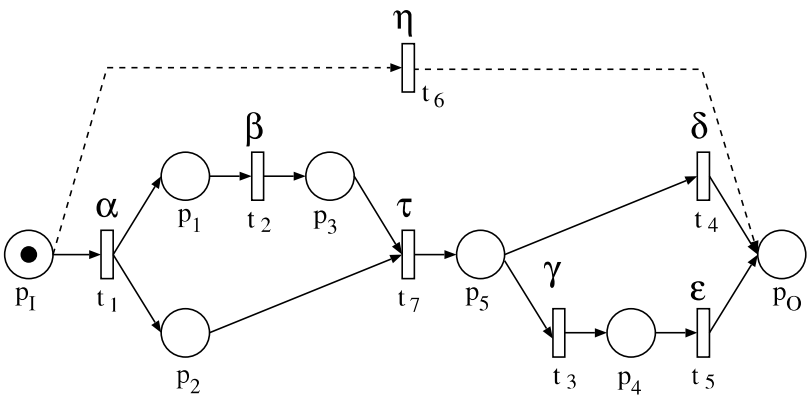
Property 3.1 ([18]) Let N_X be an acyclic EFC but non-WS WF-net whose every label in N_X except τ is unique. If N_X is CB then there is an acyclic WS WF-net N_Y such that $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$ and every label in N_Y except τ is unique. □

He has also given a necessary condition and a sufficient condition on the problem. The necessary condition is soundness. The sufficient condition is that a given WF-net has only one of a key structure of EFC [16] and its dual structure [24] as a cut-set of the net.

Let us consider a sound acyclic EFC WF-net N_1 shown in Fig. 3.1 (a). N_1 has a single EFC structure, i.e. places p_2, p_3 and transitions t_3, t_4, and the structure is a cut-set of N_1. Since N_1 satisfies Yamaguchi's sufficient condition, it can be refactored to an acyclic WS WF-net N_1' shown in Fig. 3.1 (b). Next, let us extend N_1 by adding a path $p_I t_6 p_O$. The extended WF-net



(a) Acyclic EFC WF-net N_1



(b) Acyclic WS WF-net N_1'

Figure 3.1: An instance of refactorizability problem.

does not any longer satisfy the sufficient condition, because the EFC structure is not a cut-set of the net. In fact, we can refactor the extended WF-net like Fig. 3.1 (b), but we cannot say whether it can be refactored by using only the sufficient condition. So it is necessary to improve the sufficient condition.

We propose two superior sufficient conditions on the problem. We try to remove a restriction from the previous sufficient condition: cut-setness. To do so, we make use of a key structural objects , called handles and bridges. And we construct refactoring procedures based on the conditions. Furthermore, we apply the procedures to a sample workflow reachability problem, and confirm usefulness of the procedures for the enhancement of the analysis power of acyclic EFC WF-nets.

The remainder of this chapter is organized as follows : In Sec. 3.2, we show two sufficient conditions of refactorizability regarding handles. In Sec. 3.3, we gives another sufficient conditions based on some kind of place. We give remarks in Sec. 3.4.

3.2 Sufficient Conditions on Refactorizability Problem regarding Handles

In this section, we propose two superior sufficient conditions on the soundizability problem. We try to remove a restriction from the previous sufficient condition: cut-setness. To do so, we make use of handles and bridges.

Analysis Let us first consider two instances of the refactorizability problem.

The first instance is an acyclic EFC WF-net N_1 shown in Fig. 3.1 (a). Note that $\overline{N_1}$ has TP-handles but no PT-handle. The answer of this instance is yes, i.e. we can refactor N_1 to an acyclic WS WF-net $N_1{}'$ shown in Fig. 3.1 (b). In fact, there exists a branching bisimulation relation between $(N_1, [p_I])$ and $(N_1{}', [p_I])$. It is illustrated in Fig. 3.2. Even if adding a path $p_I t_6 p_O$, the answer is still yes.

Next, the second instance is an acyclic EFC WF-net N_2 shown in Fig. 3.3 (a). Note that $\overline{N_1}$ has PT-handles but no TP-handle. The answer of this instance is also yes, i.e. we can refactor N_2 to an acyclic WS WF-net $N_2{}'$ shown in Fig. 3.3 (b). Even if adding a path $p_I t_6 p_O$, the answer still yes.

From these analysis results, we deduce that PT-handles and TP-handles play a central role of the refactorizability problem.

Sufficient Conditions In general, the short-circuited net of an acyclic sound EFC has PT-handles and TP-handles. We restrict our analysis to two special cases: case of no PT-handle; case of no TP-handle. The restriction helps us to comprehend the constitution of handles and bridges in the net. For each case, we propose a sufficient condition of the soundizability problem. Each condition is represented as a constitution of handles and bridges in the given net. In 'case of no PT-handles', we can refactor acyclic sound EFC WF-net to acyclic sound WS FC WF-net by ϕ_{Best}. On the other hand, 'case of no TP-handles', we can refactor acyclic sound FC WF-net to acyclic sound WS FC WF-net by ϕ_{Desel}, under condition of that the net with PT-handles has TP-bridge with length one.

Case of no PT-Handle

We first propose a sufficient condition for acyclic EFC WF-nets with no PT-handle.

Theorem 3.1 Let N_X be a sound acyclic EFC WF-net, whose every label in N_X except τ is

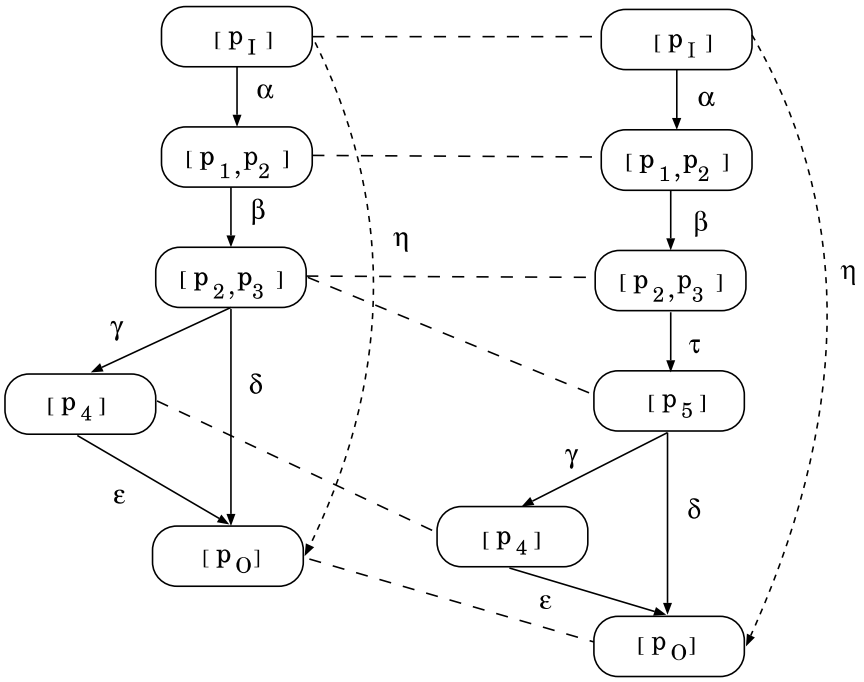


Figure 3.2: Branching bisimulation between $(N_1, [p_I])$ and $(N_1{}', [p_I])$.

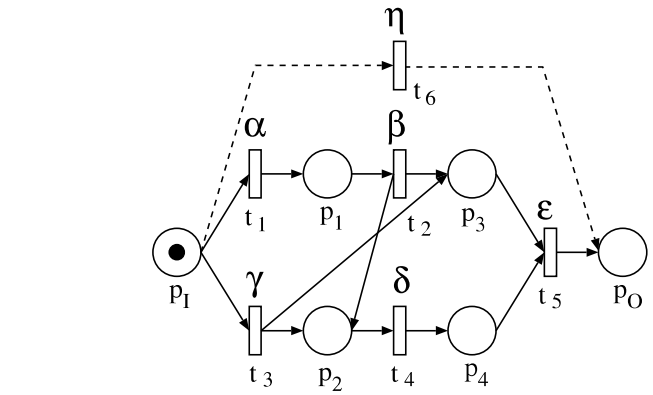
unique. If $\overline{N_X}$ has no PT-handle, then there is an acyclic WS WF-net $N_Y = (P_Y, T_Y, A_Y, \ell_Y)$ such that (i) $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$, and (ii) every label in N_Y except τ is unique. □

This theorem means that a sound acyclic EFC WF-net N is refactorable to an acyclic WS WF-net if $\overline{N}$ has no PT-handle[26]. If $\overline{N}$ has no PT-handle then it has TP-handles. To remove the TP-handles, we can apply a transformation rule, denoted by ϕ_{Best}, which has been proposed by Best [13] (See Fig. 4). The definition of ϕ_{Best} is given later. ϕ_{Best} can transform an EFC structure to its equivalent FC structure [16]. Using rule ϕ_{Best} repeatedly, we can obtain an acyclic FC WF-net $\phi^*_{Best}(N)$ from N. In order to prove this theorem, we show the following: (i) $\overline{\phi^*_{Best}(N)}$ has no PT-handle (Lemma 3.1); (ii) $\overline{\phi^*_{Best}(N)}$ has no TP-handle (Lemma 3.2); and (iii) $(\phi^*_{Best}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$ (Lemma 3.3).

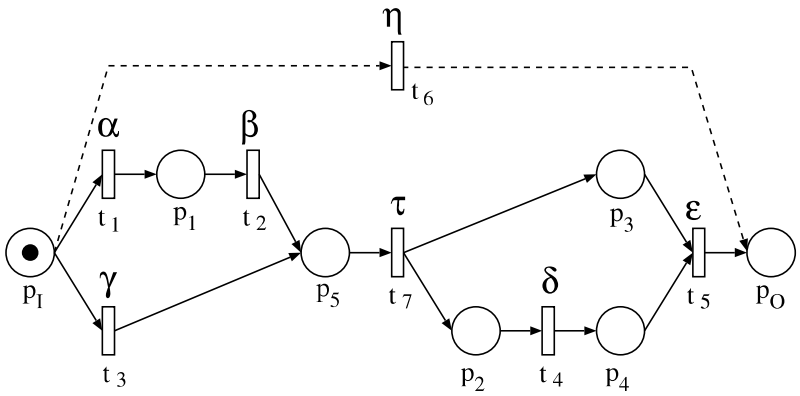
The definition of ϕ_{Best} is given as follows.

Definition 3.1 (The rule ϕ_{Best}) Let N and N' be EFC nets, where N=(P, T, A) and N'=(P', T', A'). The rule ϕ_{Best} can transform N into N' if there exists n places $p_1, p_2, \cdots, p_n$ and m transitions $t_1, t_2, \cdots, t_m$ such that:
Condition on N:

(i) ${p_1}^{\bullet}{}^{N} = {p_2}^{\bullet}{}^{N} = \cdots = {p_n}^{\bullet}{}^{N} = \{t_1, t_2, \cdots, t_m\}$



(a) Acyclic EFC WF-net N_2



(b) Acyclic WS WF-net N_2'

Figure 3.3: Another instance of refactorizability problem.

Construction of N':

(i) $P' = P \cup \{p\}$

(ii) $T' = T \cup \{t\}$

(iii) $A' = A - \{(p_i, t_j) | i=1,2,\cdots,n,\ j=1,2,\cdots,m\}$
$\cup \{(p_i, t), (t, p), (p, t_j) | i=1,2,\cdots,n,\ j=1,2,\cdots,m\}$

□

Property 3.2 For an EFC WF-net N, $\phi_{Best}(N)$ is sound iff N is sound. □

Proof : Any WF-net is sound iff its short-circuited net with initial marking $[p_I]$ is live and bounded (Theorem 1 of Ref. [6]). N is sound iff $(\overline{N}, [p_I])$ is live and bounded. ϕ_{Best} preserves

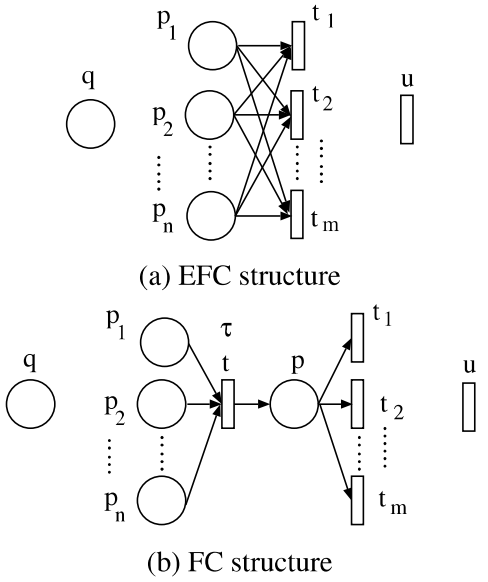


Figure 3.4: Best's transformation rule ϕ_{Best}.

liveness and boundedness (Theorems 3.2, 5.2 and 5.3 of Ref. [13]). $(\overline{N}, [p_I])$ is live and bounded iff $(\overline{\phi_{Best}(N)}, [p_I])$ is live and bounded, i.e. $\phi_{Best}(N)$ is sound.

Q.E.D.

This property means that $\phi^*_{Best}(N)$ is an acyclic FC WF-net.

Lemma 3.1 Let N be a sound acyclic EFC WF-net. If $\overline{N}$ has no PT-handle, $\overline{\phi^*_{Best}(N)}$ has no PT-handle. □

Proof : We have to check the number of node disjoint paths from any place to any transition in $\phi_{Best}(N)$. Let q denote a place such that $q \in P - \{p_1, p_2, \cdots, p_n\}$. Let u denote a transition such that $u \in T - \{t_1, t_2, \cdots, t_m\}$ (see Fig. 3.4). Let p and t respectively denote place and transition added to N.

There are four cases. The first case (The place is p_i and the transition is t_j): There is a single node disjoint path $p_i t p t_j$ in $\phi_{Best}(N)$. Therefore the number of node disjoint paths from p_i to t_j equals one. The second case (The place is p_i and the transition is u): If there is a node disjoint path from p_i to u in N, let t_j be an output transition of p on the path, there is a single node disjoint path $p_i t p t_j$ in $\phi_{Best}(N)$. Therefore the number of node disjoint paths from p_i to u equals one. Otherwise there is no node disjoint path from p_i to u in $\phi_{Best}(N)$. The third case (The place is q and the transition is t_j): If there is a node disjoint path from q to t_j in N, let p_i be an input place of t on the node disjoint path, there is a single node disjoint path $p_i t p t_j$ in $\phi_{Best}(N)$. Therefore

the number of node disjoint paths from q to t_j equals one. Otherwise there is no node disjoint path from q to t_j in $\phi_{Best}(N)$. The last case (The place is q and the transition is u): Assume that there exists a node disjoint path ρ from q to u in N. If ρ includes p_i then we can say from the same reason as the second case that the number of node disjoint paths from p_i to t equals one in $\phi_{Best}(N)$. Therefore the number of node disjoint paths from q to u equals one. Otherwise the number of node disjoint paths from q to u is not changed by ϕ_{Best}, i.e. equals one. If there is no node disjoint path from q to u in N, the number of node disjoint paths from q to u equals zero in $\phi_{Best}(N)$. In all the cases, the number of node disjoint paths from a place to a transition is at most one. This means that $\phi_{Best}(N)$ has no PT-handle. Thus $\phi^*_{Best}(N)$ has no PT-handle. **Q.E.D.**

Lemma 3.2 Let N be a sound acyclic EFC WF-net. If $\overline{N}$ has no PT-handle, $\overline{\phi^*_{Best}(N)}$ has no TP-handle. □

Proof : Since $\phi^*_{Best}(N)$ is a sound FC WF-net, it has no TP-handle from Condition (i) of Property 1. **Q.E.D.**

Lemmas 1 and 2 mean that $\phi^*_{Best}(N)$ is a WS WF-net.

Lemma 3.3 Let N be a sound acyclic EFC WF-net. If $\overline{N}$ has no PT-handle, $(\phi^*_{Best}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$. □

Proof : Assume that there is a binary relation $\mathcal{R} \subseteq R(N, [p_I]) \times R(\phi_{Best}(N), p_I)$ such that

$$\mathcal{R} = \{(M_N, M_{\phi_{Best}(N)}) | \underbrace{M_N = M_{\phi_{Best}(N)} - [p] + {}^{\phi_{Best}(N)}\bullet\, t}_{\text{if } p \in M_{\phi_{Best}(N)}} \text{ or } \underbrace{M_N = M_{\phi_{Best}(N)}}_{\text{otherwise}}\}$$

We divide this proof into two cases: (i) Before t_i fires; (ii) After t_i fired. Case (i) (Before t_i fires): Let $M \in R(N, [p_I])$ until t_i becomes firable. M is also a marking of $\phi_{Best}(N)$. Assume that $M[N, u\rangle M'$. Since $\phi_{Best}(N)$ has the same structure as N except for t and p, we have $M[\phi_{Best}(N), u\rangle M'$. Therefore, we have $M' \mathcal{R} M'$. Let $M \in R(\phi_{Best}(N), [p_I])$ when t is firable. M is also a marking of N. Assume that $M[\phi_{Best}(N), t\rangle M' = M + [p] - {}^{\phi_{Best}(N)}\bullet\, t$. Since t is an unobservable transition, i.e. $M[\phi_{Best}(N), \tau\rangle M'$, we have $M[N, (\tau)\rangle M$. Since $M = M' - [p] + {}^{\phi(N)}\bullet\, t$, we have $M \mathcal{R} M'$. Case (ii) (After t_i fired): Let $M \in R(N, [p_I])$ when t_i is firable. $M + [p] - {}^{\phi_{Best}(N)}\bullet\, t$ is a marking of $\phi_{Best}(N)$. Assume that
$M[N, t_i\rangle M' = M - {}^{N}\bullet t_i + t_i {}^{N}\bullet$. In $\phi_{Best}(N)$, $M + [p] - {}^{\phi_{Best}(N)}\bullet\, t [\phi_{Best}(N), t_i\rangle M - {}^{\phi_{Best}(N)}\bullet\, t + t_i\, {}^{\phi_{Best}(N)}\bullet$.
Since ${}^{N}\bullet t_i = {}^{\phi_{Best}(N)}\bullet\, t$ and $t_i {}^{N}\bullet = t_i\, {}^{\phi_{Best}(N)}\bullet$, we have $M' \mathcal{R} M'$. Let $M \in R(N, [p_I])$ after t_i fired. M is also a marking of $\phi_{Best}(N)$. Assume that $M[N, u\rangle M'$. Since $\phi_{Best}(N)$ has the same structure as N except for t and p, we have $M[\phi_{Best}(N), u\rangle M'$. Therefore, we have $M' \mathcal{R} M'$.

It is obvious that $[p_O]\mathcal{R}[p_O]$ holds. From the above, $\mathcal{R}$ is branching bisimulation, i.e. $(\phi_{Best}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$. Since branching bisimulation is transitive, we have $(\phi^*_{Best}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$. **Q.E.D.**

Now, we can prove Theorem 3.1 by these three lemmas.

Proof of Theorem 3.1: We only have to prove that $\phi^*_{Best}(N)$ is WS and $(\phi^*_{Best}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$. $\phi^*_{Best}(N)$ is WS by Lemmas 3.1 and 3.2. $\phi^*_{Best}(N)$ satisfies branching bisimilarity by Lemma 3.3. **Q.E.D.**

Case of no TP-Handle

We propose a sufficient condition for acyclic EFC WF-nets with no TP-handle.

Theorem 3.2 Let N_X be a sound acyclic EFC WF-net whose every label in N_X except τ is unique. If $\overline{N_X}$ has no TP-handle and each PT-handle has a TP-bridge with length one, then there is an acyclic WS WF-net $N_Y = (P_Y, T_Y, A_Y, \ell_Y)$ such that (i) $(N_X, [p^X_I]) \sim_b (N_Y, [p^Y_I])$, and (ii) every label in N_Y except τ is unique. □

If $\overline{N}$ has no TP-handle then it has PT-handles. To remove the PT-handles, we can apply a transformation rule, denoted by ϕ_{Desel}, which is proposed by Desel [15]. The definition of ϕ_{Desel} is given later. Using rule ϕ_{Desel} repeatedly, we can remove the dual of EFC structure, called TP-cross structure, and obtain an acyclic FC WF-net $\phi^*_{Desel}(N)$ from N. In order to prove this theorem, we show the following: (i) $\phi^*_{Desel}(N)$ has no PT-handle (Lemma 3.4); (ii) $\phi^*_{Desel}(N)$ has no TP-handle (Lemma 3.5); and (iii) $(\phi^*_{Desel}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$ (Lemma 3.6).

Definition 3.2 (The rule ϕ_{Desel}) Let N and N' be EFC nets, where $N=(P, T, A)$ and $N'=(P', T', A')$. The rule ϕ_{Desel} can transform N into N' if there exist m transitions $t_1, t_2, \cdots, t_m$ and n places $p_1, p_2, \cdots, p_n$ such that:
Condition on N:

(i) ${t_1}^{N}\bullet = {t_2}^{N}\bullet = \cdots = {t_m}^{N}\bullet = \{p_1, p_2, \cdots, p_n\}$

Construction of N':

(i) $P' = P \cup \{p\}$

(ii) $T' = T \cup \{t\}$

(iii) $A' = A - \{(t_i, p_j) | i=1,2,\cdots,m,\ j=1,2,\cdots,n\}$
$\cup \{(t_i, p), (p, t), (t, p_j) | i=1,2,\cdots,m,\ j=1,2,\cdots,n\}$

□

Property 3.3 For an acyclic EFC WF-net N, $\phi_{Desel}(N)$ is sound iff N is sound. □

Proof : Any acyclic EFC WF-net is sound iff its short-circuited net is well-formed. N is sound iff $\overline{N}$ is well-formed. ϕ_{Desel} preserves well-formedness (Theorem 7.1 of Ref. [15]). $\overline{N}$ is well-formed iff $\overline{\phi_{Desel}(N)}$ is well-formed, i.e. $\phi_{Desel}(N)$ is sound. **Q.E.D.**

Property 3.4 For any sound acyclic EFC and non-FC WF-net N, there exists a TP-handle in $\overline{N}$
□

Proof : N is illustrated in the upper part of Fig. 3.5. N includes an EFC structure, which is shown in the dotted box. Since N is an acyclic WF-net, there exist three paths ρ_1 (from p_I to a node x), ρ_2 (from x to p_1), and ρ_3 (from x to p_2) such that ρ_2 and ρ_3 share only x. Similarly, there exist three paths ρ_4 (from t_1 to a node y), ρ_5 (from t_2 to y), and ρ_6 (from y to p_O) such that ρ_4 and ρ_5 share only y. The lower part of Fig. 3.5 shows $\phi^*_{Best}(N)$. Note that ϕ_{Best} preserves soundness, and $\phi^*_{Best}(N)$ has the same structure as N except for the part within the dotted box. As node types for x and y, we have four cases: (1) x is a transition and y is a place; (2) x and y are places; (3) x and y are transitions; and (4) x is a place and y is a transition.

Case 1: There is a TP-handle from transition x to place y in $\overline{N}$. Case 2: There is a PT-handle from place x to transition t in $\overline{\phi^*_{Best}(N)}$. Since $\phi^*_{Best}(N)$ is an acyclic sound FC WF-net, the PT-handle has a TP-bridge b_1. b_1 is also in $\overline{N}$. There is a TP-handle from the start node of b_1 to place y in $\overline{N}$. Case 3: There is a PT-handle from place p to transition y in $\overline{\phi^*_{Best}(N)}$. Since $\phi^*_{Best}(N)$ is an acyclic sound FC WF-net, the PT-handle has a TP-bridge b_2. b_2 is also in $\overline{N}$. There is a TP-handle from transition x to the end node of b_2 in $\overline{N}$. Case 4: For the same reason as Cases 2 and 3, there exist TP-bridges b_1 and b_2. There is a TP-handle from the start node of b_1 to the end node of b_2 in $\overline{N}$. Therefore $\overline{N}$ includes a TP-handle in any case. **Q.E.D.**

This property means that any sound acyclic EFC WF-net with no TP-handle is a sound acyclic FC WF-net. ϕ_{Desel} makes no new EFC structure, so $\phi^*_{Desel}(N)$ is also a sound acyclic FC WF-net. In a similar way to Lemmas 3.1 and 3.2, we can obtain the following lemmas.

Lemma 3.4 Let N be an acyclic EFC WF-net. If $\overline{N}$ has no TP-handle and each PT-handle has a TP-bridge with length one, then $\overline{\phi^*_{Desel}(N)}$ has no PT-handle. □

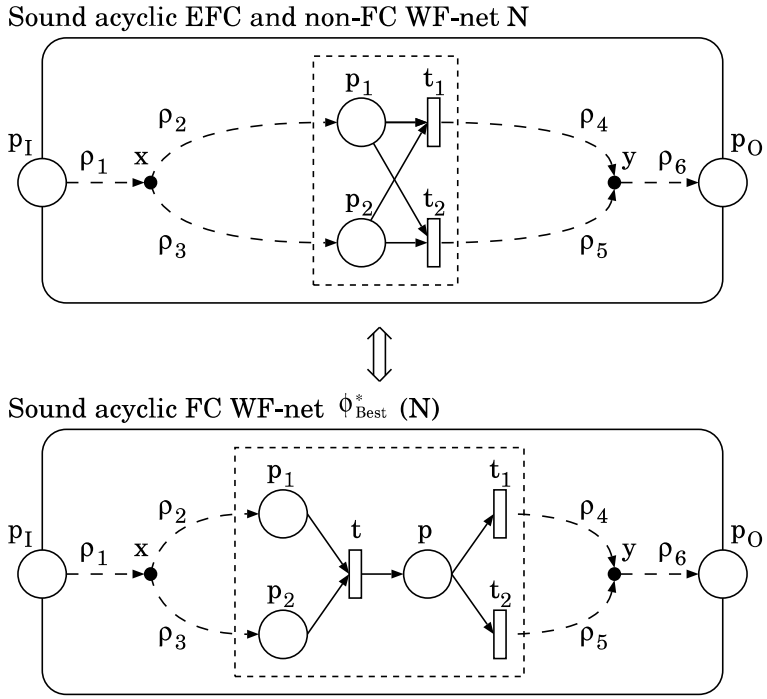


Figure 3.5: Illustration of the proof of Property 3.4.

Proof : We can prove it in a similar way to Lemma 3.1.

Q.E.D.

Lemma 3.5 Let N be an acyclic EFC WF-net. If $\overline{N}$ has no TP-handle and each PT-handle has a TP-bridge with length one, then $\overline{\phi^*_{Desel}(N)}$ has no TP-handle. □

Proof : Since $\phi^*_{Desel}(N)$ is a sound FC WF-net, it has no TP-handle from Condition (i) of Property 2.1.

Q.E.D.

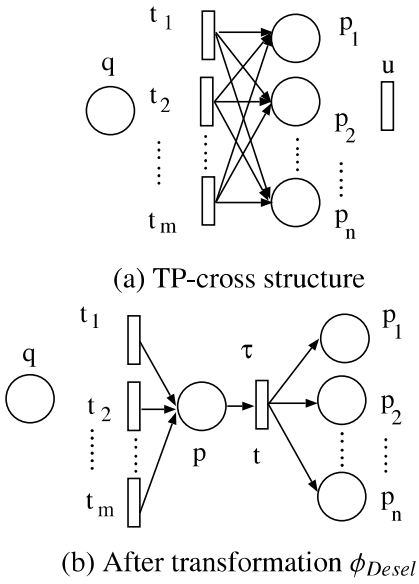
Lemmas 3.4 and 3.5 mean that $\phi^*_{Desel}(N)$ is a WS WF-net.

In a similar way to Lemma 3.3, we can obtain the following lemma.

Lemma 3.6 Let N be an acyclic EFC WF-net, If $\overline{N}$ has no TP-handle and each PT-handle has a TP-bridge with length one, then $(\phi^*_{Desel}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$.

□

Now, we can prove Theorem 3.2 by these three lemmas.



(a) TP-cross structure

(b) After transformation ϕ_{Desel}

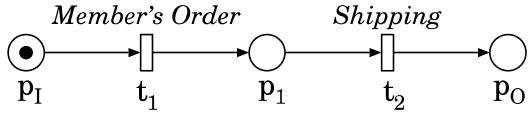
Figure 3.6: Desel's Transformation rule ϕ_{Desel}.

Proof of Theorem 3.2: We have to prove that $\phi^*_{Desel}(N)$ is WS and $(\phi^*_{Desel}(N), [p_I])$ is branching bisimilar with $(N, [p_I])$. $\phi^*_{Desel}(N)$ is WS by Lemmas 3.4 and 3.5. $\phi^*_{Desel}(N)$ satisfies branching bisimilarity by Lemma 3.6. **Q.E.D.**

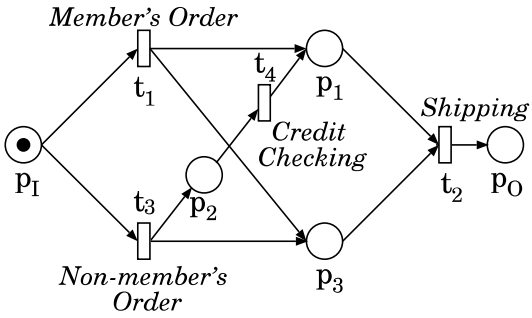
3.3 Sufficient Condition based on Implicit Place

A WF-net is often updated to adapt to the change of business environment. This may cause places that are redundant from the viewpoint of the behavior. Such places are called implicit [8]. Figure 3.7 (a) shows the WF-net representing an exclusive ordering system. One day, the system was required to deal not only with members' order but also non-members' order. Since non-members' order needs credit check every time, designers change N_1 to another WF-net N_2 shown in Fig. 3.7 (b). N_2 has an implicit place p_3. Since the implicit place has no effects to WF-net's behavior, designers should remove the place to simplify the WF-net.

Let us consider FC WF-net N_2 shown in Fig. 3.7 (b) again. We cannot decide the refactorizability by using only the existing conditions. On the other hand, by removing implicit place p_3 from N_2, N_2 will become an acyclic WS WF-net N_3. We expect that removing of implicit places is a new approach to WF-net refactoring.



(a) A WF-net N_1 representing an ordering system.



(b) The WF-net N_2 extended from N_1. It has an implicit place p_3.

Figure 3.7: An example of generating implicit places.

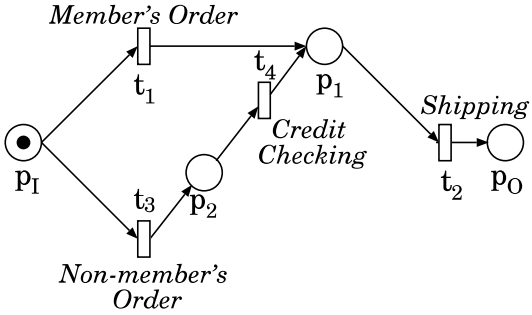


Figure 3.8: The WS WF-net N_3 reafactored from N_2 by removing the implicit place.

Sufficient Condition based on Removing Implicit Place

We propose a sufficient condition on the refactorizability problem based on implicit places. We first show that removing of implicit places is a reduction operation which forms branching bisimilarity.

Lemma 3.7 For a sound acyclic EFC WF-net N and the WF-net $N'(=(P\backslash\{p\}, T, A\backslash({}^{N}\bullet p\times\{p\})\backslash(\{p\}\times p^{N}\bullet), \ell))$ obtained by removing an implicit place p from N, $(N, [p_I])$ is branching bisimilar with $(N', [p_I])$. **Q.E.D.**

Proof: Assume the following binary relations $\mathcal{R}$ between N $R(N, [p_I])$ and $R(N', [p_I])$:

$$\mathcal{R} \stackrel{def}{=} \{(M, M') | \underbrace{M{=}M' \cup [p]}_{\text{if } M \geq [p]} \text{ or } \underbrace{M{=}M'}_{\text{otherwise}} \} \tag{3.1}$$

This relation $\mathcal{R}$ is a branching bisimulaion which is shown in the proof of Lemma 4 in Ref. [19]. **Q.E.D.**

Theorem 3.3 For a sound acyclic EFC WF-net N and the WF-net N' obtained by removing implicit places from N, if N' is a WS WF-net, then N is refactorable to an acyclic WS WF-net (N' is a result of refactoring). □

Proof: By Lemma 3.7, removing an implicit place p from N preserves branching bisimilarity. Therefore, N is branching bisimilar with N'. Since N' is the WS WF-net, the result of the refactoring. **Q.E.D.**

Examples of WF-Net Refactoring by Removing Implicit Places

Let us first consider a sound acyclic FC WF-net N_4 shown in Fig. 3.9 (a). N_4 is the same structure as N_2 we discussed in Sect. 1. There is an implicit place p_3 in $(N_4, [p_I])$ because by Theorem 4.1, $t_1 p_3 t_2$ and $t_3 p_3 t_2$ are TT-handles in $\overline{N_4}$. We can refactor N_4. We removes p_1 from N_4, we obtain a WF-net N_5 shown in Fig. 3.9 (b). N_5 is the same as N_3. N_5 is an acyclic WS WF-net. Since there exists a branching bisimulation relation between $(N_4, [p_I])$ and $(N_5, [p_I])$ as shown in Fig. 3.9 (c). In other words, N_5 is a result of refactoring of N_4. Note that we cannot decide refactorizability of N_4 by Property 1. Because N_4 has no TP-handle but PT-handle $p_I t_3 p_3 t_2$ has a TP-bridge $t_3 p_2 t_4 p_1$ with length three. Removing implicit places enables us to refactor EFC WF-nets which we cannot deal with by the former conditions.

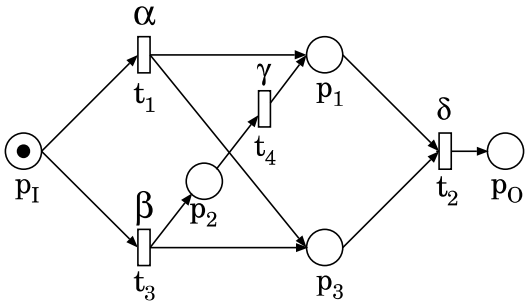
Although removing implicit places broadens out the set of refactorable EFC WF-nets, the resultant WF-net is not necessary WS. Let us consider a sound acyclic EFC WF-net N_6 shown in Fig. 3.10 (a). There is an implicit place p_5 in the $(N_6, [p_I])$. We removes p_5 from N_6, we can obtain N_7. There is a PT-handle $p_I t_3 p_2 t_5$ in $\overline{N_7}$, so N_7 is not WS. That is to say, we cannot decide the refactorizability by using Theorem 3.3. Furthermore, N_7 has a PT-handle $p_I t_3 p_2 t_5$ which has a TP-bridge $t_3 p_3 t_4 p_4$ with length three. Thus we also cannot decide the refactorizability from Teorem 3.1 and 3.2.

Role of Implicit Place in WF-net Refactoring

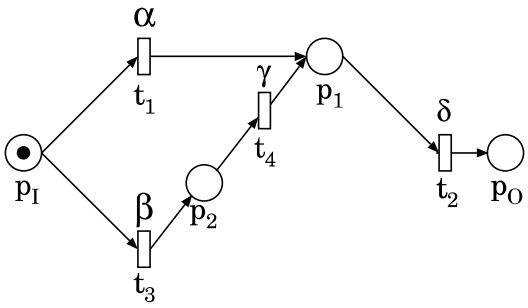
Removing implicit places can be regarded as preprocessing for the other refactoring operations. It enables us to construct more general refactoring algorithms.

Let us consider the FC WF-net N_{12} shown in Fig. 3.11 (a), representing another order system. There is one implicit place p_4. p_4 is a part of PT-handle has no TP-bridge. It is "obstacle" for applying Teorem 3.2. We removes p_4. Figure 3.11 (b) shows the result net N_{13}. From Lemma 3.7, there is a branching bisimulaion between N_{12} and N_{13}. As a consequence the PT-handle is removed. N_{13} satisfies Teorem 3.2. N_{14} shown in Fig. 3.11 (c) is refactored from N_{13} by applying ϕ_{Desel}.

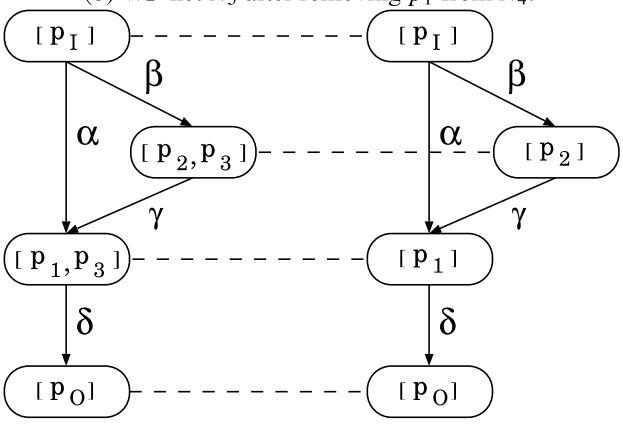
Since branching bisimulaion is transitive, there is a branching bisimulaion between N_{12} and N_{14}. It means that N_{14} is refactored from N_{12}.



(a) A sound acyclic FC WF-net N_4.

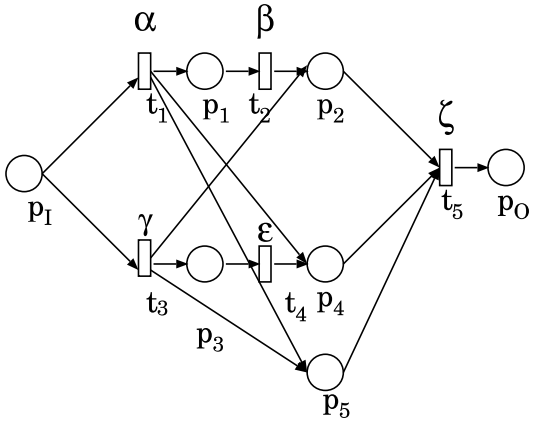


(b) WF-net N_5 after removing p_1 from N_4.

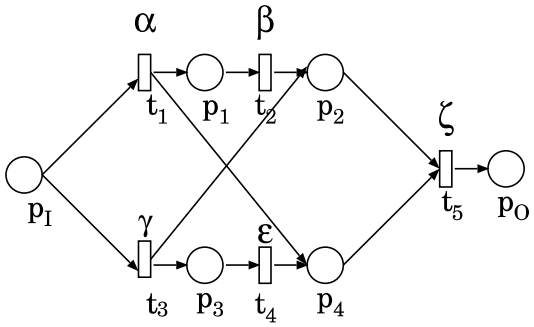


(c) Branching bisimulation relation between $(N_4, [p_I])$ and $(N_5, [p_I])$.

Figure 3.9: An example of refactoring by removing an implicit place.

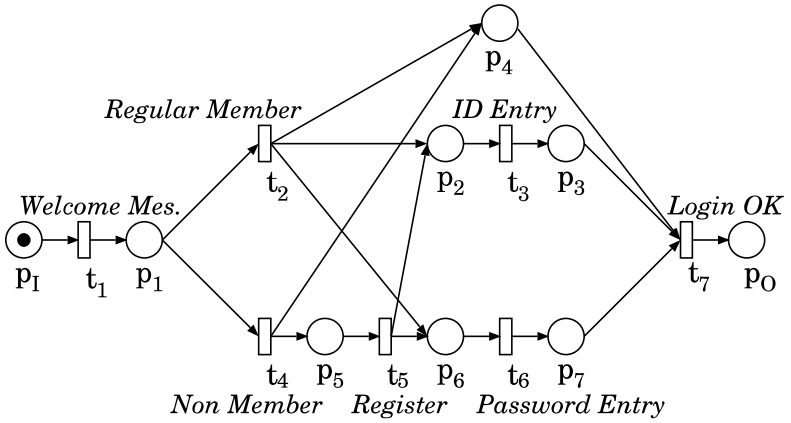


(a) A sound acyclic EFC WF-net N_6.

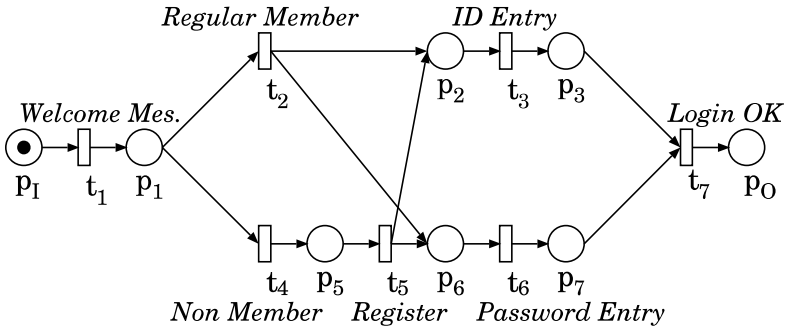


(b) WF-net N_7 which is obtained by removing p_5 from N_6.

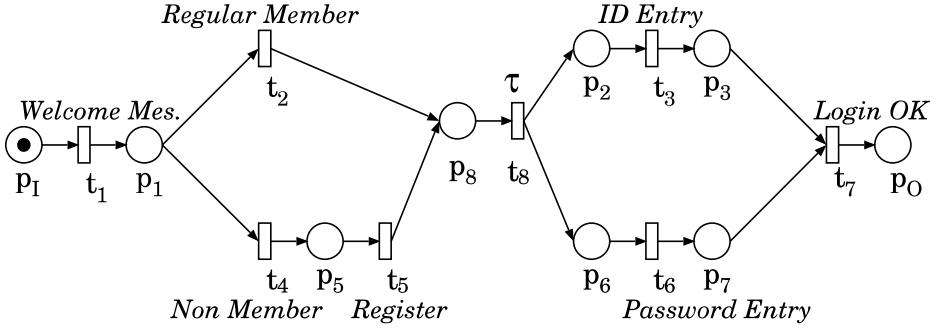
Figure 3.10: An instance of the refactorizability problem. This cannot be solved by removing and implicit places and using ϕ_{Best} and ϕ_{Desel}.



(a) A WF-net N_{12} which expresses an another order system



(b) A FC WF-net N_{13} which refactored by removing implicit placep_1



(c) A WS WF-net N_{14} which is refactored from N_{13} by applying ϕ_{Desel}

Figure 3.11: An example of applying two refactoring methods.

3.4 Remarks

In this chapter, we proposed three sufficient conditions on refactorizability.

- A sound acyclic EFC WF-net N is refactorable to an acyclic WS WF-net if $\overline{N}$ has no PT-handle.
- A sound acyclic EFC WF-net N is refactorable to an acyclic WS WF-net if $\overline{N}$ has no TP-handles and every PT-handle in $\overline{N}$ has a TP-bridge with length one.
- A sound acyclic EFC WF-net N and the WF-net N' obtained by removing implicit places from N, if N' is a WS WF-net, then N is refactorable to an acyclic WS WF-net (N' is a result of refactoring).

In the next chapter, we will see that we can also decide these conditions in polynomial time, because the conditions are whether a given WF-net has TP-handle and/ or PT-handle and/or implicit place.

Chapter 4

Refactoring Algorithm

4.1 Background and Need

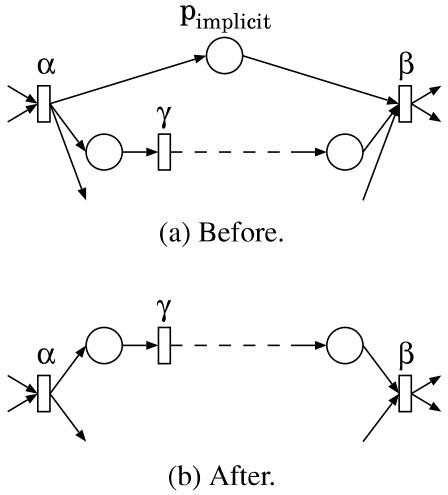
Once we model a workflow as a WF-net, we can simulate the behavior of the workflow by using tokens on the WF-net. We can also use WF-nets' abundant techniques to solve many problems associated with the modeled workflow. It is known that most actual workflows can be modeled as EFC WF-nets; acyclic WS is a subclass of acyclic EFC but has more analysis methods, e.g. polynomial time algorithm on the reachability problem [32]. If an acyclic EFC WF-net is refactored to an acyclic WS WF-net, we can use the analysis methods of WS WF-nets to analyze the EFC WF-net.

In Chapter 3, we have proposed three refactoring rules. The first two rules are based on the structure of the net. The last rule is based on implicit places. The previous work has dealt with refactoring rules separately. If we use them together, we could further expand the possibility of refactoring.

In this chapter, we analyze the relations among the three refactoring rules, and propose a refactoring algorithm for applying those rules based on the analysis result. In Sect. 4.2, we discuss the three refactoring rules and their relations between them. In Sect. 4.3, we give the refactoring algorithm. We give remarks in Sect. 4.4.

4.2 Three Refactoring Rules and their Conditions

We first present the three refactoring rules proposed in Chapter 3. These rules have been treated separately up to now. To use them together, we reveal the relation among them.



(a) Before.

(b) After.

Figure 4.1: Refactoring rule $\phi_{implicit}$.

4.2.1 Refactoring Rules

Removing implicit place

The first refactoring rule named $\phi_{implicit}$ (See Fig. 4.1) is given as follows:

Definition 4.1 (The rule $\phi_{implicit}$) Let N and N' be sound acyclic EFC WF-nets, where N=(P, T, A) and N'=(P', T', A'). The rule $\phi_{implicit}$ can transform N into N' if there exists an implicit place $p_{implicit}$ such that

Construction of N':

(i) $P' = P \backslash \{p_{implicit}\}$

(ii) $T' = T$

(iii) $A' = A \setminus ({}^{N}\bullet p_{implicit} \times \{p_{implicit}\}) \cup (\{p_{implicit}\} \times p_{implicit}{}^{N}\bullet)$

□

A sound acyclic EFC WF-net can be refactored by $\phi_{implicit}$ under the following properties [35].

First, for a sound acyclic EFC WF-net N and the WF-net $N'(=(P \setminus \{p\}, T, A \backslash ({}^{N}\bullet p \times \{p\}) \backslash (\{p\} \times p{}^{N}\bullet), \ell))$ obtained by removing implicit place p, $(N, [p_I])$ is branching bisimilar with $(N', [p'_I])$. □

Second, for a sound acyclic EFC WF-net N and the WF-net N' obtained by removing implicit places from N, if N' is a WS WF-net, then N is refactorable to an acyclic WS WF-net (N' is a result of refactoring). □

Let us consider a sound acyclic EFC WF-net N_1 shown in Fig. 3.9 (a). There is an implicit place p_1 in N_1. Removing p_1 from N_1, we can obtain N_2 shown in Fig. 3.9 (b). Since N_2 is an acyclic WS WF-net, from lemma 3.7, N_1 is refactorable.

Removing EFC structures

The second refactoring rule is called ϕ_{Best} as bellows. Let N and N' be EFC WF-nets, where N=(P, T, A) and N'=(P', T', A'). The rule ϕ_{Best} can transform N into N' if there exist n places $p_1, p_2, \cdots, p_n$ and m transitions $t_1, t_2, \cdots, t_m$ such that:
Condition on N:

(i) $p_1{}^{N}\bullet = p_2{}^{N}\bullet = \cdots = p_n{}^{N}\bullet = \{t_1, t_2, \cdots, t_m\}$

Construction of N':

(i) $P' = P \cup \{p\}$

(ii) $T' = T \cup \{t\}$

(iii) $A' = A \backslash \{(p_i, t_j) | i{=}1, 2, \cdots, n, j{=}1, 2, \cdots, m\}$
$\cup \{(p_i, t), (t, p), (p, t_j) | i{=}1, 2, \cdots, n, j{=}1, 2, \cdots, m\}$

□

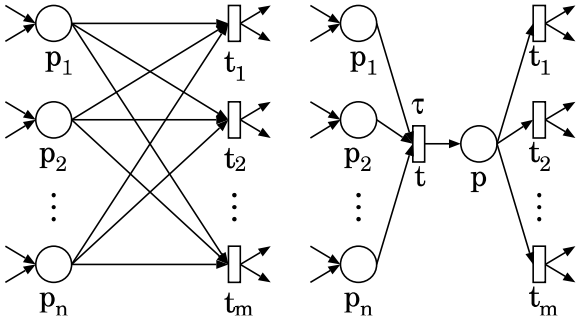
An acyclic EFC WF-net is refactorable to an acyclic WS WF-net by ϕ_{Best} under the following : Let N_X be a sound acyclic EFC WF-net, whose every label in N_X except τ is unique. If $\overline{N_X}$ has no PT-handle, then there is an acyclic WS WF-net N_Y such that (i) $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$, and (ii) every label in N_Y except τ is unique.

This property means that a sound acyclic EFC WF-net N is refactorable to an acyclic WS WF-net if $\overline{N}$ has no PT-handle. If $\overline{N}$ has no PT-handle then it has TP-handles. To remove the TP-handles, we can apply ϕ_{Best}. ϕ_{Best} can transform an EFC structure to its equivalent FC structure [16]. Using rule ϕ_{Best} repeatedly, we can obtain an acyclic FC WF-net $\phi^*_{Best}(N)$ from N.

Removing TP-cross structures

The last refactoring rule is called ϕ_{Desel} as bellows.

Let N and N' be EFC WF-nets, where N=(P, T, A) and N'=(P', T', A'). The rule ϕ_{Desel} can transform N into N' if there exist m transitions $t_1, t_2, \cdots, t_m$ and n places $p_1, p_2, \cdots, p_n$ such that:
Condition on N:



(a) Before. (b) After.

Figure 4.2: Refactoring rule ϕ_{Best}.

(i) ${}^{N}t_1\bullet = {}^{N}t_2\bullet = \cdots = {}^{N}t_m\bullet = \{p_1, p_2, \cdots, p_n\}$

Construction of N':

(i) $P' = P \cup \{p\}$

(ii) $T' = T \cup \{t\}$

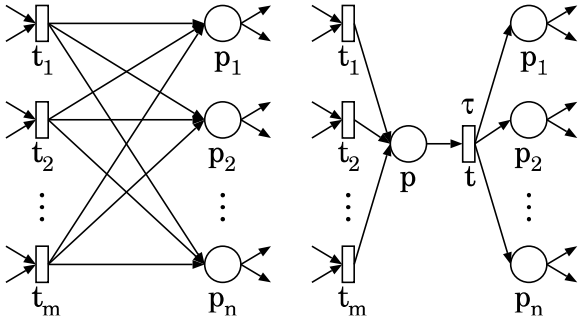
(iii) $A' = A \backslash \{(t_i, p_j) | i{=}1, 2, \cdots, m, j{=}1, 2, \cdots, n\}$
$\cup \{(t_i, p), (p, t), (t, p_j) | i{=}1, 2, \cdots, m, j{=}1, 2, \cdots, n\}$

An acyclic EFC WF-net can be refactored by ϕ_{Desel} under the following : Let N_X be a sound acyclic EFC WF-net whose every label in N_X except τ is unique. If $\overline{N_X}$ has no TP-handle and each PT-handle has a TP-bridge with length one, then there is an acyclic WS WF-net N_Y such that (i) $(N_X, [p_I^X]) \sim_b (N_Y, [p_I^Y])$, and (ii) every label in N_Y except τ is unique.

If $\overline{N}$ has no TP-handle then it has PT-handles. To remove the PT-handles, we can apply ϕ_{Desel}. Using rule ϕ_{Desel} repeatedly, we can remove the dual of EFC structure, called TP-cross structure, and obtain an acyclic FC WF-net $\phi^*_{Desel}(N)$ from N.

4.2.2 Relations among Refactoring Rules

To use the three refactoring rules together, we need to reveal the relation among them. We first summarize the input and output of those rules into Table. 4.1. Note that we suppose there exists branching bisimulaion between N and N'. $\phi_{implicit}$ does not reduce the class of N necessarily. In other words, $\phi_{implicit}$ is the weakest rule among the three ones. Meanwhile, it has no constraint for the input EFC WF-net. That is the largest class of the input. Next, the input of ϕ_{Desel} is FC WF-nets implicitly. We can obviously obtain the following property.



(a) Before. (b) After.

Figure 4.3: Refactoring rule ϕ_{Desel}.

Table 4.1: Summary of three refactoring rules.

Refactoring rule	Input	Output
Removing implicit place $\phi_{Implicit}$	Sound acyclic EFC WF-net N	Sound acyclic EFC WF-net N'
Removing EFC structure ϕ_{Best}	Sound acyclic EFC WF-net N ($\overline{N}$ has no PT-handle)	Sound acyclic WS WF-net N'
Removing TP-cross structure ϕ_{Desel}	Sound Acyclic EFC WF-net N ($\overline{N}$ has no TP-handle, each PT-handle in $\overline{N}$ has TP-bridge with length one)	Sound acyclic WS WF-net N'

Property 4.1 For any sound acyclic EFC and non-FC WF-net N, there exist TP-handles in $\overline{N}$. □

By the contraposition, a sound acyclic EFC WF-net N is FC if $\overline{N}$ has no TP-handle. FC WF-nets are a proper subclass of EFC WF-nets. So that, WF-nets which can be refactored by removing TP-cross structures may be smaller than that of removing EFC structure. Through the above analysis, we should use the three rules according to the following order.

1° Remove implicit places by $\phi_{Implicit}$

2° Remove EFC structures by ϕ_{Best}

3° Remove TP-cross structures by ϕ_{Desel}

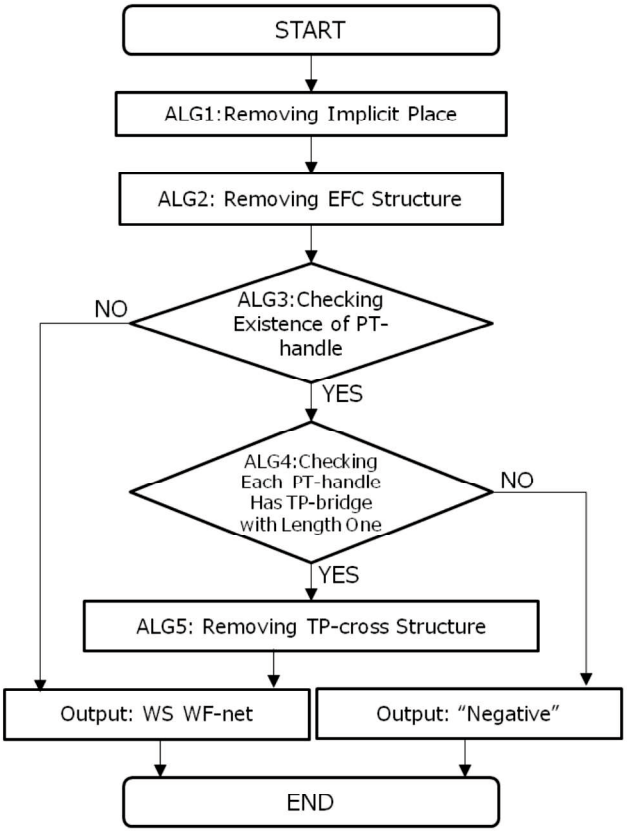


Figure 4.4: Flowchart of "3 gate refactoring".

4.3 Refactoring Algorithm

Based on the analysis result, we construct an algorithm for using the three refactoring rules together. We name it "*3 gate refactoring*" and show its flowchart in Fig. 4.4.

Firstly, to remove implicit places, We propose necessary and sufficient condition to find implicit place.

Theorem 4.1 In a sound acyclic EFC WF-net $N=(P,T,A,\ell)$, a place p is implicit iff for each transition $t_I \in {}^{N}\bullet p$ and each transition $t_O \in {}^{N}\bullet p$, path $t_I p t_O$ is a TT-handle in $\overline{N}$. □

The following property is used to prove Theorem 4.1.

Property 4.2 In a sound acyclic EFC WF-net N, for a place p, if ${}^{N}\bullet({}^{N}p\bullet) = \{p\}$, then place p is not implicit. □

Proof: From the definition of implicit place, p must satisfy $\forall M \in R(N,[p_I]) : M \geq {}^{N}\bullet({}^{N}p\bullet)\backslash\{p\}=\emptyset \Rightarrow$

$M \geq {}^{N}\bullet(p^{N}\bullet) = \{p\}$, However, for $[p_I], [p_O] (\in R(N, [p_I])$, p cannot satisfy at least one of $[p_I] \geq \{p\}$ and $[p_O] \geq \{p\}$. Thus p is not implicit. **Q.E.D.**

Proof of Theorem 4.1: Let us first prove the sufficient condition. Since removing p from N causes none of new PT-handle, TP-handle and conflict structure, the resultant net $N' = (P \setminus \{p\}, T, A \setminus ({}^{N}\bullet p \times \{p\}) \setminus (\{p\} \times p^{N}\bullet), \ell)$ is a sound EFC WF-net. Since the short-circuited net of a sound EFC WF-net is a live and safe Petri net (Theorem 1 and Lemma 1 of Ref. [29]), it can be regarded as an interconnection of strongly-connected MG components [15]. Each strongly-connected MG component can be divided into a cycle, TT-handles and TT-bridges. For each of the strongly connected MG components, we can obtain a MG WF-net by removing t^*. This enables us to analyze the behavior of the given WF-net through the analysis of those MG WF-nets.

Since N and N' are the sound EFC WF-nets, they can be divided into the same set of MG WF-net except p. A MG WF-net of N including p is denoted by MG, and the MG WF-net of N' corresponding to MG is denoted by MG'. $t_I p t_O$ is a TT-handle from the condition. This means that there exists a path from p_I to p_O which does not pass through p in MG. This path is also included in MG'. Since each transition of acyclic MG WF-net fires only once exactly, t_O must fire after t_I. In MG, p is marked immediately after t_I fired and it is marked until t_O fires. Therefore, $\forall M \in R(MG, [p_I]) : M \geq {}^{MG}\bullet t_O \setminus \{p\} \Rightarrow M \geq {}^{MG}\bullet t_O$ holds. This means that p is implicit in MG. Since p is implicit in anyone of the MG WF-nets, p is implicit in N.

Next, let us prove the necessary condition. It can be divided into the following three cases: (i) p exists on a handle with more than length three; (ii) p exists on a bridge;or (iii) p exists neither on handles nor bridges. Case (i): In the similar way as the sufficient condition, we first divide N into MG WF-nets, and then analyze each of them. The length of the handle is three or more, so p is not marked until t_O fires. Therefore, $\exists M \in R(MG, [p_I]) : M \geq {}^{MG}\bullet t_O \setminus \{p\} \Rightarrow M \ngeq {}^{MG}\bullet t_O$ holds. Thus, p is not implicit. Case (ii): This case is similar as Case (i). $t_I p t_O$ is a part of the bridge in $\overline{MG}$, so there exists a path from t_I to t_O which does not pass through p. Since p may be marked after t_I fired, it means that t_O does not always fire after t_I. $\exists M \in R(MG, [p_I]) : M \geq {}^{MG}\bullet t_O \setminus \{p\} \Rightarrow M \ngeq {}^{MG}\bullet t_O$ holds. Therefore, p is not implicit.

Case (iii): p is an articulation point. Since it is neither the start point nor the end point of a handle or a bridge ($|{}^{N}\bullet p| = |p^{N}\bullet| = 1$), from Property 4.2, it is not implicit. **Q.E.D.**

We first propose a polynomial time subroutine to decide whether a given place is implicit based on Theorem 4.1.

$\ll$: Checking of implicit place $\gg$

Input： Sound acyclic EFC WF-net N, place p of N

Output： Is p implicit ?

1° Obtain a Petri net N' by removing p from N.

2° For every pair $(t_I, t_O) \in {}^{N'}\bullet p \times p^{N'}\bullet$, check whether there exists a directed path from t_I to t_O in N'. If no, output no and stop.

3° Output yes and stop.

Now, we can construct below ALG1.

≪ALG1 : Removing implicit places≫

Input： Sound acyclic EFC WF-net $N(=(P, T, A, \ell))$

Output：Sound acyclic EFC WF-net N' without implicit place

1° For each place $p \in P$

Apply ≪ : Checking of implicit place ≫ to p.

If the result is yes then remove p from N.

2° Output N as N' and stop.

As a result, we obtain a sound acyclic EFC WF-net N' with no implicit place. Next, to remove EFC structure, we use ALG2 below.

≪ALG2 : Removing EFC structure≫

Input: Sound acyclic EFC WF-net N

Output: Acyclic WS WF-net N'

Constraints: (i) $(N, [p_I]) \sim_b (N', [p'_I])$; (ii) every label in N' except τ is unique.

1° For each place $p \in P$: $S_p \leftarrow \{p\}$

For each place $q \in P \setminus \{p\}$: if(${}^N\bullet p = {}^N\bullet q$) $S_p \leftarrow S_p \cup \{q\}$

2° For each place $p \in P$:if($p \in P$) apply ϕ_{Best} to the EFC structure constructed by S_p and $S_p{}^N\bullet$

3° Output N and stop.

After applying ALG2, the resultant WF-net has no TP-handle. If $\overline{N}$ has no PT-handle, N is WS. We must check the condition by ALG3 below.

≪ALG3 : Checking Existence of PT-handle≫

Input: Sound acyclic EFC WF-net N

Output: Is there a PT-handle in $\overline{N}$?

1° Construct the flow network D_N (=(V_N, E_N)) whose every edge has capacity 1, where V_N=$P \cup T$, E_N=A.

2° For each vertex pair (v_i, v_j) ($\in V_N \times V_N$), if

- v_i corresponds to a place p of N s.t. $|p^{N}\bullet| \geq 2$;
- v_j corresponds to a transition t of N s.t. $|\bullet^{N} t| \geq 2$; and
- The maximum value of flow between v_i and v_j exceeds 1,

then output yes and stop.

3° Output no and stop.

If the ALG3's output is "No", "3 gate refactoring" outputs N as a WS WF-net and stop, otherwise, apply ALG4. ALG4 checks the pre-conditions of ALG5. If ALG4 outputs "Yes", ALG5 can refactor N. As a consequence, "3 gate refactoring" outputs WS WF-net N, otherwise, outputs "Negative". We show ALG4 and ALG5 below.

≪ALG4 : Checking Each PT-handle Has TP-Bridge with Length One≫

Input: Sound acyclic EFC WF-net N with no TP-handle (Sound acyclic FC WF-net with PT-handles)

Output: Does each PT-bridge have a TP-bridge with length one?

1° For each transition $t \in T$:$S_t \leftarrow \emptyset$

For each transition $u \in T \setminus \{t\}$: if($t^{N}\bullet = u^{N}\bullet$) $S_t \leftarrow S_t \cup \{t, u\}$

2° ▷ Compute a Petri net $\overline{N_X}'$ obtained by removing all TP-bridges with length one from $\overline{N_X}$.

$\overline{N_X}' = (P, T, A \setminus \bigcup_{t \in T}(S_t \times S_t{}^{N}\bullet), \ell)$

3° Construct the flow network D_N (=(V_N, E_N)) whose every edge has capacity one, where V_N=$P \cup T$, E_N=A.

4° For each vertex pair (v_i, v_j) ($\in V_N \times V_N$), if

- v_i corresponds to a place p of N s.t. $|p^{N}\bullet| \geq 2$;
- v_j corresponds to a transition t of N s.t. $|\bullet^{N} t| \geq 2$;

- The maximum value of flow between v_i and v_j exceeds 1; and
- v_j is reachable from v_i in $\overline{N_X}'$,

then output no and stop.

5° Output yes and stop.

Next, we give ALG5. it is also very similar to ALG2.

≪ALG5 : Removing TP-cross structure≫

Input: Sound acyclic EFC WF-net N with no TP-handle and each PT-handle has a TP-bridge with length one.

Output: Acyclic WS WF-net N'

Constraints: (i) $(N, [p_I]) \sim_b (N', [p'_I])$; (ii) every label in N' except τ is unique.

1° For each transition $t \in T$

$S_t \leftarrow \{t\}$

For each transition $u \in T - \{t\}$

if($t^{\bullet}_{N} = u^{\bullet}_{N}$)

$S_t \leftarrow S_t \cup \{u\}$

2° For each place $t \in T$

if($t \in T$)

Apply ϕ_{Desel} to the EFC structure constructed by S_t and $S_t{}^{\bullet}_{N}$

3° Output N and stop.

Finally, combining those algorithms, we give a refactoring algorithm named “3 gate refactoring”.

≪3 gate refactoring≫

Input: Sound acyclic EFC WF-net N

Output: Acyclic WS WF-net N'

1° Apply ALG1 to N

2° Apply ALG2 to N

3° Apply ALG3 to N'. If the result is “No”, output N' and stop.

4° Apply ALG4 to N'. If the result is “No”, output “Negative” and stop.

5° Apply ALG5 to N'

6° Output N' and stop

4.4 Remarks

In this chapter, we proposed a refactoring algorithm for EFC WF-net to WS WF-net, by combining the three refactoring rules.

As a future work, we refine and validate the algorithm and estimate its computational complexity.

Chapter 5

Application Examples

5.1 Reachability Checking : Case 1

The proposed refactoring algorithm helps us to solve some problems of WF-nets. For example,

- WS WF-nets have simpler structure than EFC WF-nets. That helps us to understand the behavior of the net.
- Some analysis methods can be applied to WS WF-nets, which can not be applied to EFC WF-nets. They enable us to verify the specification of the net more precisely.

We show an example of website workflows. This example represents the above merits. In this chapter, we try to solve the reachability problem for EFC WF-net with many workflow instances. We know the reachability problem can be solve polynomial-time if the WF-net has only one workflow instance. But, it is unclear whether the problem can be solved in polynomial time if there exist two or more instances in the same WF-net. This seems intractable problem.

Market growth or law revision requires us to change a workflow definition even if there exist workflow instances. Such a situation is called dynamic workflow change. We must appropriately move all the instances in the old workflow definition to the new one. If not, "dynamic bugs" would occur. For example, some instance is lost or becomes impossible to terminate normally. In terms of WF-nets, dynamic workflow change is to replace a WF-net (N_{old}, M_{old}) by another WF-net (N_{new}, M_{new}). If M_{old} is reachable from $[(p_I^{old})^k]$ in N_{old}, i.e. $[(p_I^{old})^k][N_{old}, *\rangle M_{old}$, then M_{new} must be reachable from $[(p_I^{new})^k]$ in N_{new}, i.e. $[(p_I^{new})^k][N_{new}, *\rangle M_{new}$. This is why reachability is very important to verify the correctness of workflow instances.

Let us consider an EFC WF-net N_{ex1} shown in Fig. 5.6. Assume that the initial marking of N_{ex1} is $[(p_I^1)^2]$. This means that there are two workflow instances. Let N_{ex1} has a marking $M_{ex1} = [p_1^1, p_2^1, p_8^1]$.

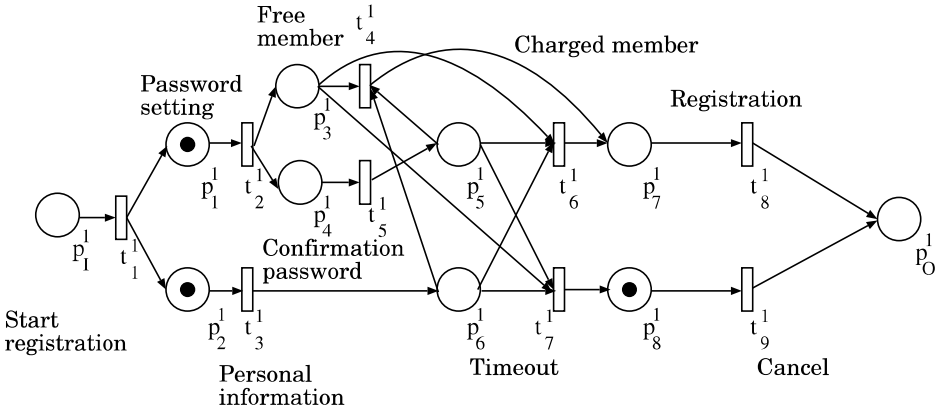


Figure 5.1: Old WF-net N_{ex1}.

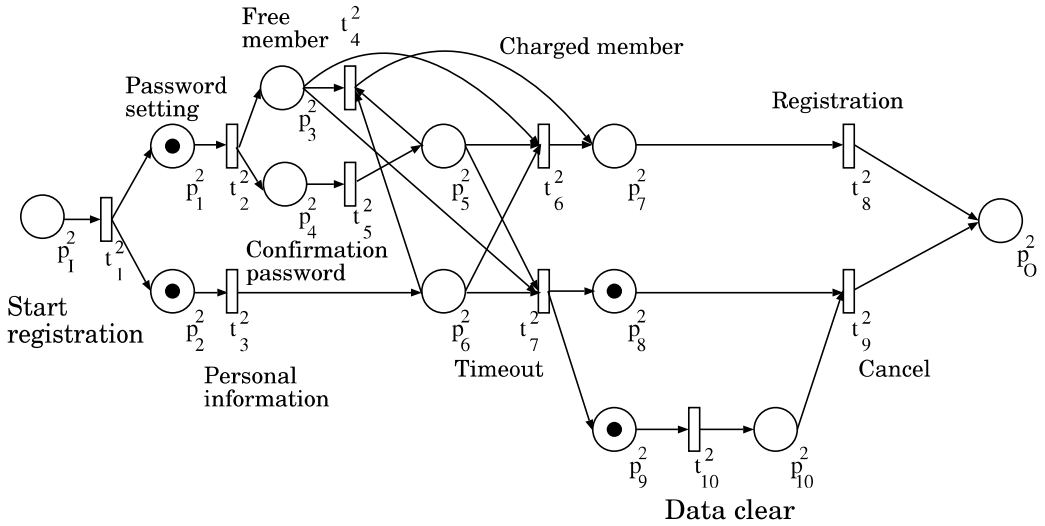


Figure 5.2: New WF-net N_{ex2} which is obtained by extending N_{ex1}.

Let us change N_{ex1} to another EFC WF-net N_{ex2} shown in Fig. 5.2. N_{ex2} has a transition t_{10} labeled as Data clear and new places p_9 and p_{10} are also added. This change is performed in runtime. All the workflow instances in N_{ex1} must be miraged to N_{ex2}. Assume that M_{ex1} is migrated to a marking $M_{ex2} = [p_1^2, p_2^2, p_8^2, p_9^2]$ of N_{ex2} (See Fig. 5.3). Does M_{ex2} represent correct workflow instances? We have to check whether M_{ex2} is reachable from $[(p_I^2)^2]$. Unfortunately, since N_{ex2} has two workflow instances, we cannot apply Yamaguchi's reachability analysis method [32] to this problem. So we try to refactor N_{ex2} to a WS WF-net and then apply Yamaguchi's method to the WS WF-net. We first apply algorithm ALG3 to N_{ex2}. Step 1° generates the flow network $D_{N_{ex2}}$ shown in Fig. 5.4. Step 2° checks maximum flow from any place to any transition. All the values of maximum flow are at most 1. Thus this algorithm outputs No.

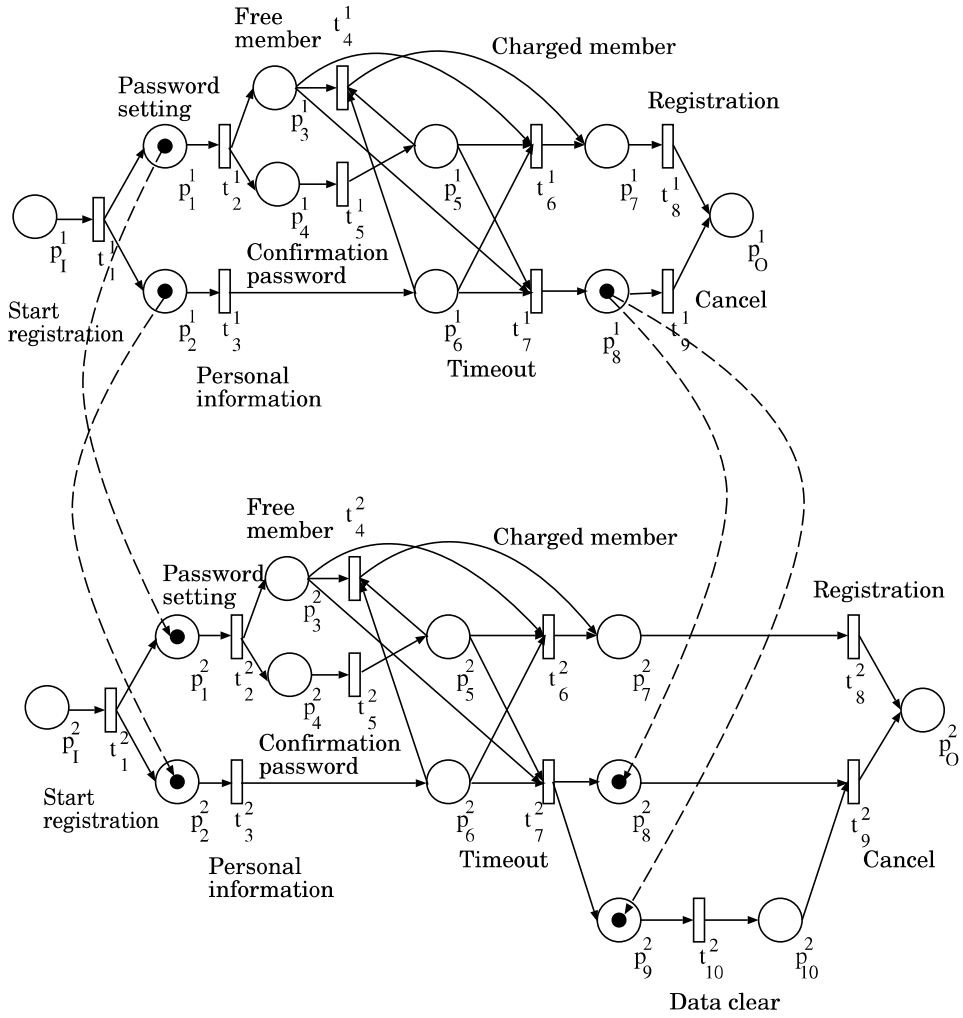


Figure 5.3: Dynamic workflow change of $(N_{ex1}, [p_1^1, p_2^1, p_8^1])$ to $(N_{ex2}, [p_1^2, p_2^2, p_8^2, p_9^2])$.

Next, we apply ALG2 to N_{ex2}. Step 1 ° finds a set of places with common output transitions. We have $S_{p_3} = \{p_3^2, p_5^2, p_6^2\}$. Step 2 ° applies ϕ_{Best} to the EFC structure constructed by places p_3^2, p_5^2, p_6^2 and transitions t_4^2, t_6^2, t_7^2. The resulting net (N_{ex3}, M_{ex3}) is shown in Fig. 5.5. N_{ex3} is a WS WF-net.

Next, we check the reachability by using the WS WF-net. Yamaguch has proposed a necessary and sufficient condition on the reachability problem [32] for acyclic WS WF-net, and has also given a polynomial-time algorithm to solve the problem [32] .

Property 5.1 *Let N be an acyclic WS WF-net. A marking M is reachable from* $[p_I{}^k]$ *in N iff (i) let* $\mathbf{A}_{\overline{N}}$ *be the incidence matrix of* $\overline{N}$*, the equation* $\mathbf{A}_{\overline{N}} \cdot \mathbf{X} = \vec{M} - [\vec{p_I{}^k}]$ *has some rational-valued solution for* $\mathbf{X}$*; and (ii) M marks every proper trap of* $\overline{N}$ *[16].* □

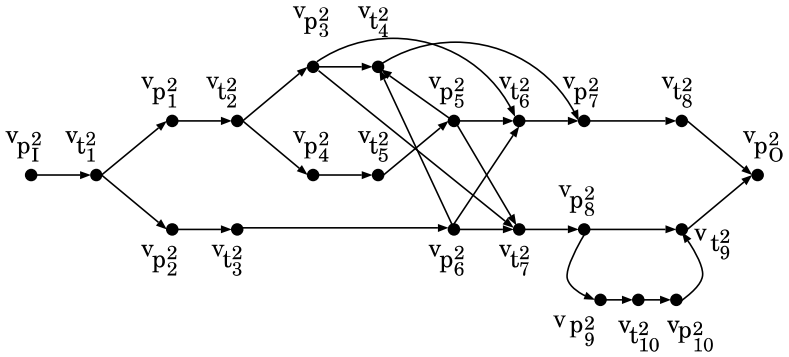


Figure 5.4: Flow Network of N_{ex2}

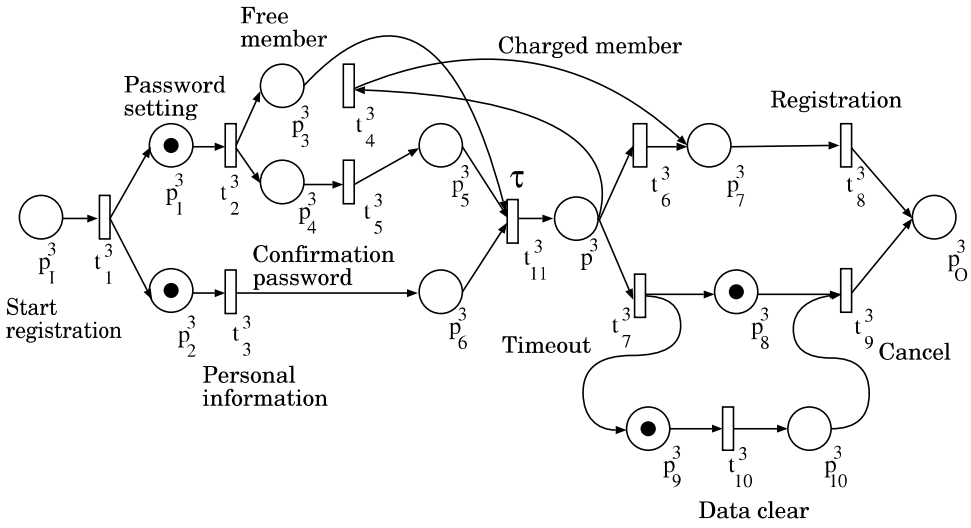


Figure 5.5: N_{ex3} refactored from N_{ex2}

Let us check whether M_{ex3} is correct (M_{ex3} is reachable from $[(p_I^3)^2]$) by using our procedure. Firstly, we check the equation has some rational-valued solution for $\mathbf{X}$. The equation

$$\mathbf{A}_{\overline{N_{ex3}}} \cdot \mathbf{X} = \vec{M_{ex3}} - [(\vec{p_I^3})^2]$$

$$
\begin{array}{c}
\\
p_I^3 \\ p_1^3 \\ p_2^3 \\ p_3^3 \\ p_4^3 \\ p_5^3 \\ p_6^3 \\ p_7^3 \\ p_8^3 \\ p_9^3 \\ p_{10}^3 \\ p^3 \\ p_O
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} t_1^3 & t_2^3 & t_3^3 & t_4^3 & t_5^3 & t_6^3 & t_7^3 & t_8^3 & t_9^3 & t_{10}^3 & t_{11}^3 & t_*^3 \end{array} \\
\begin{pmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1
\end{pmatrix}
\end{array}
\cdot \mathbf{X} =
\begin{pmatrix}
-2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0
\end{pmatrix}
$$

has a rational-valued solution

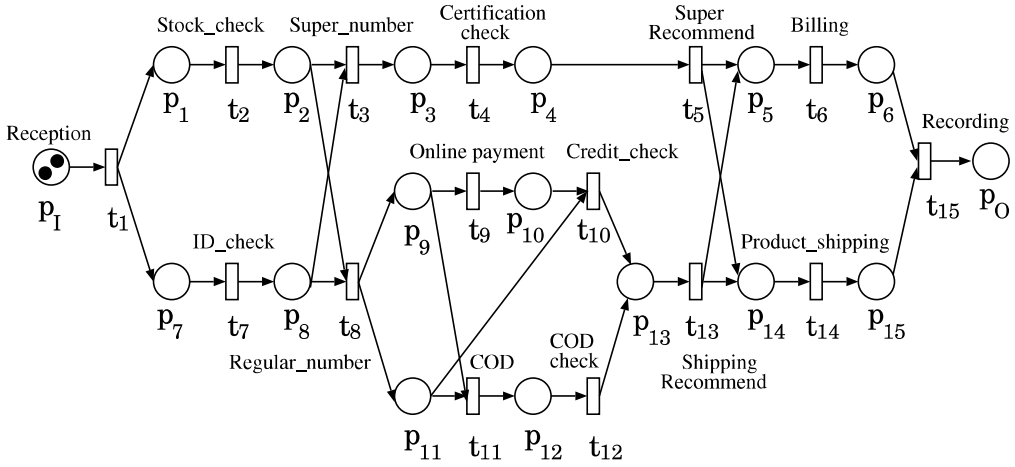
$$
\mathbf{X} = \begin{array}{c}
\begin{array}{cccccccccccc} t_1^3 & t_2^3 & t_3^3 & t_4^3 & t_5^3 & t_6^3 & t_7^3 & t_8^3 & t_9^3 & t_{10}^3 & t_{11}^3 & t_*^3 \end{array} \\
\begin{pmatrix} 2 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}^T
\end{array}.
$$

Secondly, we check whether M_{ex3} marks every trap of $\overline{N_{ex3}}$.
The maximal trap $\overline{Q_{ex3}}$ in $\overline{P_{ex3}} \setminus \{p_1^3, p_2^3, p_8^3, p_9^3\} = \{p_I^3, p_3^3, p_4^3, p_5^3, p_6^3, p_7^3, p_{10}^3, p_O^3\}$ is $\emptyset$. Therefore, M_{ex3} marks every proper trap of $\overline{N_{ex3}}$. As a result, we can say the algorithm outputs yes. This means that M_{ex3} is reachable from $[(p_I^3)^2]$ in N_{ex3}. This means that the workflow instances correctly moved to N_{ex3} from N_{ex2} . On the other hand, we confirm whether the original net get the same result about reachability. We see sequence of transition's fire of N_{ex2}. There is a marking $M_{ex2} = [p_1^2, p_2^2, p_8^2, p_9^2]$ if we fire transitions like $t_1^2 t_2^2 t_3^2 t_4^2 t_5^2 t_*^2 t_7^2 t_1^2$. Thus M_{ex2} is reachable from $[(p_I^2)^2]$.

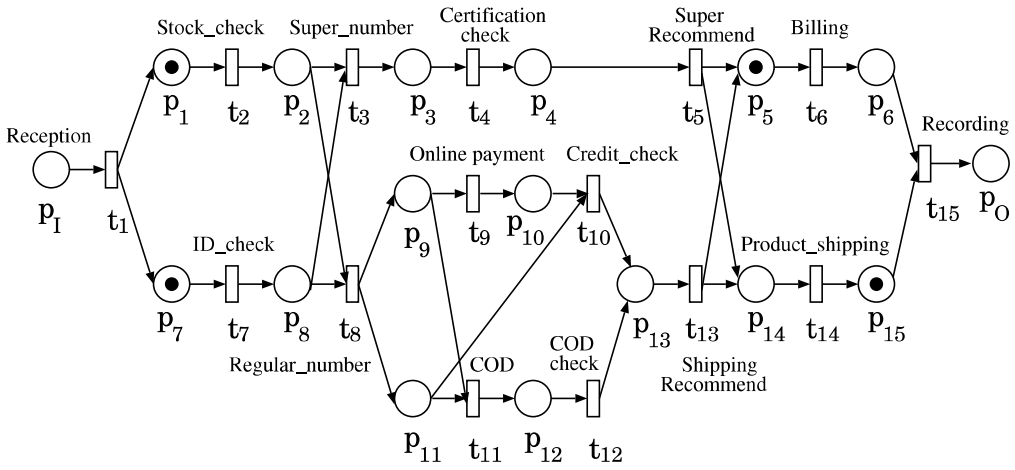
5.2 Reachability Checking : Case 2

Let us consider another example of the reachability problem. In this example, we deal with an ordering workflow with two instances. Assume that this workflow is modeled as a sound acyclic EFC WF-net N_3 shown in Fig. 5.6. Figure 5.6 (a) and (b) respectively represent the initial state $[{p_I}^2]$ and the current state $[p_1, p_5, p_7, p_{15}]$. Is the current state correct? Formally speaking,

Is $[p_1, p_5, p_7, p_{15}]$ reachable from $[{p_I}^2]$ in N_3?



(a) The initial state $[p_I{}^2]$.



(b) The current state $[p_1, p_5, p_7, p_{15}]$.

Figure 5.6: An example of the reachability problem. The input is a sound acyclic EFC WF-net N_3.

Yamaguchi [32] has proposed polynomial time procedures to solve the reachability problem for (i) sound EFC WF-nets with a marking representing one workflow instance; and (ii) acyclic WS WF-nets with a marking representing one or more workflow instances. Unfortunately, N_3 is a sound EFC WF-net with a marking representing two workflow instances. Therefore, the procedures cannot be applied to this example.

To solve this problem, we try to apply our 3 gate refactoring algorithm to N_3, and then apply Yamaguchi's procedure to the refactored WS WF-net. We first apply the 3 gate refactoring algorithm to N_3. Figure 5.7 shows the progress of the algorithm. In Step 1°, ALG1 is applied

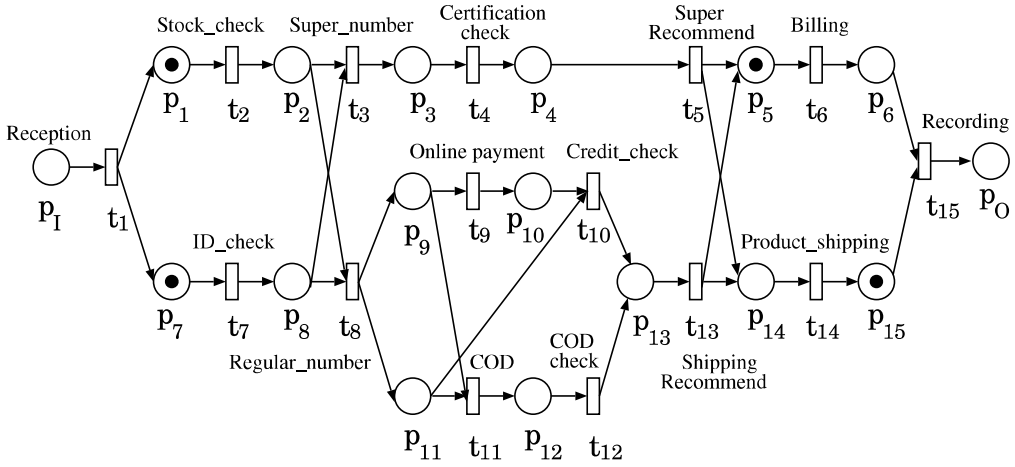
to N_3. ALG1 removes the implicit place p_{11}. Figure 5.7 (b) shows the acyclic EFC WF-net N_4 obtained by applying ALG1 to N_3. In Step 2°, ALG2 removes the EFC structure composed of nodes t_3, t_8 and p_2, p_8. In Step 3°, ALG3 finds a PT-handle $p_{16}t_8p_9t_9p_{10}t_{10}p_{13}t_{13}p_{14}t_{14}p_{15}t_{15}$ and thus outputs "Yes". In Step 4°, ALG4 finds a TP-bridge $t_{13}p_5$ for the PT-handle, and thus outputs "Yes". In Step 5°, ALG5 removes the TP-cross structure composed of nodes t_5, t_{13} and p_5, p_{14}. Figure 5.7 (c) shows the acyclic WS WF-net N_5 obtained by applying ALG2, ALG3, ALG4, and ALG5 to N_4. The 3 gate refactoring algorithm outputs N_5 as a WS WF-net.

Next, we apply Yamaguchi's procedure to the refactored WS WF-net N_5.

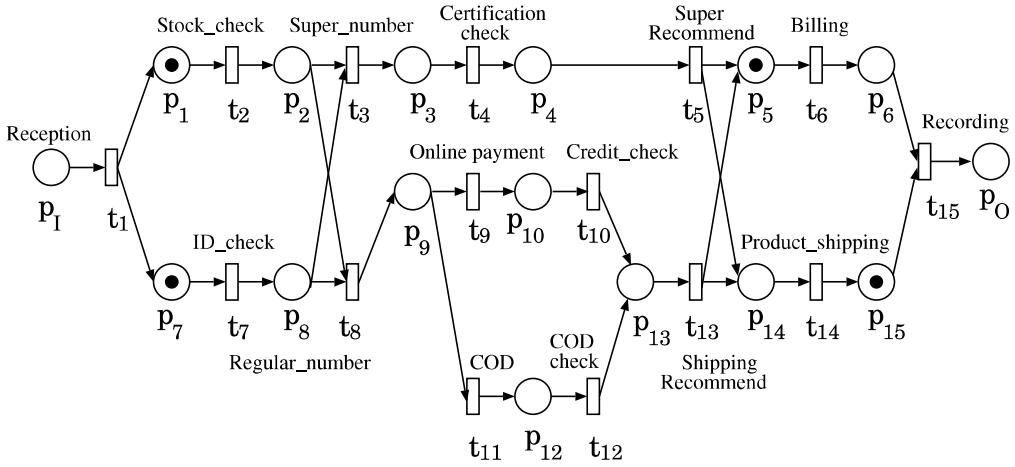
Firstly, we check the equation has some rational-valued solution for $\mathbf{X}$. The equation

$$\mathbf{A}_{\overline{N_5}} \cdot \mathbf{X} = [p_1, p_5, \vec{p_7}, p_{15}] - [\vec{p_I}^2]$$

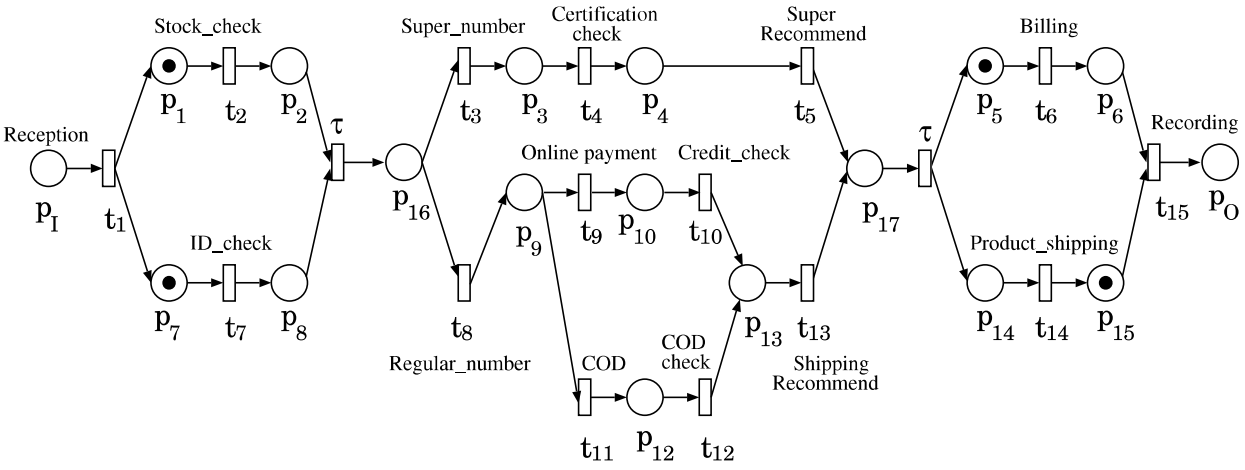
has a rational-valued solution
$\mathbf{X} = (2, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0)$. Secondly, we check whether $[p_1, p_5, p_7, p_{15}]$ marks every trap of $\overline{N_5}$. The maximal trap $\overline{Q_5}$ in $\overline{P_5} \setminus \{p_1, p_5, p_7, p_{15}\}$ $= \{p_I, p_2, p_3, p_4, p_6, p_8, p_9, p_{10}, p_{12}, p_{12}, p_{13}, p_{14}, p_{16}, p_{17}, p_O\}$ is $\emptyset$. Therefore, $[p_1, p_5, p_7, p_{15}]$ marks every proper trap of $\overline{N_5}$. As a result, the algorithm outputs yes. This means that $[p_1, p_5, p_7, p_{15}]$ is reachable from $[{p_I}^2]$ in N_5. We can say that $[p_1, p_5, p_7, p_{15}]$ is reachable from $[{p_I}^2]$ in N_3, i.e. the given workflow instances are correct.



(a) N_3.



(b) The WF-net N_4 obtained by applying ALG1 to N_3.



(c) The WF-net N_5 obtained by applying ALG2, ALG3, ALG4, and ALG5 to N_4.

Figure 5.7: The progress of the 3 gate refactoring algorithm.

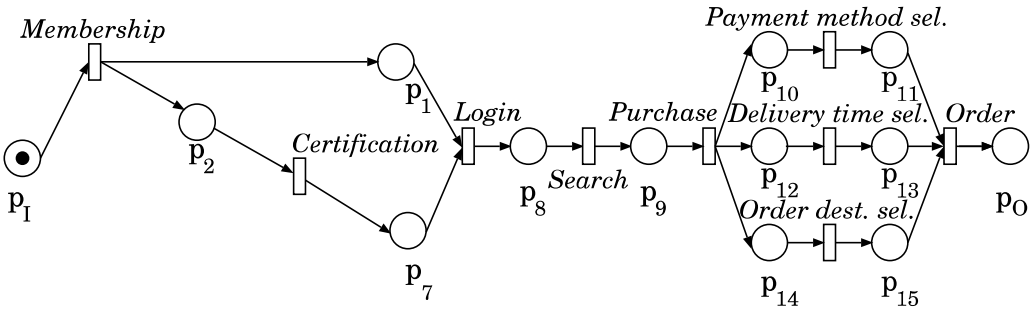
5.3 State Number Calculation

In WF-nets' analysis, it is important to know the state number to decide whether model checking is applicable. For example, SPIN [33][34] is available to WF-nets with less than 1 million states. The state number calculation problem is given as follows: Given a WF-net $(N, [p_I])$, calculate $|R(N, [p_I])|$? As an example, let us consider an extension of WF-net N_8 shown in Fig. 5.8. N_6 is a WF-net representing an ordering system. We extend N_6 to a WF-net N_7 to deal with non-members' order. For FC WF-nets, unfortunately, we cannot calculate the state number in polynomial time. On the other hand, for acyclic bridge-less WS WF-nets, we can caluculate it in polynomial time. Let us apply ALG1 to FC WF-net N_7 for refactoring to a WS WF-net N_8, and then calculate the state number by the method in Ref. [37][38][39].

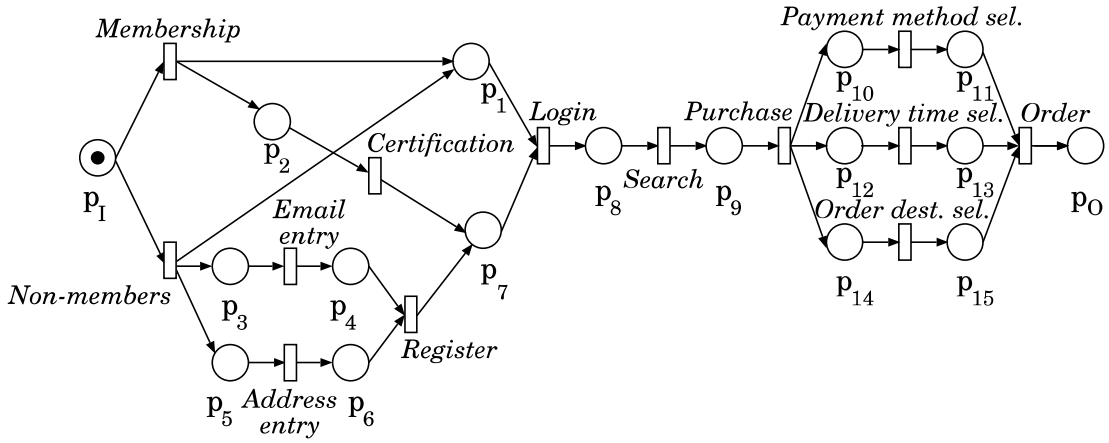
Removing p_1 from N_7 by ALG1, we can obtain N_8. N_8 is WS. From Theorem 3.3, N_8 is a result of refactoring. We can convert N_8 to the process tree shown in Fig. 5.9. We can calculate the state number of $(N_8, [p_I])$ as 18 by using the process tree. Now, we introduce one property about removing implicit place.

Property 5.2 Let N be a sound acyclic EFC WF-net, and N' a net obtained by removing implicit places from N. If N' is an acyclic WS WF-net, $|R(N, [p_I])| = |R(N', [p_I])|$ holds. □

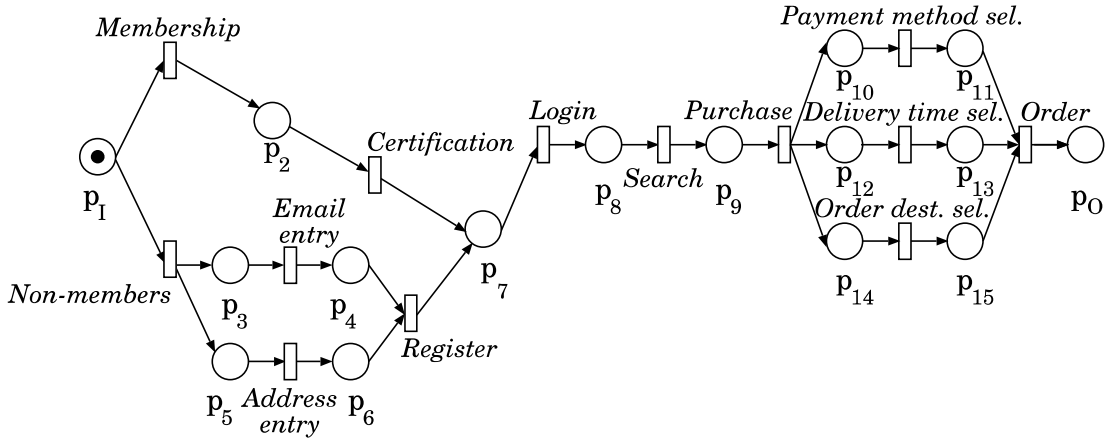
Proof: As shown in Eq. (3.1), the binary relation $\mathcal{R}$ is one-to-one correspondence. Therefore, we have $|R(N, [p_I])| = |R(N', [p_I])|$. **Q.E.D.** Regarding above property, the state number will not change, so the state number of $(N_7, [p_I])$ is also 18. As stated above, the state number serves as a criterion for selecting analysis methods. SPIN is available for analyzing the behavior of N_8.



(a) A WF-net N_6 representing an ordering system.



(b) The FC WF-net N_7 obtained by extending N_6.
It represents the extended ordering system.



(c) The WS WF-net N_8 obtained by
removing implicit place p_1. N_8 is a result of refactoring of N_7.

Figure 5.8: An application example to state number calculation.

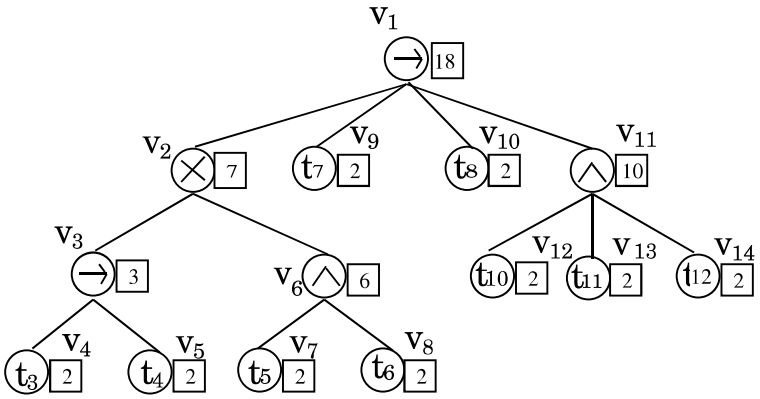


Figure 5.9: The process tree of $(N_8, [p_I])$.

5.4 Response Property Checking

We present an application example of our refactoring algorithm to Response property checking. A workflow consists of many activities. Some of the activities can always occur in any workflow instance, but others can not. As an example, let us consider a workflow for order processing. This workflow may be changed to adopt to the following new business rule.

Business rule R_1: If the price of the ordered item is over \$1,000, then the manager always accepts the item.

There are two cases in this workflow. Case 1: If the price is under \$1,000 (Order less than a thousand), then the ID of the staff will be checked (ID check). Next, the staff orders the item (Ordering), and then checks the booking of ordering item (Check). Finally, the staff accepts the item (Regular acceptance). Case 2: If the price is over \$1,000 (Order more than a thousand), then the staff orders the item (Ordering), and then checks the booking of ordering item (Check). Finally, the manager accepts the item (Manager acceptance).

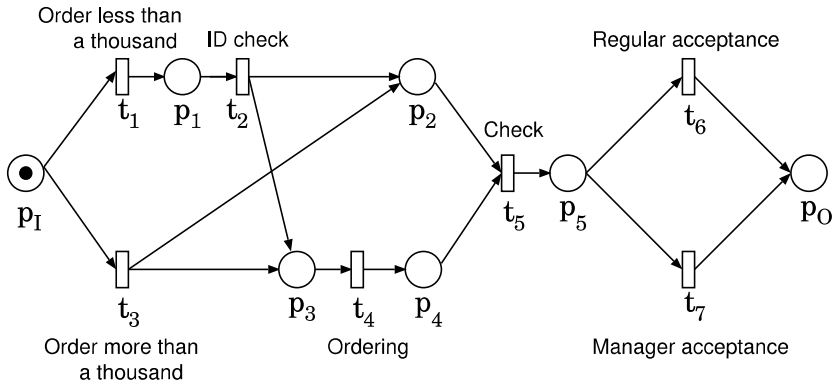
However, there is another case in this workflow. Even if the price is over \$1,000 (Order more than a thousand), then the staff accepts the item (Regular acceptance) instead of the manager. When this situation happens, we can say the business rule is violated.

It is important to check whether the workflow follows the business rule. To do so, we make use of WF-net and its analysis techniques. The workflow can be modeled as an FC WF-net N_5 shown in Fig. 3.9 (a). In terms of WF-nets, business rule R_1 is rewritten as following property.

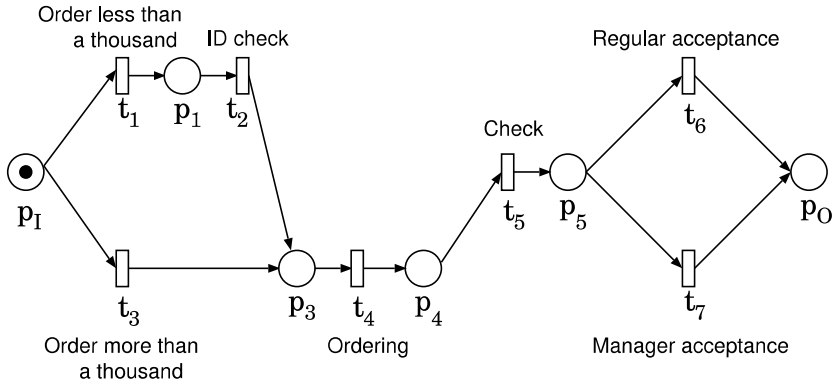
R_1: If t_3 fired, then t_7 always fires.

To check such a property, called response property, Bin Ahmadon et al. [37] have proposed a polynomial time procedure. Unfortunately, the procedure cannot be used for FC WF-nets. In order to utilize the method, we should transform FC WF-net N_5 to a WS WF-net N_6 by applying ALG3. Since p_2 is implicit, p_2 is removed from N_5. We can obtain N_6 shown in Fig. 5.10 (b) and N_6 is WS. Thus N_5 is refactorizable and N_6 is the result of refactoring.

After the refactoring, we can utilize Bin Ahmadon et al.'s procedure to check whether R_1 holds for N_6. As a result, for N_6, R_1 does not hold. This is because, if t_3 fires, t_7 cannot always fire. This enables us to know whether the workflow follows the new business rule in polynomial time.



(a) A sound acyclic FC WF-net N_5.



(b) A sound acyclic WS WF-net N_6 obtained by removing the implicit place p_2 in N_5.

Figure 5.10: An application example.

5.5 Remarks

We presented three application examples, “Reachability Checking”, “State Number Calculation” and “Response Property Checking”. They reduce the costs in system design process. As future work, we have to address them to upstream of system development.

Chapter 6

Refactoring of Timed System

6.1 Introduction

With the spread of systems which include time concept (e.g.communication protocol, bioinformatics) these days, the analysis of these systems has become indispensable. For many years, some mathematical models have been studied to analize timed system [40]. Timed Petri nets [41] are useful to model and analyze systems which include time concept, and have made a lot of contributions. One of the analysis methods of timed Petri nets is model checking. Model checking is an automatic and usually quite fast verification technique for finite state concurrent systems. Model checking can analyze comprehensively, however, there is a state-space explosion problem. Many researchers have tried to reduce this problem.

There are two approaches to reduce the state space of a timed Petri net. One is reduction methods. The other is translation methods from timed Petri nets to timed automata. Reduction methods make net size small to make state space small. Murata [16] has proposed a reduction method for untimed Petri nets. Sloan et al. [42] have proposed a reduction method for time Petri nets which maintain equivalence. However, there is no reduction method for timed Petri nets which consider observability [8]. Observability means whether actions are observed or not. All actions in a system are divided into observable and unobservable actions. When considering observability, model checking would become available to larger-scale systems because users should check only a part to observe, and state space can be reduced.

6.2 Timed Petri Nets and Timed Branching Bisimulation

Let A be a set. A^n is an n-dimensional vector on A. The usual operators $+, -, <$ and $=$ are used on A^n with $A = \mathbb{N}, \mathbb{Q}, \mathbb{R}$, where $\mathbb{N}$ is the set of natural number, $\mathbb{Q}$ is the set of quotient, $\mathbb{R}$ is the set of

real number. $\mathbb{R}_{\geq 0}$ is the set of positive real number. For a *valuation* $\mathbf{v} \in A^n$, $d \in A$, $\mathbf{v} + d$ denotes the vector with $\forall i \in [1..n]$, $(\mathbf{v}+d)_i = \mathbf{v}_i + d$, and for $A' \subseteq A$, $\mathbf{v}[A' \mapsto 0]$ denotes the valuation $\mathbf{v}'$ with $\mathbf{v}'(x) = 0$ for $x \in A'$ and $\mathbf{v}'(x) = \mathbf{v}(x)$ otherwise. $\mathcal{C}(X)$ denotes the *simple-constraints* over a set of variables X. $\mathcal{C}(X)$ is defined to be the set of boolean combinations (with the correctives $\{\wedge, \vee, \neg\}$) of terms of the form $x - x' \bowtie c$ or $x \bowtie c$ for $x, x' \in X$ and $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. Given a formula $\varphi \in \mathcal{C}(X)$ and a valuation $\mathbf{v} \in A^n$, $\varphi(\mathbf{v})$ denotes the truth value obtained by substituting each occurrence of x in φ by $\mathbf{v}(x)$. For a transition system we write transitions as $s \xrightarrow{a} s'$. Let Act be a nonempty set of observable actions, and τ a special action to represent unobservable events, with $\tau \notin Act$. We use Act_τ to denote $Act \cup \{\tau\}$. The time domain $Time$ is a totally ordered set with a least element 0. We say that $Time$ is discrete if for each pair $u, v \in Time$ there are only finitely many $w \in Time$ such that $u < v < w$.

Timed Labeled Transition System

Definition 6.1 (Timed Labeled Transition Systems) [43]

A timed labeled transition system (TLTS for short) is a tuple $(\mathcal{S}, \mathcal{T}, \mathcal{U})$, where: $\mathcal{S}$ is a set of states, including a special state $\surd$ to represent successful termination; $\mathcal{T} \subseteq \mathcal{S} \times Act_\tau \times Time \times \mathcal{S}$ is a set of transitions; $\mathcal{U} \subseteq \mathcal{S} \times Time$ is a delay relation, which satisfies: (a)if $\mathcal{T}(s, act, u, r)$, then $\mathcal{U}(s, u)$ and (b)if $u < v$ and $\mathcal{U}(s, v)$, then $\mathcal{U}(s, u)$. □

Transition (s, act, u, s') expresses that state s evolves into state s' by the execution of action act at (absolute) time u.

Labeled Timed Petri Nets

Definition 6.2 (Labeled Timed Petri Net) [41]

A labeled timed Petri net N (LTPN for short) is a tuple $(P, T, \mathcal{E}, M_0, D, Act_\tau)$ where: P is a set of places; T is a set of transitions; $\mathcal{E} \subseteq (P \times T) \cup (T \times P)$ is a set of arcs; $M_0 \in \mathbb{N}^P$ is the initial marking; $D : T \to \mathbb{N}$ is the function to assign delay time to a transitions. d_i denotes $D(t_i)$. Act_τ is the set of action. □

Let x be a place or transition. ${}^\bullet x$ and $x^\bullet$ respectively denote $\{y | (y, x) \in \mathcal{E}\}$ and $\{y | (x, y) \in \mathcal{E}\}$. x is called a *split* if $|x^\bullet| \geq 2$. x is a called a *join* if $|{}^\bullet x| \geq 2$. $\mathbf{v}_N \in (\mathbb{R}_{\geq 0})^T$ is a *valuation* such that each value $\mathbf{v}_{Ni}$ is the elapsed time since the last time transition t_i was enabled. $\mathbf{0}$ is the initial valuation with $\forall i \in [1..n], \mathbf{0}_i = 0$. A *marking* M of a LTPN is a mapping in $\mathbb{N}^P$ and $M(p_j)$ is the number of tokens in place p_j. We represent M as a bag over P :$M = [p^{M(p)} | p \in P, M(p) > 0]$. A transition

t is enabled in a marking M iff $M \geq {}^\bullet t$. This is represented by the predicate $\uparrow enabled1(t_i, M)$ is true iff $M \geq {}^\bullet t_i$. It is necessary to the following firing rule when a transition is enabled after firing the other transition. The predicate $\uparrow enabled2(t_k, M, t_i)$ is true if t_k is enabled by the firing of transition t_i from marking M, and false otherwise. In this framework, a transition t_k is *newly enabled* after firing t_i in marking M if "it is not enabled by $M - {}^\bullet t_i$ and is enabled by $M' = M - {}^\bullet t_i + t_i{}^\bullet$". Formally this is given by $\uparrow enabled2(t_k, M, t_i) = (M - {}^\bullet t_i + t_i{}^\bullet \geq {}^\bullet t_k) \wedge ((M - {}^\bullet t_i < {}^\bullet t_k) \vee (t_k = t_i))$ The semantics of LTPNs can be given in term of TLTS.

Definition 6.3 (The semantics of LTPNs) [41]

The semantics of a LTPN N is a TLTS $(\mathcal{S}_N, \mathcal{T}_N, \mathcal{U}_N)$ where: $\mathcal{S}_N = \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^T$, $\mathcal{T}_N = \mathcal{S}_N \times Act_\tau \times D \times \mathcal{S}_N$, $\mathcal{U}_N = \mathcal{S}_N \times Time$ consists of the discrete and continuous transition relation;

- The discrete transition relation is defined for all $t_i \in T$ by $(M, \mathbf{v}_N) \overset{act(t_i)}{\rightarrow}_u (M', \mathbf{v}'_N)$ iff:

$$\begin{cases} M \geq {}^\bullet t_i \wedge M' = M - {}^\bullet t_i + t_i{}^\bullet \\ \mathbf{v}_{Ni} = d_i \\ \mathbf{v}'_{Nk} = \begin{cases} 0 & \text{if } \uparrow enabled2(t_k, M, t_i) \\ \mathbf{v}_{Nk} & \text{otherwise} \end{cases} \end{cases} \tag{6.1}$$

- The continuous transition relation is defined by $\mathcal{U}((M, \mathbf{v}_N), u)$ iff:

$$\begin{cases} M' = M \\ \mathbf{v}'_N = \mathbf{v}_N + u \\ \forall k \in [1..|T|], (M_N \geq {}^\bullet t_k \Rightarrow \mathbf{v}'_{Nk} \leq d_k) \end{cases} \tag{6.2}$$

□

If delay time $D(t_i)$ passes, transition t_i will fire immediately. The firing order of simultaneous firing of two or more transitions is nondeterministic.

The reachability graph of a LTPN is a labeled directed graph $G = (V, \mathsf{E})$, where V is the set of markings of the LTPN, and E is the set of arcs labeled with transitions representing all possible transition firings.

Timed Branching Bisimulation Timed branching bisimulation (TBB for short) is an equivalence relation on temporal behavior. For $u \in Time$, the reflexive transitive closure of $\overset{\tau}{\rightarrow}_u$ is denoted by $\Rightarrow_u$.

Definition 6.4 (Timed Branching Bisimulation) [43]

Assume a TLTS $(\mathcal{S}, \mathcal{T}, \mathcal{U})$. A collection B of binary relations $B_u \subseteq \mathcal{S} \times \mathcal{S}$ for $u \in Time$ is a *timed branching bisimulation* if sB_ut implies: (1) if $s \xrightarrow{act}_u s'$, then (i)either $act = \tau$ and $s'B_ut$, (ii) or $t \Rightarrow_u \hat{t} \xrightarrow{act} t'$ with $sB_u\hat{t}$ and $s'B_ut'$; (2) if $t \xrightarrow{act}_u t'$, then vice versa; (3) if $s \downarrow$, then $t \Rightarrow_u t' \downarrow$ with sB_ut'; (4) if $t \downarrow$, then vice versa; (5) if $u < v$ and $\mathcal{U}(s, v)$, then for some $n \geq 0$ there are $t_0, \cdots, t_n \in \mathcal{S}$ with $t = t_0$ and $\mathcal{U}(t_n, v)$ and $u_0 < \cdots < u_n \in Time$ with $u = u_0$ and $v = u_n$, such that for $i < n$, $t_i \Rightarrow_{u_i} t_{i+1}$ and sB_ut_{i+1} for $u_i \leq w \leq u_{i+1}$; (6) if $u < v$ and $\mathcal{U}(t, v)$, then vice versa. □

6.3 Timed Behavioral Inheritance and the Reduction Operators

In this section, we propose reduction operators with consideration of observability. We define the concept of timed behavioral inheritance. Then we give definitions of reduction operators for safe LTPNs.

A Concept of Timed Behavioral Inheritance

At first, we define the concept of timed behavioral inheritance. Timed behavioral inheritance is regarded as an extension of behavioral inheritance [8]. We add time concept to behavioral inheritance. This concept means the equivalence of the systems with time concept in consideration of observability. We use the following operator called *abstraction*.

- For any $I(\subseteq \Omega)$, the abstraction operator τ_I is a function that renames all transition labels in I to τ. Formally, $\tau_I : N \mapsto (P, T, \mathcal{E}, D, M_0, Act'_\tau)$ such that $\forall t \in T : (act(t) \in I \Rightarrow Act'_\tau(t) = \tau)$ and $(act(t) \notin I \Rightarrow Act'_\tau(t) = Act_\tau)$.

Using the operator, we give the definition of the concept:*timed projection inheritance* that is one concept of timed behavioral inheritance. [1]

Definition 6.5 (Timed Projection Inheritance) Let N_X, N_Y be LTPNs. N_X is a subclass under *timed projection inheritance* of N_Y iff there is a set $I(\in Act_\tau)$ of transision labels that TLTS of N_X: $(\mathcal{S}_X, \mathcal{T}_X, \mathcal{U}_X)$ and TLTS of N_Y: $\tau_I(\mathcal{S}_Y, \mathcal{T}_Y, \mathcal{U}_Y)$ are TBB. □

Theorem 6.1 If N_X is a subclass under timed projection inheritance of N_Y, firing schedules of transitions which have external action labels of N_X and N_Y are equal. □

[1] We add more two concepts to timed behavior inheritance, *timed protocol inheritance* and *timed life-cycle inheritance*. We will report them in the near future.

Proof : If N_X is a subclass under timed projection inheritance of N_Y, TLTS of N_X and TLTS of N_Y are timed branching bisimilar. From [43], timed branching bisimilarity is an equivalence relation. From [41], an equivalence relation is that firing schedules of transitions which are left completely unmodified by applying operators are equal. Therefore, firing schedules of transitions which have external action labels of N_X and N_Y are equal. **Q.E.D.**

Now, we represent an example of *timed projection inheritance*.Let us consider a LTPN $(N_0, [p_1])$ shown in Fig. 6.1. This LTPN corresponds to a sequential business process which consists of three tasks: *register*, *handle*, and *archive*. The firing schedule of N_0 is as follows: After 15 time unit from the initial state, t_1 fires and *register* is carried out. After 10 time unit

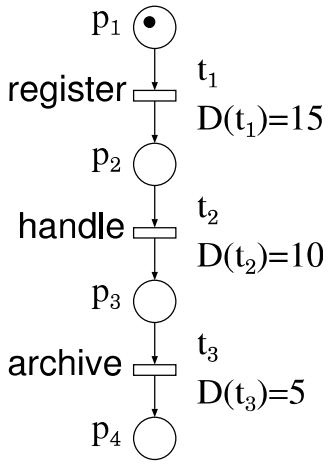


Figure 6.1: A LTPN N_0 representing business process

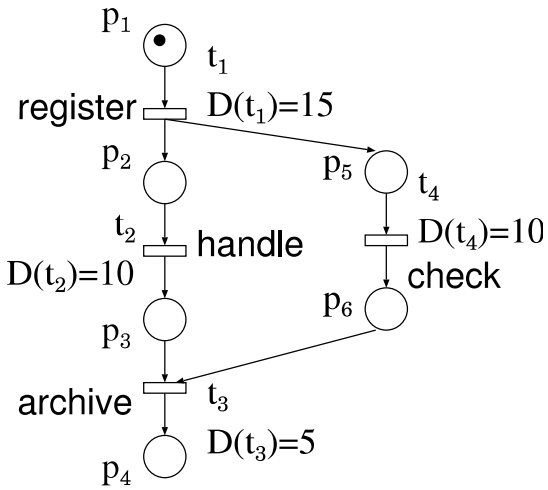


Figure 6.2: A LTPN N_1 that task *check* is executed in parallel with task *handle*.

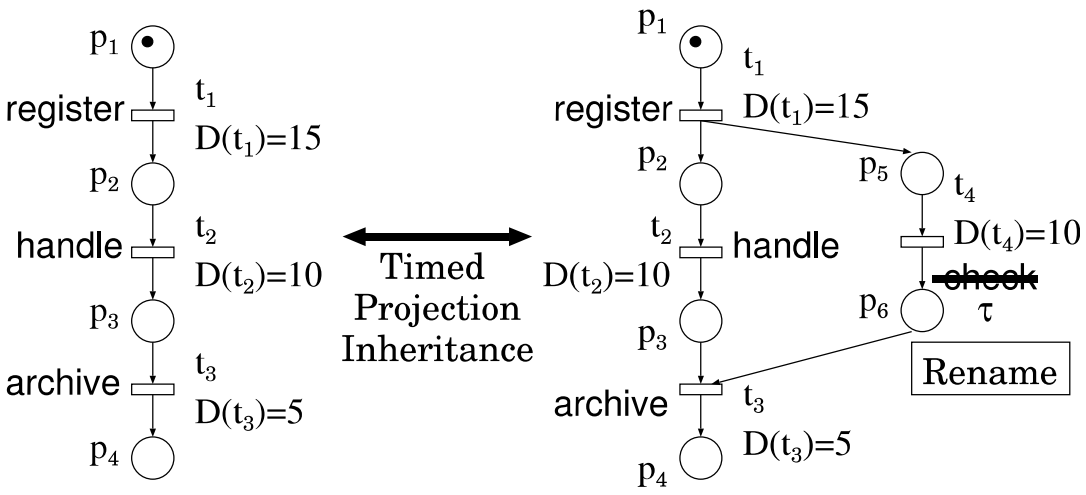


Figure 6.3: N_1 is a subclass of N_0 with respect to timed projection inheritance

from t_1 fired (after 25 time unit from the initial state), t_2 fires and *handle* is carried out. After 5 time unit from t_2 fired (after 30 time unit from the initial state), t_3 fires and *archive* is carried out. Let us consider an extension of N_0. N_1 shown in Fig. 6.2 extends N_0 with additional tasks *check*. task *check* is executed in parallel with task *handle*. N_2 is a subclass of N_0 with respect to timed projection inheritance(Fig. 6.3); if *check* is abstracted, we cannot distinguish N_1 and N_0. Formally this is given by that TLTS of N_0: $(\mathcal{S}_0, \mathcal{T}_0, \mathcal{U}_0)$ and TLTS of $\tau_{\{check\}}(N_1)$: $(\mathcal{S}_2, \mathcal{T}_2, \mathcal{U}_2)$ are TBB. N_1 is a subclass of N_0 with respect to timed behavioral inheritance, so that each firing schedule of task *register*, *handle*, and *archive* is equal.

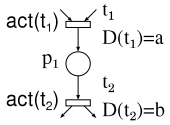
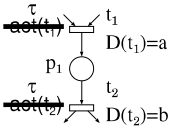
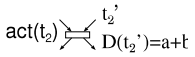
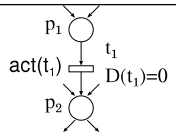
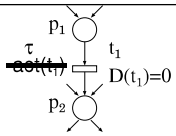
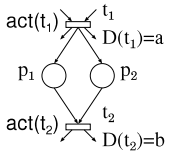
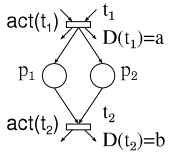
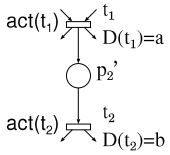
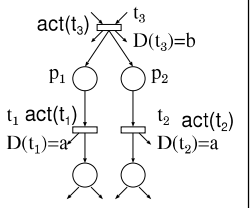
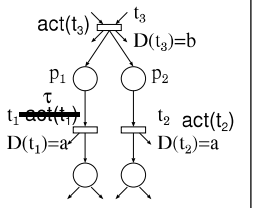
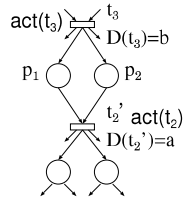
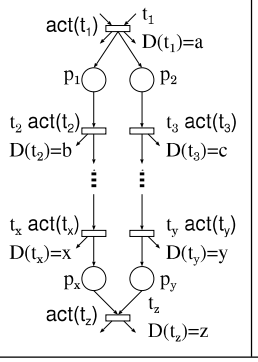
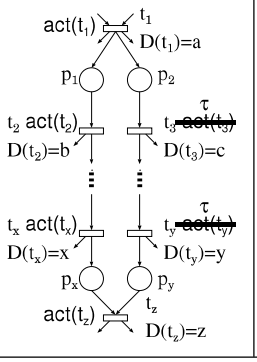
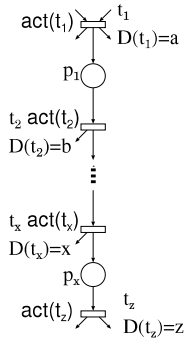
The Reduction Operators for Safe Labeled Timed Petri Nets Based on Timed Behavioral Inheritance

We propose reduction operators for safe timed Petri nets based on timed behavioral inheritance. The reduction operators are applicable to safe LTPNs. We propose five reduction operators and prove those five operators maintain timed behavioral inheritance. Table 6.1 shows N, $\tau(N)$ obtained by applying abstraction operator, and N' which derived from N We give definitions of the five operators and propose a theorem on the operators at the end of this section.

Fusion of Series Transitions (FST)

- *Preconditions*: There exist two transitions ${}^N t_1$ and ${}^N t_2$ and a place ${}^N p_1$ satisfying:
  - Place ${}^N p_1$ is unmarked in the initial marking.

Table 6.1: Reduction operators for safe timed Petri nets based on timed behavioral inheritance

	N	$\tau(N)$	N'
Fusion of Series Transitions (FST)			
Fusion of Series Places (FSP)			
Fusion of Parallel Places (FPP)			
Lateral Fusion (LF)			
Reduction of Parallel Paths (RPP)			

  - ${}^{N}t_1{}^{\bullet} = \{{}^{N}p_1\} = {}^{N\bullet}t_2$. Also, $\{{}^{N}p_1\} \notin {}^{N\bullet}t_1$ and $\{{}^{N}p_1\} \notin {}^{N}t_1{}^{\bullet}$.
  - Place ${}^{N}p_1$ is disconnected from all other transitions.
  - Either ${}^{N}({}^{\bullet}t_1)^{\bullet} = \{{}^{N}t_1\}$, or $D({}^{N}t_1) = 0$.
  - N' is safe.
  - Action $act({}^{N}t_1)$ is unobserved.

- *Operation*: Action $act({}^{N}t_1)$ is renamed to τ. Transitions ${}^{N}t_1$ and ${}^{N}t_2$ may be replaced by a single transition ${}^{N'}t'_2$ with delay time $D({}^{N'}t'_2) = D({}^{N}t_1) + D({}^{N}t_2)$, with input set ${}^{N'\bullet}t'_2 = {}^{N\bullet}t_1$, and with output set ${}^{N'}t'_2{}^{\bullet} = {}^{N}t_2{}^{\bullet}$.

Fusion of Series Places (FSP)

- *Preconditions*: There exist a transition ${}^{N}t_1$ and places ${}^{N}p_1$ and ${}^{N}p_2$ satisfying:
  - Place ${}^{N}p_1$ is unmarked in the initial marking.
  - ${}^{N\bullet}t_1 = [{}^{N}p_1]$ and ${}^{N}t_1{}^{\bullet} = [{}^{N}p_2]$.
  - $D({}^{N}t_1) = 0$.
  - Action $act({}^{N}t_1)$ is unobserved.
- *Operation*: Action $act({}^{N}t_1)$ is renamed to τ. Places ${}^{N}p_1$ and ${}^{N}p_2$ may be replaced by a single place ${}^{N'}p'_2$ with input set ${}^{N'\bullet}p'_2 = {}^{N\bullet}p_1 \cup ({}^{N\bullet}p_2 - {}^{N}t_1{}^{\bullet})$, and with output set ${}^{N'}p'_2{}^{\bullet} = {}^{N}p_2{}^{\bullet}$.

Fusion of Parallel Places (FPP)

- *Preconditions*: There exist two transitions ${}^{N}t_1$ and ${}^{N}t_2$ and two places ${}^{N}p_1$ and ${}^{N}p_2$ satisfying:
  - Places ${}^{N}p_1$ and ${}^{N}p_2$ have the same number of initial tokens.
  - ${}^{N\bullet}p_1 = {}^{N\bullet}p_2 = \{{}^{N}t_1\}$, and ${}^{N}p_1{}^{\bullet} = {}^{N}p_2{}^{\bullet} = \{{}^{N}t_2\}$
- *Operation*: Places ${}^{N}p_1$ and ${}^{N}p_2$ may be replaced by a single place ${}^{N'}p'_2$ with input set ${}^{N'\bullet}p'_2 = {}^{N\bullet}p_2$, and with output set ${}^{N'}p'_2{}^{\bullet} = {}^{N}p_2{}^{\bullet}$.

Lateral Fusion (LF)

- *Preconditions*: There exist three transitions ${}^{N}t_1$, ${}^{N}t_2$ and ${}^{N}t_3$ and two places ${}^{N}p_1$ and ${}^{N}p_2$ satisfying:
  - Places ${}^{N}p_1$ and ${}^{N}p_2$ have the same number of initial tokens.
  - ${}^{N\bullet}p_1 = {}^{N\bullet}p_2 = \{{}^{N}t_3\}$. Also, ${}^{N}p_1{}^{\bullet} = \{{}^{N}t_1\}$, ${}^{N}p_2{}^{\bullet} = \{{}^{N}t_2\}$.
  - ${}^{N\bullet}t_1 = \{{}^{N}p_1\}$, ${}^{N\bullet}t_2 = \{{}^{N}p_2\}$.
  - $D({}^{N}t_1) = D({}^{N}t_2)$.
  - Action $act({}^{N}t_1)$ is unobserved.
- *Operation*: Action $act({}^{N}t_1)$ is renamed to τ. Transitions ${}^{N}t_1$ and ${}^{N}t_2$ may be replaced by a single transition ${}^{N'}t'_2$ with delay time $D({}^{N'}t'_2) = D({}^{N}t_2)$, with input set ${}^{N'\bullet}t'_2 = {}^{N\bullet}t_1 \cup {}^{N\bullet}t_2$, and with output set ${}^{N'}t'_2{}^{\bullet} = {}^{N}t_1{}^{\bullet} \cup {}^{N}t_2{}^{\bullet}$.

Reduction of Parallel Paths (RPP)

- *Preconditions*: There exist parallel paths $\mathcal{Y}_1(= \langle {}^N p_1 {}^N t_2 \cdots {}^N t_x {}^N p_x \rangle)$ and $\mathcal{Y}_2(= \langle {}^N p_2 {}^N t_3 \cdots {}^N t_y {}^N p_y \rangle)$ satisfying:
  - All places in $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are unmarked in the initial marking.
  - All places in $\mathcal{Y}_1$ have no input or output transitions in $\mathcal{Y}_2$. Also, all places in $\mathcal{Y}_2$ have no input or output transitions in $\mathcal{Y}_1$.
  - All transitions in $\mathcal{Y}_1$ have no input or output places in $\mathcal{Y}_2$. Also, all transitions in $\mathcal{Y}_2$ have no input or output places in $\mathcal{Y}_1$.
  - The sum of delay time in $\mathcal{Y}_1(= D({}^N t_2) + \cdots + D({}^N t_x))$ is greater than that in $\mathcal{Y}_2(= D({}^N t_3) + \cdots + D({}^N t_y))$.
  - All actions in $\mathcal{Y}_2$ are unobserved.
- *Operation*: All actions in $\mathcal{Y}_2$ are renamed to τ. Path $\mathcal{Y}_2$ may be reduced.

Theorem 6.2 Let N be a safe LTPN, and N' a safe LTPN obtained by applying one of the proposed operators. N' is a subclass of N under timed projection inheritance. □

Proof 6.1 *(FST)* : Let $\tau(N)$ be a safe LTPN which derived from a LTPN N by applying abstraction operator. Of course, $\tau(N)$ is a subclass of N with respect to timed projection inheritance. Let u be arrival time to ${}^{\tau(N)\bullet} t_1$. First, we prove that a binary relation B_u which ${}^{\tau(N)\bullet} t_1 B_u {}^{N'\bullet} t'_2$ holds is a TBB.

(1) In $\tau(N)$, ${}^{\tau(N)\bullet} t_1 \overset{{}^{\tau(N)} t_1}{\rightarrow}_{(u+a)} [{}^{\tau(N)} p_1]$.

- case i: $act({}^{\tau(N)} t_1) = \tau$ and $[{}^{\tau(N)} p_1] B_{(u+a)} {}^{N'\bullet} t'_2$ holds.

(2) In N', ${}^{N'\bullet} t'_2 \overset{{}^{N'} t'_2}{\rightarrow}_{(u+a+b)} {}^{N'} t'^{\bullet}_2$.

- case ii: In $\tau(N)$, ${}^{\tau(N)\bullet} t_1 \Rightarrow_{(u+a)} [{}^{\tau(N)} p_1] \overset{{}^{\tau(N)} t_2}{\rightarrow}_{(u+a+b)} {}^{\tau(N)} t_2{}^{\bullet}$, and ${}^{\tau(N)} t_2{}^{\bullet} B_{(u+a+b)} {}^{N'} t'^{\bullet}_2$ holds.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}({}^{\tau(N)\bullet} t_1, u+a)$. In N', $\mathcal{U}({}^{N'\bullet} t'_2, u+a)$ and ${}^{\tau(N)\bullet} t_1 B_{(u+a)} {}^{N'\bullet} t'_2$ holds.

(6) In N', $\mathcal{U}({}^{N'\bullet}t'_2, u+a+b)$. In $\tau(N)$, ${}^{\tau(N)\bullet}t_1 \Rightarrow_{(u+a)} [{}^{\tau(N)}p_1]$ and $\mathcal{U}({}^{\tau(N)}p_1, u+a+b)$, and ${}^{\tau(N)}p_1 B_{(u+a+b)} {}^{N'\bullet}t'_2$ holds.

Next, we prove that a binary relation B_{u+a} which $[{}^{\tau(N)}p_1] B_{(u+a)} {}^{N'\bullet}t'_2$ holds is a TBB.

(1) In $\tau(N)$, $[{}^{\tau(N)}p_1] \overset{{}^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} {}^{\tau(N)}t_2{}^{\bullet}$.

- case ii: In N', ${}^{N'\bullet}t'_2 \overset{{}^{N'}t'_2}{\rightarrow}_{(u+a+b)} {}^{N'}t'^{\bullet}_2$ and ${}^{\tau(N)}t_2{}^{\bullet} B_{(u+a+b)} {}^{N'}t'^{\bullet}_2$ holds.

(2) In N', ${}^{N'\bullet}t'_2 \overset{{}^{N'}t'_2}{\rightarrow}_{(u+a+b)} {}^{N'}t'^{\bullet}_2$.

- case ii: In $\tau(N)$, ${}^{\tau(N)\bullet}t_1 \Rightarrow_{(u+a)} [{}^{\tau(N)}p_1] \overset{{}^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} {}^{\tau(N)}t_2{}^{\bullet}$, and ${}^{\tau(N)}t_2{}^{\bullet} B_{(u+a+b)} {}^{N'}t'^{\bullet}_2$ holds.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([{}^{\tau(N)}p_1], u+a+b)$. In N', $\mathcal{U}({}^{N'\bullet}t'_2, u+a+b)$ and $[{}^{\tau(N)}p_1] B_{(u+a+b)} {}^{N'\bullet}t'_2$ holds.

(6) In N', $\mathcal{U}({}^{N'\bullet}t'_2, u+a+b)$. In $\tau(N)$, $\mathcal{U}({}^{\tau(N)}p_1, u+a+b)$, and ${}^{\tau(N)}p_1 B_{(u+a+b)} {}^{N'\bullet}t'_2$ holds.

In addition, it is obvious that a binary relation B_{u+a} which $[{}^{\tau(N)}p_1] B_{(u+a)} {}^{N'\bullet}t'_2$ holds is a TBB. Therefore, binary relations ${}^{\tau(N)\bullet}t_1 B_u {}^{N'\bullet}t'_2$ and $[{}^{\tau(N)}p_1] B_{(u+a)} {}^{N'\bullet}t'_2$ are held. This is held for all u, N and N' are timed branching bisimilar. Then N' is a subclass of N with respect to timed projection inheritance. **Q.E.D.**

(FSP) : Let $\tau(N)$ be a safe LTPN which derived from a LTPN N by applying abstraction operator. Of course, $\tau(N)$ is a subclass of N with respect to timed projection inheritance. Let u be arrival time to ${}^{\tau(N)}p_1$ and w be a shortest delay time of ${}^{\tau(N)}p_2{}^{\bullet}$. First, we prove that a binary relation B_u which $[{}^{\tau(N)}p_1] B_u [{}^{N'}p'_2]$ holds is a TBB.

(1) In $\tau(N)$, $[{}^{\tau(N)}p_1] \overset{{}^{\tau(N)}t_1}{\rightarrow}_{u} [{}^{\tau(N)}p_2]$.

- case i: $act({}^{\tau(N)}t_1) = \tau$ and $[{}^{\tau(N)}p_1] B_u {}^{N'}p'_2$ is held.

(2) In N', $[{}^{N'}p'_2] \overset{{}^{N'}p'^{\bullet}_2}{\rightarrow}_{(u+w)} {}^{N'}(p'^{\bullet}_2)^{\bullet}$.

- case ii: In $\tau(N)$, $[{}^{\tau(N)}p_1] \Rightarrow_u [{}^{\tau(N)}p_2] \overset{{}^{\tau(N)}p_2{}^{\bullet}}{\rightarrow}_{(u+w)} {}^{\tau(N)}(p_2{}^{\bullet})^{\bullet}$, and ${}^{\tau(N)}(p_2{}^{\bullet})^{\bullet} B_{(u+w)} {}^{N'}(p'^{\bullet}_2)^{\bullet}$ holds.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1], u+0)$. In N', $\mathcal{U}([^{N'}p'_2], u+0)$ and $[^{\tau(N)}p_1]B_u[^{N'}p'_2]$ holds.

(6) In N', $\mathcal{U}([^{N'}p'_2], u+w)$. In $\tau(N)$, $[^{\tau(N)}p_1] \Rightarrow_{(u)} [^{\tau(N)}p_2]$ and $\mathcal{U}([^{\tau(N)}p_2], u+w)$, and $[^{\tau(N)}p_2]B_u[^{N'}p'_2]$ holds.

In addition, it is obvious that a binary relation B_{u+w} which $[^{\tau(N)}p_2]B_u[^{N'}p'_2]$ holds is a TBB. Therefore, a binary relation $[^{\tau(N)}p_1]B_u[^{N'}p'_2]$ holds. This holds for all u, N and N' are timed branching bisimilar. Then N' is a subclass of N with respect to timed projection inheritance. **Q.E.D.**

(FPP) : Let $\tau(N)$ be a safe LTPN which derived from a LTPN N by applying abstraction operator. Of course, $\tau(N)$ is a subclass of N with respect to timed projection inheritance. Let u be arrival time to $^{\tau(N)\bullet}t_1$. It is obvious that a binary relation B_u which $^{\tau(N)\bullet}t_1 B_u {}^{N'\bullet}t_1$ holds and a binary relation $B_{(u+a+b)}$ which $^{\tau(N)}t_2{}^{\bullet} B_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$ holds are TBBs. Therefore, we prove that a binary relation B_u which $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_{(u+a)}[^{N'}p'_2]$ holds is a TBB.

(1) In $\tau(N)$, $[^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} {}^{\tau(N)}t_2{}^{\bullet}$.

- case ii: In N', $[^{N'}p'_2] \overset{^{N'}t_2}{\rightarrow}_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$, and $^{\tau(N)}t_2{}^{\bullet} B_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$ holds.

(2) In N', $[^{N'}p'_2] \overset{^{N'}t_2}{\rightarrow}_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$.

- case ii: In $\tau(N)$, $[^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} {}^{\tau(N)}t_2{}^{\bullet}$, and $^{\tau(N)}t_2{}^{\bullet} B_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$ holds.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1, {}^{\tau(N)}p_2], u+a+b)$. In N', $\mathcal{U}([^{N'}p'_2], u+a+b)$ and $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_u[^{N'}p'_2]$ holds.

(6) In N', $\mathcal{U}([^{N'}p'_2], u+a+b)$. In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1, {}^{\tau(N)}p_2], u+a+b)$ and $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_u[^{N'}p'_2]$ holds.

This holds for all u, N and N' are timed branching bisimilar. Then N' is a subclass of N with respect to timed projection inheritance. **Q.E.D.**

(LF) : Let $\tau(N)$ be a safe LTPN which derived from a LTPN N by applying abstraction operator. Of course, $\tau(N)$ is a subclass of N with respect to timed projection inheritance. Let u be arrival time to $^{\tau(N)\bullet}t_3$. It is obvious that a binary relation B_u which $^{\tau(N)\bullet}t_3 B_u {}^{N'\bullet}t_3$ holds and a

binary relation $B_{(u+b+a)}$ which $({}^{\tau(N)}t_1{}^{\bullet}, {}^{\tau(N)}t_2{}^{\bullet}) B_{(u+b+a)} {}^{N'}t_2{}^{\bullet}$ holds are TBBs. Therefore, we prove that a binary relation $B_{(u+b)}$ which $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] B_{(u+b)} [{}^{N'}p_1, {}^{N'}p_2]$ holds is a TBB.

(1) In $\tau(N)$, $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{{}^{\tau(N)}t_1, {}^{\tau(N)}t_2}{\rightarrow}_{(u+b+a)} ({}^{\tau(N)}t_1{}^{\bullet}, {}^{\tau(N)}t_2{}^{\bullet})$.

- case ii: In N', $[{}^{N'}p_1, {}^{N'}p_2] \overset{{}^{N'}t_2}{\rightarrow}_{(u+b+a)} {}^{N'}t_2{}^{\bullet}$, and $({}^{\tau(N)}t_1{}^{\bullet}, {}^{\tau(N)}t_2{}^{\bullet}) B_{(u+b+a)} {}^{N'}t_2{}^{\bullet}$ is held.

(2) In N', $[{}^{N'}p_1, {}^{N'}p_2] \overset{{}^{N'}t_2}{\rightarrow}_{(u+b+a)} {}^{N'}t_2{}^{\bullet}$.

- case ii: In $\tau(N)$, $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{{}^{\tau(N)}t_1, {}^{\tau(N)}t_2}{\rightarrow}_{(u+b+a)} ({}^{\tau(N)}t_1{}^{\bullet}, {}^{\tau(N)}t_2{}^{\bullet})$, and $({}^{\tau(N)}t_1{}^{\bullet}, {}^{\tau(N)}t_2{}^{\bullet}) B_{(u+b+a)} {}^{N'}t_2{}^{\bullet}$ is held.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([{}^{\tau(N)}p_1, {}^{\tau(N)}p_2], u+b+a)$. In N', $\mathcal{U}([{}^{N'}p_1, {}^{N'}p_2], u+b+a)$ and $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] B_u [{}^{N'}p_1, {}^{N'}p_2]$ holds.

(6) In N', $\mathcal{U}([{}^{N'}p_1, {}^{N'}p_2], u+b+a)$. In $\tau(N)$, $\mathcal{U}([{}^{\tau(N)}p_1, {}^{\tau(N)}p_2], u+b+a)$ and $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] B_u [{}^{N'}p_1, {}^{N'}p_2]$ holds.

This holds for all u, N and N' are timed branching bisimilar. Then N' is a subclass of N with respect to timed projection inheritance. **Q.E.D.**

(RPP) : Let $\tau(N)$ be a safe LTPN which derived from a LTPN N by applying abstraction operator. Of course, $\tau(N)$ is a subclass of N with respect to timed projection inheritance. Let u be arrival time to ${}^{\tau(N)\bullet}t_1$. It is obvious that a binary relation B_u which ${}^{\tau(N)\bullet}t_1 B_u {}^{N'\bullet}t_1$ holds and a binary relation $B_{(u+a+b+\cdots+x+z)}$ which ${}^{\tau(N)}t_z{}^{\bullet} B_{(u+b+\cdots+x+z)} {}^{N'}t_z{}^{\bullet}$ holds are TBBs. First, we prove that a binary relation $B_{(u+a)}$ which $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] B_{(u+a)} [{}^{N'}p_1]$ holds is a TBB.

- Case $b < c$

(1) In $\tau(N)$, $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{{}^{\tau(N)}t_3}{\rightarrow}_{(u+a+c)} ([{}^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet})$.

- case i: $act({}^{\tau(N)}t_3) = \tau$ and $([{}^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet}) B_{(u+a+c)} [{}^{N'}p_1]$.

(2) In N', $[{}^{N'}p_1] \overset{{}^{N'}t_2}{\rightarrow}_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$.

- case ii: In $\tau(N)$, $[{}^{\tau(N)}p_1, {}^{\tau(N)}p_2] \Rightarrow_{(u+a+c)} ([{}^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet}) \overset{{}^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} ({}^{\tau(N)}t_2{}^{\bullet}, {}^{\tau(N)}t_3{}^{\bullet})$, and $({}^{\tau(N)}t_2{}^{\bullet}, {}^{\tau(N)}t_3{}^{\bullet}) B_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$ is held.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1, {}^{\tau(N)}p_2], u{+}a{+}c)$. In N', $\mathcal{U}([^{N'}p_1], u{+}a{+}c)$ and $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_{(u+a+c)}[^{N'}p_1]$ holds.

(6) In N', $\mathcal{U}([^{N'}p_1], u{+}a{+}b)$. In $\tau(N)$, $[^{\tau(N)}p_1, {}^{\tau(N)}p_2] \Rightarrow_{(u+a+c)} ([^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet})$ and $\mathcal{U}(([^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet}), u + a + b)$, and $([^{\tau(N)}p_1], {}^{\tau(N)}t_3{}^{\bullet})B_{(u+a+b)}[^{N'}p_1]$ is held.

- Case $b > c$

(1) In $\tau(N)$, $[^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{{}^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} ({}^{\tau(N)}t_2{}^{\bullet}, [^{\tau(N)}p_2])$.

  – case ii: In N', $[^{N'}p_1] \overset{{}^{N'}t_2}{\rightarrow}_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$ and $({}^{\tau(N)}t_2{}^{\bullet}, [^{\tau(N)}p_2])\ B_{(u+a+b)}{}^{N'}t_2{}^{\bullet}$ holds.

(2) In N', $[^{N'}p_1] \overset{{}^{N'}t_2}{\rightarrow}_{(u+a+b)} {}^{N'}t_2{}^{\bullet}$.

  – case ii: In $\tau(N)$, $[^{\tau(N)}p_1, {}^{\tau(N)}p_2] \overset{{}^{\tau(N)}t_2}{\rightarrow}_{(u+a+b)} ({}^{\tau(N)}t_2{}^{\bullet}, [^{\tau(N)}p_2])$, and $({}^{\tau(N)}t_2{}^{\bullet}, [^{\tau(N)}p_2])\ B_{(u+a+b)}{}^{N'}t_2{}^{\bullet}$ is held.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1, {}^{\tau(N)}p_2], u{+}a{+}b)$. In N', $\mathcal{U}([^{N'}p_1], u{+}a{+}b)$ and $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_{(u+a+b)}[^{N'}p_1]$ holds.

(6) In N', $\mathcal{U}([^{N'}p_1], u{+}a{+}b)$. In $\tau(N)$, $\mathcal{U}([^{\tau(N)}p_1, {}^{\tau(N)}p_2], u{+}a{+}b)$ and $[^{\tau(N)}p_1, {}^{\tau(N)}p_2]B_{(u+a+b)}[^{N'}p_1]$ holds.

Binary relations between the other states of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ can be proved as above.

In the last, we prove a binary relation $B_{(u+a+b+\cdots+x)}$ which $[^{\tau(N)}p_x, {}^{\tau(N)}p_y]B_{(u+a+b+\cdots+x)}[^{N'}p_x]$ holds is a TBB.

(1) In $\tau(N)$, $[^{\tau(N)}p_x, {}^{\tau(N)}p_y] \overset{{}^{\tau(N)}t_z}{\rightarrow}_{(u+a+b+\cdots+x+z)} {}^{\tau(N)}t_z{}^{\bullet}$.

  – case ii: In N', $[^{N'}p_x] \overset{{}^{N'}t_z}{\rightarrow}_{(u+a+b+\cdots+x+z)} {}^{N'}t_z{}^{\bullet}$ and ${}^{\tau(N)}t_z{}^{\bullet}\ B_{(u+a+b+\cdots+x+z)}{}^{N'}t_z{}^{\bullet}$ holds.

(2) In N', $[^{N'}p_x] \overset{{}^{N'}t_z}{\rightarrow}_{(u+a+b+\cdots+x+z)} {}^{N'}t_z{}^{\bullet}$.

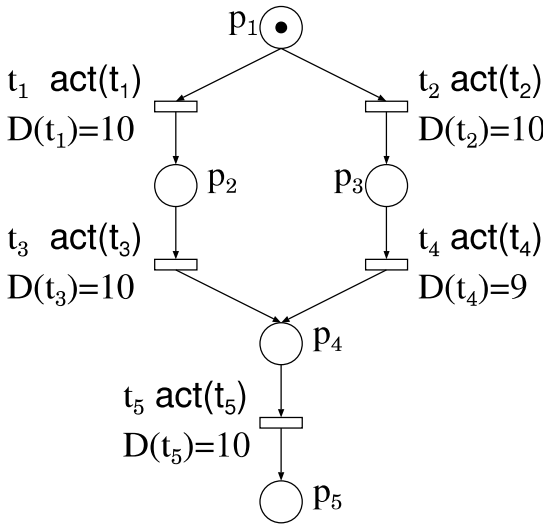


Figure 6.4: A LTPN N_3 which have select structure

- case ii: In $\tau(N)$, $[{}^{\tau(N)}p_x, {}^{\tau(N)}p_y] \overset{{}^{\tau(N)}t_z}{\rightarrow}_{(u+a+b+\cdots+x+z)} {}^{\tau(N)}t_z{}^{\bullet}$, and ${}^{\tau(N)}t_z{}^{\bullet}\ B_{(u+a+b+\cdots+x+z)}\ {}^{N'}t_z{}^{\bullet}$ holds.

(3) Not applicable.

(4) Not applicable.

(5) In $\tau(N)$, $\mathcal{U}([{}^{\tau(N)}p_x, {}^{\tau(N)}p_y], u+a+b+\cdots+x+z)$. In N', $\mathcal{U}([{}^{N'}p_x], u+a+b+\cdots+x+z)$ and $[{}^{\tau(N)}p_x, {}^{\tau(N)}p_y]B_{(u+a+b+\cdots+x+z)}[{}^{N'}p_x]$ hold.

(6) In N', $\mathcal{U}([{}^{N'}p_x], u+a+b+\cdots+x+z)$. In $\tau(N)$, $\mathcal{U}([{}^{\tau(N)}p_x, {}^{\tau(N)}p_y], u+a+b+\cdots+x+z)$ and $[{}^{\tau(N)}p_x, {}^{\tau(N)}p_y]B_{(u+a+b+\cdots+x+z)}[{}^{N'}p_x]$ holds.

This holds for all u, N and N' are timed branching bisimilar. Then N' is a subclass of N with respect to timed projection inheritance. □

We proposed the five reduction operators for safe LTPNs and prove that five operators maintain timed behavioral inheritance. However, we currently have no reduction operator which treats place split and/or place join. Because if we apply reduction operators for these structures, the firing schedule may be changed. Let us consider two LTPN N_3 in Fig. 6.4 and N_3' in Fig. 6.5. The schedule of t_5 is if t_1 fires, then t_5 fires in 30 time unit, else if t_2 fires, then t_5 fires 29 timed unit. For example, we reduce the path $\langle t_2 p_3 t_4 \rangle$, the firing schedule of t_5 is always 30 time unit from the initial state. Therefore, we assume that it is very difficult to treat place split and place join.

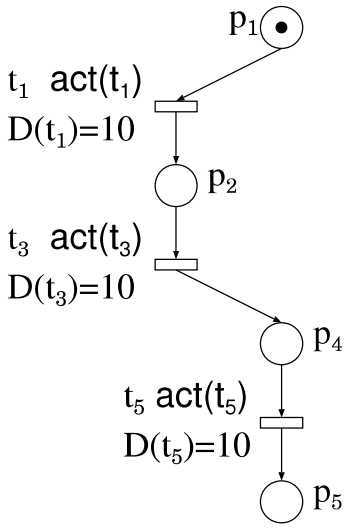


Figure 6.5: A LTPN N_3' which obtained by reducing path $t_2p_3t_4$ from N

6.4 Application Example of the Proposed Operators

In this section, we apply our reduction operators to an artificial safe LTPN and evaluate the state size of the LTPN. The sample of safe LTPN N_4 is Fig. 6.6. Let α be the actions we observe. We focus on the leftmost sub LTPN enclosed by broken line. Firstly, we remove t_4 by operator LF and p_3, p_5 by operator FPP. Next, we remove t_2, p_4 by operator FST. The leftmost sub LTPN is reduced to the sub LTPN by the right side. We can compute size of state space by method proposed by Nakano et al.[44]. Fig. 6.7 shows the number of the state of the safe LTPN. X-axis is the number of sub LTPN above. We can see the state explosion without our reduction operators. Our operators can suppress the state explosion as far as the experiment.

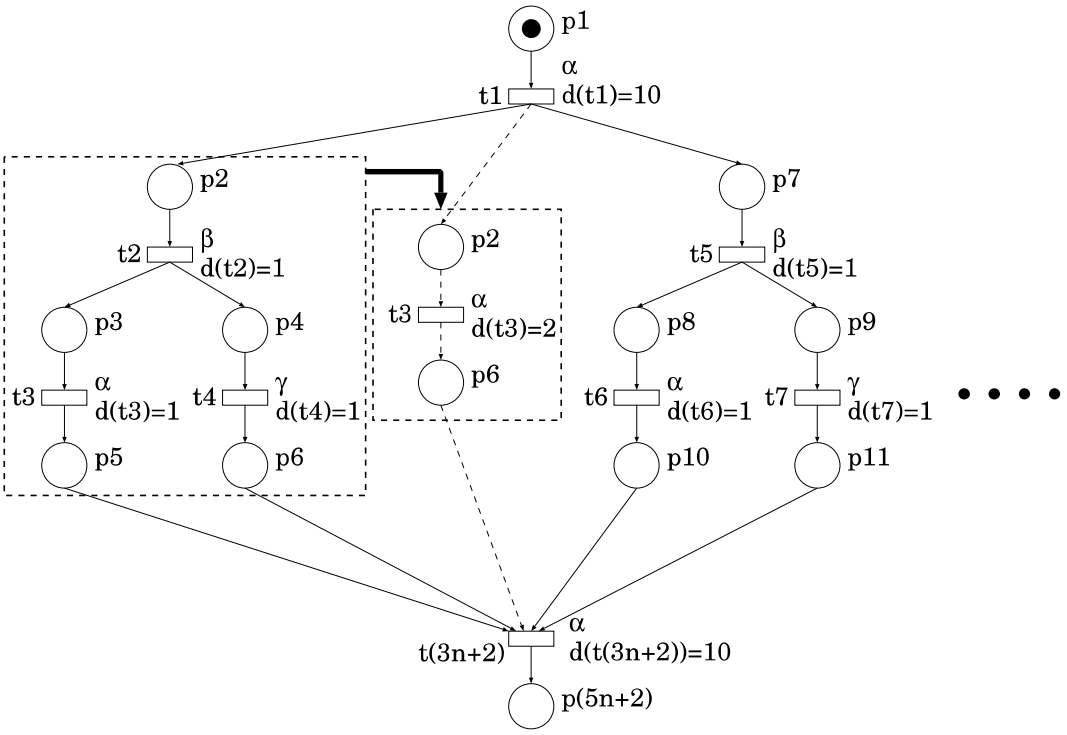


Figure 6.6: A LTPN N_4

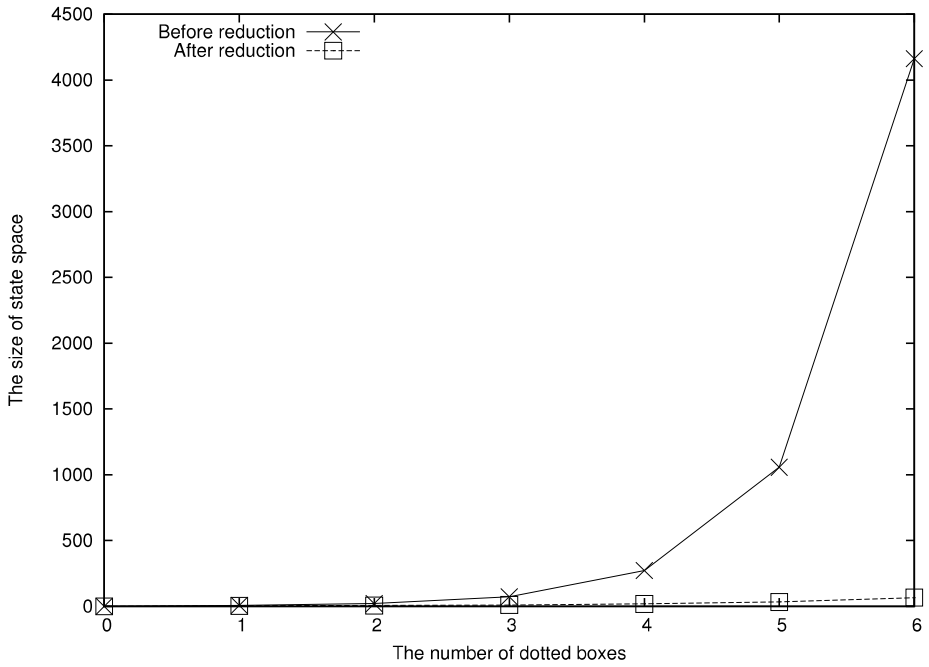


Figure 6.7: Trends in the state space of N_4

6.5 Remarks

In this chapter, we have proposed reduction operators to refactor timed system. We proposed the concept of timed projection inheritance. This concept means the equivalence of the systems in consideration of observability and there is no change of firing schedule of observed actions. Then, we have proposed five reduction operators based on the concept for timed Petri nets which are safe . We have proved that these reduction operators preserve timed projection inheritance.

Next, we have applied our proposed operators for an artificial safe timed Petri net. The results show that by considering observability and reducing transitions which users need not to observe, net size can be made smaller.

As future works, we plan to extend our reduction operators preserving timed protocol inheritance and timed lifecycle inheritance, and to develop a tool which can apply reduction operators automatically.

Chapter 7

Conclusion

In this thesis, we have proposed reduction technique, called "refactoring", for business workflow to check its various consistency easily. Furthermore, we have presented its application.

It is inevitable that workflows become complex, since business rules changing or business combination force their workflows to change and coalesce with other one, against designers' intent. Our approach reduces the complexity by mathematical model and related algorithm.

In Chap. 3, we have presented theorems and proofs about three refactorizability sufficient conditions. Firstly, in the no PT-handle case, we can remove EFC-structures from the EFC WF-net by using the rule ϕ_{Best}. There is a branching bisimilarity between the resultant net and the original net. We can refactor the original net to WS WF-net by using ϕ_{Best} repeatedly. Secondly, in the no TP-handle case, we can remove TP-cross structures from the EFC WF-net by using the rule ϕ_{Desel}. There is also branching bisimilarity between the resultant net and the original net. We can refactor the original net to WS WF-net by using ϕ_{Desel} repeatedly. Thirdly, in the case that there are some redundant places in the EFC-net, we can remove implicit places from the EFC WF-net by using the rule $\phi_{Implicit}$. There is also branching bisimilarity between the resultant net and the original net. We can refactor the original net to WS WF-net by using $\phi_{Implicit}$ repeatedly.

There are some Petri net reduction techniques discussed in Chap. 1. They preserve one or more properties (i.e. partial behavior). Branching Bisimilarity our refactoring algorithm preserve is the wider concept than properties the reduction techniques preserve. Because Branching Bisimilarity corresponds to overalls of the system behavior [1]. There are two novel aspects of this thesis. First, we define the result of refactoring as WS WF-net. WS WF-net has more analysis methods. Second, we give the concept of refactorizability and the checking algorithms. The concept helps us to design concrete refactoring algorithm. The aspects have not been discussed

[1]Further mathematical studies are needed in order to clarify the relation between Polyvyanyy's reduction and our refactoring rules.

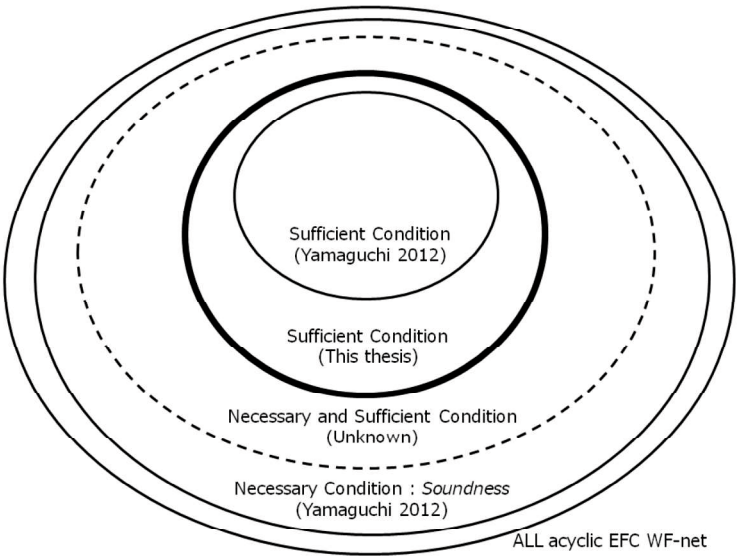


Figure 7.1: The relations between the sets correspond to own refactorizability conditions.

in the other reduction techniques

In Chap. 4, we have presented algorithm composed from above three rules. We have checked the three rules' pre-conditions and post-conditions precisely. The three rules are ordered to deal with a broader subclass of EFC WF-net. The algorithm liberate us from checking refactoring conditions manually. We have to only input the EFC WF-net to obtain a WS WF-net which is branching bisimilar with the original net. The algorithm is a novelty result, since it has not proposed yet.

In Chap. 5, we have presented three applications of refactoring from EFC WF-nets to WS WF-nets. The three are "Reachability Checking", "State Number Calculation" and "Response Property Checking". They may cut off the cost of upstream design process in system development.

In Chap. 6, we have proposed an introduce advanced concept about timed behavior and proposed refactoring rules based on it. It may be a useful concept for refactoring WF-net with real-time constraints.

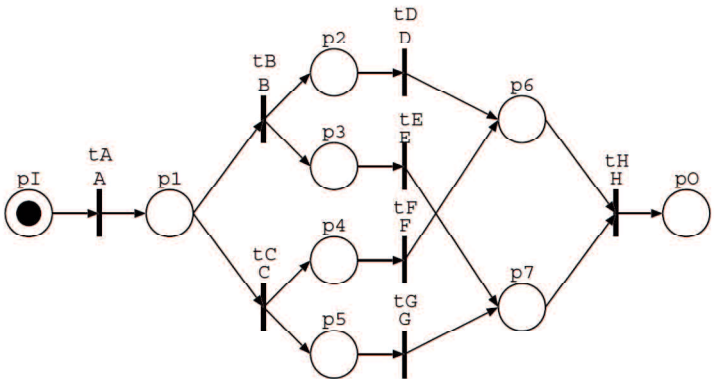
The future subjects in research field related to this thesis are as follows: The first subject relates to the discovery of the necessary and sufficient condition about refactorizability of EFC WF-net. In this thesis, we gave new sufficient condition(s) of refactorizability which are superior to Yamaguchi's condition[18]. The Venn diagram shown in Figure. 7.1 represents the relations between the sets correspond to own refactorizability conditions. We think that the three are

proper subset of the necessary and sufficient condition. Since we have a sample FC WF-net shown in Fig. 7.2 [45] which is refactorizable but does not satisfy our conditions.

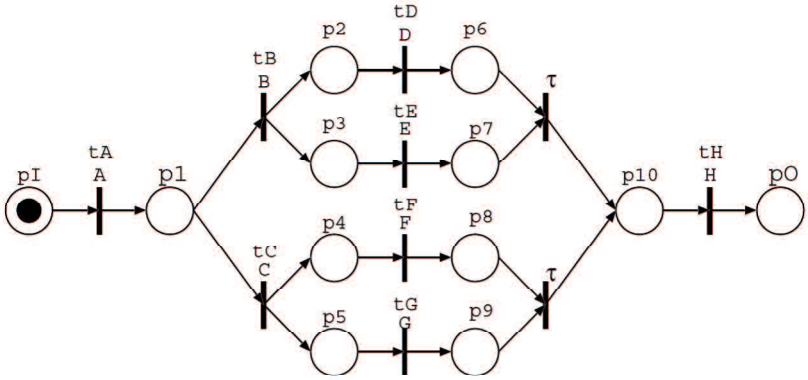
Furthermore, we have an another sample WF-net shown in Fig. 7.3 [45] which may be not refactorizable. We will survey the net to clarify the necessary and sufficient condition [2].

The second subject relates to the building the process in system development clearly, to apply our WF-net refactoring algorithm. In this thesis, we have presented some refactoring applications. It is indispensable to clarify the process for the real world advantages.

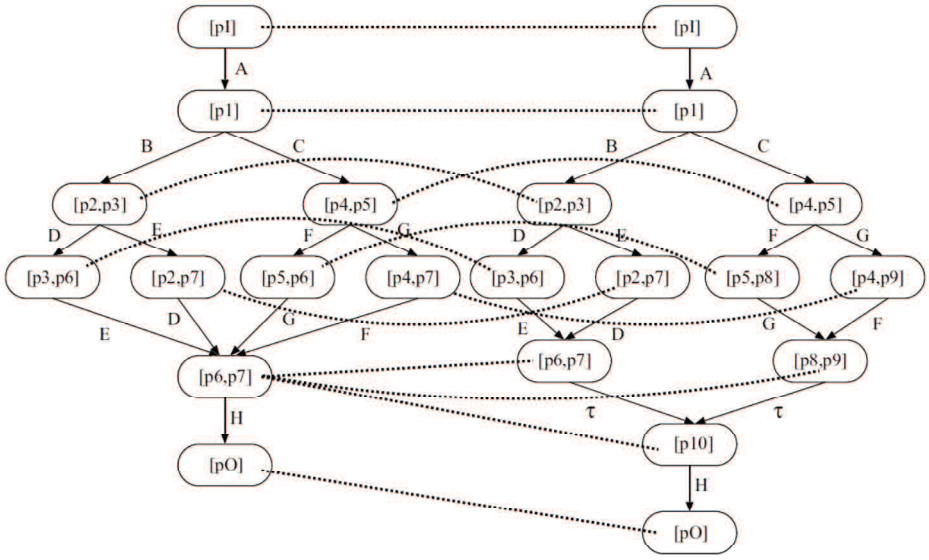
[2]Unfortunately, checking the necessary and sufficient conditions seem has no polynomial time algorithm. Because the problem equals to constructing WS WF-net from state graph. It may be intractable [46][47] or more difficult problem.



(a) An original net $N_{refactorizable}$.



(b) A new WF-net $N_{refactored}$ which is refactored from $N_{refactorizable}$.



(c) Branching bisimulation between the reachability graph of $(N_{refactorizable}, [p_I])$ and $(N_{refactored}, [p_I])$.

Figure 7.2: A sample of refactorizable WF-net [45] which does not satisfy the three sufficient conditions in this thesis.

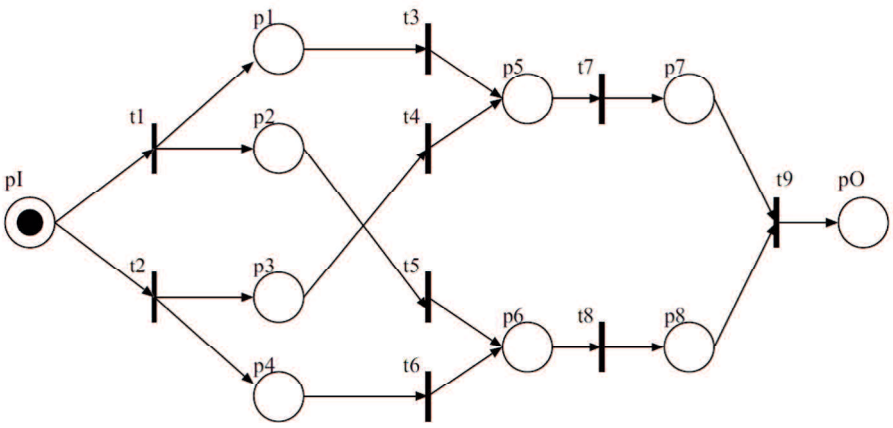


Figure 7.3: A sample WF-net [45] which may be not refactorizable.

Bibliography

[1] WfMC, *Terminology & Glossary*, WfMC–TC–1011, 1999.

[2] WfMC, *Interface 4 – Interoperability – Abstract Specification*, WfMC–TC–1012, 1996.

[3] S. Kumagai, "Actiivities on net theory in Japan," IEICE Trans. Fundamentals, vol.E77-A, no.7, pp.1125–1131, 1994.

[4] J.L. Peterson, *Petri Net Theory and the Modeling of System*, Prentice Hall, 1981.

[5] C. Cassandras, S. Laforture, *Intruduction to Discrete Event System*, Springer Verlag, 2007.

[6] W.M.P. van der Aalst, "Making work flow:On the application of Petri nats to business process management," Lecture Notes in Computer Science, vol.2360, pp.1–22, 2002.

[7] D. Hauschildt, H.M.W. Verbeek, W.M.P. van der Aalst, "WOFLAN: a Petri-net-based workflow analyzer," Computing Science Report, vol.97, no.12, Eindhoven University of Technology, 1997.

[8] W.M.P. van der Aalst and T. Basten, "Inheritance of workflows:An approach to tackling problems related to change," J. Circuits, Systems, Computers, vol.8, no.1, pp.125–203, 2002.

[9] W.M.P. van der Aalst, "The application of Petri nets to workflow management," J. Circuits, Systems, Computers, vol.8, no.1, pp.21–65, 1998.

[10] S. Yamaguchi, Q.W. Ge, "Performance evaluation on change time of dynamic workflow change," IEICE Trans. Fundamentals, vol.E83-A, no.11, pp.2177–2187, 2000.

[11] S. Yamaguchi, Y. Shiode, Q.W. Ge, and M. Tanaka, "Performance evaluation on transient time of dynamic workflow change," IEICE Trans. Fundamentals, vol.E84-A, no.11, pp.2852–2864, 2001.

[12] S. Yamaguchi, A. Mishima, Q.W. Ge, and M. Tanaka, “A flexible and efficient workflow change type:Selective shift” IEICE Trans. Fundamentals, vol.E88-A, no.6, pp.1487–1496, 2005.

[13] E. Best and M.W. Shields, “Some equivalence results for free choice nets and simple nets and on the periodicity of live free choice nets,” Lecture Notes in Computer Science vol.159, pp.141–154, 1983.

[14] E. Best, J. Desel, and J. Esparza, “Traps characterize home states in free choice systems,“ Theoretical Computer Science, vol.101, issue 2, pp.161–176, 1992.

[15] J. Desel, J. Esparza, *Free Choice Petri Nets*, Cambridge University Press, 2005.

[16] T. Murata, “Petri nets: properties, analysis and applications,” Proc. IEEE, vol.77, no.4, pp.541–580, 1989.

[17] A. Polyvyanyy, L. Garcia-Banuelos, D. Fahland, and M.Weske, “Maximal structuring of acyclic process models,” The Computer Journal, vol.57, no.1, pp.12–35, 2014.

[18] S. Yamaguchi, “Refactoring problem of acyclic extended free-choice workflow nets to acyclic well-structured workflow nets,” IEICE Trans. Information and Systems, vol.E95-D, no.5, pp.1375–1379, 2012.

[19] S. Yamaguchi, T. Hirakawa, “Polynomial time verification of protocol inheritance between acyclic extended free-choice workflow nets and their subnets ,” IEICE Trans. Fudamentals, vol.E96-A, no.2, pp.505–513, 2013.

[20] M. Fowler, *Refactoring : Improving the Design of Existing Code*, Addison Wesley, 1999.

[21] I. Toyoshima, S. Yamaguchi and Y. Murakami, “Two sufficient conditions on refactorizability of acyclic extended free choice workflow nets to acyclic well-structured workflow nets and their application,” IEICE Trans. Fundamentals, vol.E98-A, no.2, pp.635–644, 2015.

[22] I. Toyoshima, S. Yamaguchi, J. Zhang, “A Refactoring Algorithm of Workflows based on Petri Nets,” Proc.of The 4th International Congress on Advanced Applied Informatics (AAI), pp.79–84, July, 2015.

[23] I. Toyoshima, S. Nakano, and S. Yamaguchi, “Reduction operators based on behavioral inheritance for timed petri nets,” IEICE Trans. Fundamentals, vol.E97-A, no.2, pp.484–489, 2015.

[24] J. Esparza, and M. Silva, "Circuits, handles, bridges and nets," Lecture Notes in Computer Science, vol.483, pp.210–242, 1990.

[25] K. van Hee, N. Sidorova, and M. Voorhoeve, "Soundness and separability of stepwise refinement approach," Lecture Notes in Computer Science, vol.2679, pp.337–356, 2003.

[26] Y. Murakami, I. Toyoshima, S. Yamaguchi, "Sufficient condition on refactorizability of acyclic extended free choice workflow nets to acyclic well-structured workflow Nets," Proc. of First International Symposium on Computing and Networking, pp.592–596, 2013.

[27] S. Yamaguchi, M. Yamaguchi, and M. Tanaka, "A model checkin method of soundness for acyclic workflow nets using the SPIN model checker" Int. J. Information, vol.12, no.1, pp.163–172, 2009.

[28] M. Yamaguchi, S. Yamaguchi, and M. Tanaka, "A model checkin method of soundness for workflow nets, " IEICE Trans. Fundamentals, vol.E92-A, no.11, pp.2723–2731, 2009.

[29] W.M.P. van der Aalst, K.M. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, 2002.

[30] H.M.W. Verbeek and T. Basten, "Deciding life-cycle inheritance on Petri nets," Proc. of ICATPN 2003, pp.44–63, 2003.

[31] H.M.W. Verbeek and T. Basten, "Life-cycle inheritance: A Petri-net-based approach, " Lecture Notes in Computer Science, vol.1248, pp.62–81, 1997.

[32] S. Yamaguchi, "Polynomial time verification of reachability in sound extended free-choice workflow nets," IEICE Trans. Fundamentals, vol.E97-A, no.2, pp.468–475, 2014.

[33] G.J. Holtzman, *The Model Checker SPIN*, Addison Wesley, 2003.

[34] SPIN model checker, http://spinroot.com/ .

[35] J. Zhang, I. Toyoshima, S. Yamaguchi, "On Implicit Place and Its Application to Refactoring in Acyclic Extended Free Choice Workflow Nets," Proc.of ITC-CSCC 2015, pp.264–267, 2015.

[36] M.A. Bin Ahmadon, S. Yamaguchi, "Convertibility and conversion algorithm of well-structured workflow net to process tree," Proc. CANDAR 2013, pp.122–127, 2013.

[37] M. A. Bin Ahmadon and S. Yamaguchi, "State number calculation problem of workflow nets," IEICE Trans. Information and Systems, vol.E98-D, no.6, pp.1128–1136, 2015.

[38] S. Yamaguchi, "The state number calculation problem for free-choice workflow nets is intractable," Proc. ITC-CSCC 2014, pp.838–841, 2014.

[39] T. Susaki, S. Yamaguchi, "On Process Tree Based Calculation of the Number of States in Petri Nets," Proc. ITC-CSCC 2013, pp.84–87, 2013.7.

[40] R. Alur and D.L. Dill, "A theory of timed automata," Theoretical Computer Science, vol.126, pp.183–235, 1994.

[41] J. Wang, *Timed Petri Nets: Theory and Application*, Kluwer Academic Publishers, ISBN: 0-7923-8270-6, 1998.

[42] R.H. Sloan, U. Buy, "Reduction rules for time petri nets," Acta Informatica, vol. 3, pp. 687–706, 1996.

[43] W. Fokkink, J. Pang, A. Wijs, "Is timed branching bisimilarity a congruence indeed?", Fundamenta Informaticae, vol.87, issue 3–4, pp.287–311, 2008.

[44] S. Nakano, S. Yamaguchi, "An efficient translation method from timed Petri nets to timed automata", IEICE Trans. Fundamentals, vol. E95-A, no. 8, pp. 1402–1411, 2012.

[45] Y. Kuroda, S. Yamaguchi, and M. Tanaka, "A proposal of refactoring to WS workflow nets," Proc.of ITC-CSCC 2009, pp.875–878, 2009.

[46] M. Sipser, *Intruduction to the Theory of Computation*, Course Technology Inc., 2005.

[47] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison Wesley, 1979.