Title	Design and Evaluation of Efficient Algorithms for Feature Interaction Detection in Telecommunication Services
Author(s)	中村, 匡秀
Citation	
Issue Date	
Text Version	ETD
URL	https://doi.org/10.11501/3155467
DOI	10.11501/3155467
rights	
Note	

Design and Evaluation of Efficient Algorithms for Feature Interaction Detection in Telecommunication Services

Masahide Nakamura

January 1999

Design and Evaluation of Efficient Algorithms for Feature Interaction Detection in Telecommunication Services

Masahide Nakamura

January 1999 Dissertation submitted to the Graduate School of Engineering Science of

Osaka University in partial fulfilment of the requirements

for the degree of Doctor of Engineering

Abstract

Feature interaction is a kind of inconsistent conflict between multiple telecommunication services, which was never expected from the single services' behavior. In practical service development, the analysis of interactions has been conducted in an ad hoc manner by subject matter experts. This leads to time-consuming service design and testing without any interaction-free guarantee.

In order to tackle this problem, we discuss the formal approach for *detection* of feature interaction. The detection process exists to check if interactions occur or not between given multiple services, which is one of the most fundamental and important steps in feature interaction analysis.

In general, the formal method for interaction detection consists of two parts: formulation and algorithm. Formulation includes the definition of service specifications and the definition of interactions to be detected, while the algorithm determines concrete procedures for the detection based on the formulation.

A number of formal methods for the feature interaction detection have been proposed so far. Most of the these methods adopt specification methods fundamentally based on the state transition model, in which a state consisting of users' local states successively moves to a next state in response to the occurrence of user's events. Then, the interactions are defined by some undesirable conditions which hold on a certain state within the model.

Unfortunately, most previous research has primarily focused on the formulation using their respective frameworks, and developing concrete algorithms for efficient interaction

detection remains an open issue. Although there are a few exceptions, most conventional frameworks adopt an exhaustive search method, which explores all possible states using a *reachability graph*, in order to identify states at which interactions occur.

The interaction detection algorithm based on the exhaustive search principle is fundamentally powerful, in the sense that all interactions are exactly detected. However, when analysing complex and large-scale services, it suffers from the *state explosion problem*. That is, the number of states in the model grows exponentially according to the numbers of users and features analysed. Hence, the cost of interaction detection becomes too expensive, which results in a crucial limitation on its practical application.

The primary goal of this dissertation is to address the problem of how to reduce such an expensive cost for practical interaction detection. For this purpose, we propose two new algorithms, called Algorithm SYM and Algorithm PINV, for efficient feature interaction detection.

The first, Algorithm SYM exploits an equivalence relation, called *permutation symmetry*, for the reduction of the reachability graph. In the telecommunication services, there exists a specific constraint that all users of a service must be able to use the same functionality of that service. Under this constraint, the permutation symmetry works as follows. If we know the possible behavior of user A, then we can infer user B's possible behavior, since B can use the service in the same way as A. Hence, we can discard B's information from the reachability graph. By exploiting permutation symmetry for construction of the reachability graph, we succeed in reducing the size of the graph while preserving the necessary information for the interaction detection. As a result, we can exactly identify all interactions with smaller state space and time.

The second, Algorithm PINV takes completely different approach from that of Algorithm SYM. This algorithm is categorised as a *static* algorithm in the sense that it does not explore possible states by means of a reachability graph. Instead of a graph, Algo-

rithm PINV extensively utilizes a *P-invariant method*, in order to check the reachability of states. First, we determine the *candidate states* at which the interaction may occur. Then we check if each candidate is actually reachable from the initial state by means of the P-invariant method. This check can only be performed within the structure of the service specification. Hence, an expensive state search using a reachability graph is no longer necessary and therefore, drastic cost reduction is expected. Due to the nature of static algorithms however, Algorithm PINV has a drawback. Since the P-invariant method's success is based on a necessary condition of reachability, Algorithm PINV may not always attain the optimal detection quality.

We have also conducted experimental evaluation through an application to practical services. We evaluate both Algorithms SYM and PINV from the following viewpoints: detection quality, performance and scalability.

The results show that Algorithm SYM achieves the optimal detection quality as expected, and about 80% reduction in space and time for practical interaction detection problems with three users. Also, it has good scalability with respect to the number of users. As for Algorithm PINV, it is shown that the detection quality is semi-optimal, comparable to optimal, and that both performance and scalability are significantly improved by several orders of magnitude.

This dissertation is organised as follows. In Chapter 1, we summarize the background and related topics and describe an outline of the dissertation. In Chapter 2, in addition to some practical examples of feature interactions, we describe the fundamental definitions of rule-based service specification and the state transition model. Then, based on those, we formulate the detection problem. In Chapter 3, we present a conventional detection algorithm. In Chapter 4, we propose a new detection algorithm, SYM. We give a definition of permutation symmetry and present theories to implement Algorithm SYM. In Chapter 5, we propose Algorithm PINV. We first describe a Petri net model and the P-

invariant method. Then, we discuss the candidate generation procedures and Algorithm PINV itself. In Chapter 6, we perform the experimental evaluation of the proposed two algorithms. Through the application to practical services, we evaluate their effectiveness by three metrics: detection quality, performance and scalability. Finally, in Chapter 7, we conclude this dissertation with a summary and future work.

List of Major Publications

(1) Kakuda, Y., Nakamura, M. and Kikuno, T., "Automated synthesis of protocol specifications from service specifications with parallelly executable multiple primitives", *IEICE Trans. Fundamentals*, Vol. E77-A No.10, pp.1634-1645, Oct. 1994.

(2) Nakamura, M., Kakuda, Y. and Kikuno, T., "On constructing communication protocols from component- based service specifications", *Journal of Computer Communications*, Vol. 19, No. 14, pp.1200-1215, Dec. 1996.

(3) Nakamura, M. and Kikuno, T., "A new approach in feature interaction testing", *INTEGRATION, the VLSI journal*, Vol. 26, pp.211-223, Dec. 1998.

(4) Nakamura, M. and Kikuno, T., "Exploiting symmetric relation for efficient feature interaction detection", *IEICE Trans. on Information and Systems.* (submitted)

(5) Nakamura, M., Kakuda, Y. and Kikuno, T., "Protocol synthesis from acyclic formed service specifications," *Proc. of IEEE Int'l. Conf. on Information Networking (ICOIN'94)*, pp.177-182, Dec. 1994.

(6) Nakamura, M., Kakuda, Y. and Kikuno, T., "An integration-oriented approach to designing communication protocols from component-based service specifications," *Proc. of IEEE Int'l. Conf. on Computer Communication (INFOCOM'96)*, pp.1157-1164, Mar. 1996.

(7) Nakamura, M., Kakuda, Y. and Kikuno, T., "Analyzing non-determinism in telecommunication services using P-invariant of Petri net model," *Proc. of IEEE Int'l. Conf. on Computer Communication (INFOCOM'97)*, pp.1253-1260, Apr. 1997.

(8) Nakamura, M., Kakuda, Y. and Kikuno, T., "Petri net based detection method for non-deterministic feature interactions and its experimental evaluation," *Proc. of IEEE Fourth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'97)*, pp.138-152, Jun. 1997.

(9) Nakamura, M., Kakuda, Y. and Kikuno, T., "Feature interaction detection using permutation symmetry", *Proc. of IEEE Fifth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'98)*, pp.187-201, Sep. 1998.

(10) Hatanaka, Y., Nakamura, M., Kakuda, Y. and Kikuno, T., "A synthesis method for fault-tolerant and flexible multipath routing protocols", *Proc. of IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS'97)*, pp.96-105, Sep.1997.

Acknowledgements

During the course of this work, I have been fortunate to have received assistance from many individuals. I would especially like to thank my supervisor Professor Tohru Kikuno for his continuous support, encouragement, and guidance during this work.

I am also very grateful to the members of my thesis review committee: Professor Hideo Miyahara and Professor Katsuro Inoue for their invaluable comments and helpful criticisms of this thesis.

Many of the courses that I have taken during my graduate career have been helpful in preparing this dissertation. I would like to acknowledge the guidance of Professors Toru Fujiwara, Masaharu Imai, Nobuki Tokura, Ken-ichi Taniguchi, Ken-ichi Hagihara, Mamoru Fujii, Toshinobu Kashiwabara and Akihiro Hashimoto.

I also would like to express my gratitude to Professor Yoshiaki Kakuda of Hiroshima City University who I was directly responsible to for four years in Master and Ph.D courses.

I wish to thank Kyle Richardson of the School of Computer Science at Birmingham University for his proofreading of this thesis. His careful comments were invaluble in the completion of this dissertation.

Thanks are also due to many friends in the Department of Informatics and Mathematical Science at Osaka University who gave me many useful comments. Especially, I wish to thank Takuro Ikeda, who gave me invaluable assistance with the experiments and drawing figures. Without his help, I would not have finalized this dissertation.

Contents

Chapter 1

Introduction

1.1 Background

Recent advancement of new telecommunication platforms such as IN (intelligent network) [64] and AIN (advanced intelligent network) [67], enable functional enrichment in telecommunication services. As a result, a lot of new services are being developed and deployed in order to achieve the various requirements of customers.

When such new services are added to the system, functional conflicts can occur between new and existing services, which may even trigger system down. This conflict is generally recognised as *feature interaction*[11], and it becomes a serious obstacle which prevents the rapid development of new services.

In practical service development, the analysis of feature interactions has been conducted in an ad hoc manner by subject matter experts. However, as the number and complexity of services grow, the ad hoc analysis does not work in a feasible way, which leads to time-consuming service design and testing without any interaction-free guarantee.

Therefore, systematic techniques and methodologies for the feature interaction problem are strongly required today. For the purpose of an essential solution to feature interaction, much work has been carried out by researchers and practitioners from academia,

research centres and industry [8, 18, 22, 40]. These works include techniques for various purposes such as detection, resolution, prevention and management of feature interactions (see an excellent survey [35]).

Among those issues, *feature interaction detection* is one of the most fundamental steps towards an essential solution. The detection process exists to identify whether the interaction occurs or not for given multiple services.

One of the promising ways to achieve systematic interaction detection is to exploit *formal methods*, since they allow the choice of a suitable abstraction level and automatic verification with proper tool support. In general, the use of formal methods for interaction detection requires the following two things.

Formulation: Define the notation and model of the target service, which is specifically called *service specification*. Then, define the conditions that identify target interactions to be detected.

Algorithm: Determine a detection algorithm based on the formulated service specification framework.

A number of formal methods for interaction detection have been proposed so far. Most previous research has focused primarily on **Formulation**, that is, they concentrate on how to define the interactions based on their respective specification frameworks. There are a lot of service specification methods utilised (or newly proposed) for interaction detection. For example, Specification and Description Language (SDL)[65, 9, 16], linear time temporal [5, 6], Object Z [42], SMV language [53], PROMELA [30, 43], LOTOS [10, 54, 55], FSM-based method [4, 21, 37, 41, 50]), rule-based method [24, 27, 29, 34, 51, 62]. Although there are a few exceptions, the model behind the specification is generally a state transition model, in which a state consisting of users' local states successively moves to a next state in response to the occurrence of user's events. This is because the

behaviour of services can be naturally captured by the *finite state machine* (FSM) or its extension. In addition, the interactions are often defined by some undesirable conditions (e.g., deadlock) that hold on a certain state within the model.

On the other hand, unfortunately, most conventional methods pay little attention to **Algorithm**. Except for a few methods, concrete detection algorithms remain open issues, and the *exhaustive search* method [31, 59] is utilised for the implementation of the algorithm in general.

The detection algorithm based on exhaustive search (denoted by Algorithm EXH for convenience), enumerates all possible states within the state transition model, and then identifies those states at which an interactions occurs. Algorithm EXH is quite simple, but powerful in the sense that all interactions defined within the state transition model are exactly detected. So, it is adopted by most conventional detection frameworks (e.g., [4, 16, 21, 27, 37, 43, 51, 54]).

However, due to concurrent characteristics of telecommunication services, the number of states in the model grows exponentially in accordance with the numbers of users and features analysed. Therefore, when analysing complex and large-scale services, Algorithm EXH suffers from the so-called *state explosion problem* [60]. Hence, the cost of interaction detection becomes too expensive, which results in a crucial limitation on its practical application. The primary goal of this dissertation is to address the design of efficient algorithms for feature interaction detection, and their evaluation through practical applications.

1.2 Main Results

1.2.1 Design of Algorithms

By using a rule-based service specification which is similar to the conventional ones [24, 27, 29, 34, 51, 62] as a service description, we first formulate the interaction detection problem for four typical classes of interactions: (1) *deadlock*, (2) *loop*, (3) *non-determinism* and (4) *violation of invariants* [21, 24, 37, 27, 51]. Then, we propose two new algorithms, called Algorithm SYM and Algorithm PINV for detection of these four kinds of interactions.

The conventional algorithm, EXH, utilises a *reachability graph* [31, 59] in an exhaustive search, in order to explore all possible states within the state transition model. When the numbers of users and features increase, the size of the reachability graph grows exponentially. This is the main source of the state explosion problem.

The aim of Algorithm SYM is to reduce the size of the reachability graph in a certain manner. In order to achieve the reduction, we utilise an equivalence relation called *permutation symmetry* with respect to users. The reduction works successfully due to a specific constraint in telecommunication systems which states that "all subscribers of a service X are guaranteed to be able to use the same functionality of X". Under this constraint, suppose that both users A and B are subscribers of X. Intuitively speaking, if we know subscriber A's possible behavior on service X, then we then we can infer B's behavior on X from A's by swapping A and B, because B can use X in the same way as A. Therefore, we no longer have to store the information about B's behavior. As a result, we can discard the relevant state transitions from the reachability graph.

The proposed theory extends this idea to possible permutation of users, and defines an equivalence relation among states (and transitions), called permutation symmetry. By extensively exploiting permutation symmetry for the generation of a reachability graph, we can reduce the size of the graph while completely preserving all necessary information

for interaction detection. As a result, Algorithm SYM can achieve exact interaction detection based on necessary and sufficient conditions, with a reachability graph much smaller than that of Algorithm EXH.

Algorithm PINV takes a completely different approach from algorithms EXH and SYM. This algorithm is a so-called *static* algorithm [32], in the sense that it does not require reachable state exploration using a reachability graph (Algorithms EXH and SYM are called *dynamic* on the other hand). Instead of the reachability graph, Algorithm PINV exploits the *P-invariant method* of Petri nets [48, 32, 33] in order to check the reachability of states. We first generate *candidate states* at which interactions may occur, and then check the reachability of each candidate by means of the P-invariant method. Since this check can only be performed within the structure of the service specification, PINV works in a static way.

Due to the nature of static algorithms, Algorithm PINV is applied to the detection of only two classes of interactions: non-determinism and violation of invariants. Also, P-invariant method is based only on the necessary condition of reachability. This fact implies that Algorithm PINV may not always attain the optimal detection quality. In other words, Algorithm PINV may detect some interactions which do not actually occur. However, incorporating the above drawbacks, Algorithm PINV achieves drastic cost reduction of interaction detection since the expensive reachability graph is no longer necessary.

1.2.2 Evaluation of Algorithms

Also, we conduct the experimental evaluation of the proposed algorithms PINV and SYM through application to interaction detection within practical standard services [64, 66].

In the experiment, we evaluate algorithms SYM and PINV from the following viewpoint.

Detection Quality: whether the algorithms can exactly identify all interactions or not.

Performance: how much time and space are needed for the algorithms.

Scalability: how many users and features can be scaled by the algorithms.

The results show that Algorithm SYM achieves the optimal detection quality as expected, and about 80% reduction in space and time for practical interaction detection with three users. Also, it has a good scalability with respect to the number of users.

As for Algorithm PINV, the detection quality is semi-optimal in comparison, in that non-actual interaction is never identified. This means that the necessary condition of the P-invariant method, fundamentally worked as a necessary sufficient condition for practical services prepared in the experiment. Also, it is shown that both performance and scalability are significantly improved by several orders of magnitude.

As a result, it is shown that both methods are well applicable to practical interaction detection.

1.3 Overview of the Dissertation

The dissertation is organized as follows: In Chapter 2, after some practical examples of feature interactions, we describe the fundamental definitions of rule-based service specification and the state transition model. Then, based on those, we formulate the detection problem for four classes of interactions: deadlock, loop, non-determinism and violation of invariants.

In Chapter 3, we present the conventional detection algorithm, EXH. In the rule-based service specification, Algorithm EXH utilises a reachability graph called FRG. Hence, we first define FRG, then Algorithm EXH is presented.

In Chapter 4, we propose a new detection algorithm SYM. We give a definition of permutation symmetry after explanation of the key idea. Next, we define a new reachability graph called SRG by means of permutation symmetry and give the proof rules

for interaction detection. By means of SRG and these proof rules, we propose Algorithm SYM.

In Chapter 5, we propose Algorithm PINV. First, we define a Petri net model onto which the rule-based service specification is mapped. We then explain the P-invariant method of the Petri net. Next, we show the candidate generation procedures for interactions of non-determinism and violation of invariants. Combining these procedures with the P-invariant method, we present Algorithm PINV.

In Chapter 6, we perform experimental evaluation of the proposed two algorithms. Through the application to practical services, we show their effectiveness by three metrics: detection quality, performance and scalability.

Finally, in Chapter 7, we conclude this dissertation with a summary and future works.

Chapter 2

Preliminary

2.1 Practical Examples

Before explaining fundamental definitions, we present three practical examples of feature interactions. More instances can be referred to [13, 14, 25, 57, 61, 63].

Example 2.1 Interaction between Call Waiting and Call Forwarding.

Call Waiting (CW): This service allows the subscriber to receive an additional call while he is talking. Suppose that x subscribes to CW. Even when x is busy talking with y, x can receive an additional call from a third party, z.

Call Forwarding (CF): This service allows the subscriber to have his incoming calls forwarded to another number. Suppose that x subscribes to CF, and that x specifies z to be a forwarding address. Then, any incoming call to x is automatically forwarded to z.

Interaction $CW\&CF$: Let A, B, C and D be subscribers of the telephone network. Assume that A subscribes to both CW and CF. Suppose that (1) A is talking with B, (2) C is ready to idle, and (3) D is in A's forwarding address and is idle. Then, if C dials A, should the call from C to A be received by A's CW feature, or should it

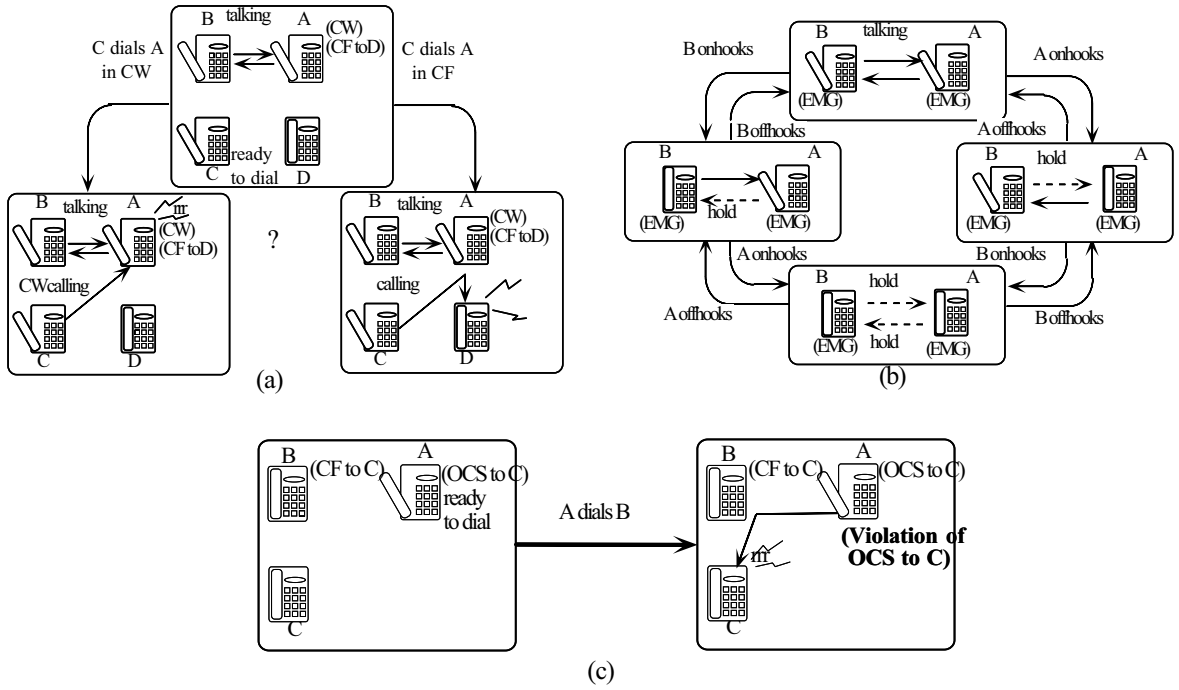


Figure 2.1: Interaction examples

be forwarded to D by the CF feature? This non-determinism will make A confused, thus should be avoided (See Figure 2.1(a)). □

Example 2.2 Interaction between OCS service and CF service

Originating Call Screening (OCS): This service allows the subscriber to specify that outgoing calls be either restricted or allowed according to a screening list. Suppose that x subscribes to OCS and that x puts y in a screening list. Then, any outgoing call to y from x is restricted, while any other call to z from x is allowed.

Call Forwarding (CF): The same as the one in Example 2.1.

Interaction $OCS\&CF$ Suppose that (1) A is an OCS subscriber who restricts the outgoing calls to C, and (2) B is CF subscriber who sets the forwarding address to C. At this time, if A dials B, the call is forwarded to C, so A will be calling C. This nullifies A's call restriction to C (See Figure 2.1(b)) □

Example 2.3 Interaction of Emergency call with itself.

Emergency call (EMG): This service is usually deployed on police and fire stations. In the case of an emergency incident, the call will be held even when the caller mistakenly onhooks. Suppose that x is a police station on which EMG is deployed, and that y has made a call to x and is now busy talking with x. Then, even when y onhooks, the call is on hold without being disconnected. Followed by that, if y offhooks, the held line reverts to a connected line and y can talk with x again. In order to disconnect the call, x has to onhook.

Interaction EMG_A & EMG_B: Suppose that both A and B subscribe to EMG and are talking to each other. Here, if A onhooks, the call is on hold by B's EMG. At this time, if A offhooks, the call reverts to the talking state. On the other hand, if B onhooks, the call is also held by A's EMG without being disconnected. Symmetrically, this is true when B onhooks first. Thus, neither A nor$B$ can disconnect the call. As a result, the call falls into a trap from which it never returns to the idle state (See Figure 2.1(b)). □

2.2 Service Specification

In order to formalise the feature interaction detection problem, we present fundamental definitions in this section. For the formalisation, we have to first describe services in a certain way. There are a number of researches concerning service description to formulate the feature interaction problem. We briefly enumerate them as follows. Specification Description Language (SDL) [65, 9, 16], linear time temporal [5, 6], Object Z [42], SMV language [53], PROMELA [30, 43], LOTOS [10, 54, 55], FSM-based method [4, 21, 37, 41, 50]), rule-based method [24, 27, 29, 34, 51, 62]. Each algorithm has its advantage and disadvantage.

In this paper, we adopt a *rule-based service specification* for a service description method such as State Transition Rules (STR)[29, 51] and declarative transition rules [24]. Rule-based methods have been widely studied for the practical use since:

- the modularity of the rule facilitates the addition or modification of the new service, and
- a simple IF-THEN form of each rule enables non-experts to easily design the service logic[51].

2.2.1 Notation

First, we define the syntax notation of the specification.

Definition 2.4 A *service specification* $\mathcal{S}$ is defined as $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$, where

(a) U is a set of constants representing service users.

(b) V is a set of variables.

(c) P is a set of predicate symbols.

(d) E is a set of event symbols.

(e) R is a set of rules.

(f) s_0 is the (*initial*) *state*.

Each rule $r \in R$ is defined as follows:

r: pre-condition [event] post-condition.

Pre(post)-condition is a list of *predicates* $p(x_1, ..., x_k)$'s, where $p \in P$,$x_i \in V$ and k is called *arity* which is a fixed number for each p. Especially, pre-condition can include

negations of predicates such as $\neg p(x_1, ..., x_k)$'s which implies $p(x_1, ..., x_k)$ does not hold. Next, *Event* is a predicate $e(x_1, ..., x_k)$, where $e \in E$, $x_i \in V$. For convenience, we represent pre-condition, event and post-condition of rule r as $Pre[r]$, $Ev[r]$ and $Post[r]$, respectively.

A *state* is defined as a list of *instances of predicates* $p(a_1, ..., a_k)$'s, where $p \in P$,$a_i \in U$. We think of each state as representing a *truth valuation*[47] where instances in the list are true, and instances not in the list are false.

Example 2.5 Figure 2.2 shows an example of a service specification for Plain Ordinary Telephone Service (POTS). For example, $pots_3$ means that "Suppose that user x receives dialtone and y is not idle. At this time, if x dials y, then x will receive a busytone". State s_0 means that two users A and B are idle. In state s_0, two instances $idle(A)$ and $idle(B)$ are true because they are included in s_0. On the other hand, any other instances (e.g., $dialtone(B)$ or $calling(A, B)$) are false since they are not included in s_0.

Since a state represents a truth valuation for all instances of predicates, it can also be described by a *Boolean vector* $\{true, false\}^n$. For example, letting true and false denote 1 and 0, respectively, the state s_0 in Example 2.5 is written as:

$$s_0 = \begin{bmatrix} i(A) & i(B) & d(A) & d(B) & c(A,A) & c(A,B) & ... & t(B,B) \\ 1 & 1 & 0 & 0 & 0 & 0 & ... & 0 \end{bmatrix}$$

where i, d, c and t stand for *idle, dialtone, calling* and *talk*, respectively.

2.2.2 State Transition Model

Next, we define the state transition specified by the rule-based specification.

Definition 2.6 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. For $r \in R$, let $x_1, ..., x_n$ $(x_i \in V)$ be variables appearing in r, and let $\theta = \langle x_1|a_1, ..., x_n|a_n \rangle$ $(a_i \in U)$ be a

$$
\begin{array}{lcl}
U & = & \{A, B\} \\
V & = & \{x, y\} \\
P & = & \{idle, dialtone, calling, busytone, talk\} \\
E & = & \{offhook, onhook, dial\} \\
R & = & \{ \\
 & & \begin{array}{lrcl}
pots_1: & idle(x) & [offhook(x)] & dialtone(x) \\
pots_2: & dialtone(x) & [onhook(x)] & idle(x) \\
pots_3: & dialtone(x), \neg idle(y) & [dial(x,y)] & busytone(x) \\
pots_4: & dialtone(x), idle(y) & [dial(x,y)] & calling(x,y) \\
pots_5: & calling(x,y) & [onhook(x)] & idle(x), idle(y) \\
pots_6: & calling(x,y) & [offhook(y)] & talk(x,y), talk(y,x) \\
pots_7: & talk(x,y), talk(y,x) & [onhook(x)] & idle(x), busytone(y) \\
pots_8: & busytone(x) & [onhook(x)] & idle(x)
\end{array} \\
 & & \} \\
s_0 & = & idle(A), idle(B)
\end{array}
$$

Figure 2.2: Rule-based specification for POTS

substitution replacing each x_i in r with a_i. Then, an *instance* of r based on θ (denoted by $r\theta$) is defined as a rule obtained from r by applying $\theta = \langle x_1|a_1, ..., x_n|a_n\rangle$ to r.

Definition 2.7 Let s be a state. We say rule r is *enabled* for θ at s, denoted by $en(s, r, \theta)$, iff all instances in $Pre[r\theta]$ take a true value at s (i.e., all instances are included in s). When $en(s, r, \theta)$, the *next state*, s' of s, can be generated by deleting all instances in $Pre[r\theta]$ from s and adding all instances in $Post[r\theta]$ to s. For convenience, we describe it by

$$s' = s - Pre[r\theta] + Post[r\theta]$$

where + and − respectively represent addition and deletion operators on the list. These operators work in the same way as union and subtraction operators on a set, respectively. At this time, we say a *state transition* from s to s' caused by an event $Ev[r\theta]$ is defined on $\mathcal{S}$, which is denoted by $s - Ev[r\theta] \rightarrow s'$ (or simply $s \rightarrow s'$). We say that state s is *reachable* from s_0 iff $s = s_0$ or a sequence of state transitions $s_0{-}e_0{\rightarrow}s_1$, $s_1{-}e_1{\rightarrow}s_2$, ..., $s_n{-}e_n{\rightarrow}s$ exists, which is denoted by $s_0{\rightarrow}^*s$ (reflexive and transitive closure of $\rightarrow$).

Example 2.8 Let us consider rule $pots_1$ and state s_0 in Figure 2.2. For a substitution $\theta = \langle x|A\rangle$, we have $pots_1\theta$

$$pots_1\theta: \quad idle(A) \quad [offhook(A)] \quad dialtone(A).$$

Here, $en(s_0, pots_1, \theta)$ holds since $idle(A)$ in $Pre[pots_1\theta]$ is included in s_0. Then, next state s_1 can be defined as follows:

$$\begin{aligned} s_1 &= s_0 - Pre[pots_1\theta] + Post[pots_1\theta] \\ &= (idle(A), idle(B)) - idle(A) + dialtone(A) \\ &= dialtone(A), idle(B) \end{aligned}$$

As a result, a state transition $s_0{-}offhook(A){\rightarrow}s_1$ is defined, which implies that if A offhooks at s_0, then A receives a dialtone, while B remains idle.

Next, let us apply $pots_4$ to s_1. For $\theta' = \langle x|A, y|B\rangle$, $en(s_1, pots_4, \theta')$ holds.

$$pots_3\theta' : \quad dialtone(A), idle(B) \quad [dial(A, B)] \quad calling(A, B).$$

Then, the successive next state s_2 can be generated as follows:

$$\begin{aligned} s_2 &= s_1 - Pre[pots_4\theta'] + Post[pots_4\theta'] \\ &= (dialtone(A), idle(B)) - (dialtone(A), idle(B)) + calling(A, B) \\ &= calling(A, B) \end{aligned}$$

So, a state transition $s_1 - dial(A, B) \rightarrow s_2$ is defined, which implies that if A dials B at s_1, then A is calling B.

2.3 Invariant Property

Practically, service(feature) designers may want to specify some properties which the target service must satisfy *at any time*. For example, POTS must satisfy the following property: "If user x is idle, then x is not busy at any time". Also, for OCS, the designer may describe that "If x specifies y in the screening list, then x is never calling y at any time". These properties are generally called *invariant properties* [37, 59]. In our framework, we describe these properties as follows.

Definition 2.9 Let $\mathcal{S} = \langle U, V, P, E, R, s_0\rangle$ be a service specification. Let $p(x_1, ..., x_k)$ with $p \in P, x_i \in V$ be any predicate in $\mathcal{S}$. Then, the *invariant formula* is recursively defined as follows:

(a) $p(x_1, ..., x_k)$ is an invariant formula.

(b) If f and g are invariant formulas, then $\neg f$ (negation), $f \vee g$ (disjunction) and $f \wedge g$ (conjunction) are invariant formulas.

The invariant formula $I_{\mathcal{S}}$ for a service $\mathcal{S}$ is intended to be satisfied by $\mathcal{S}$ at any time. In other words, $I_{\mathcal{S}}$ is intended to be satisfied at any states reachable from the initial state s_0 of $\mathcal{S}$. Next, we define the evaluation of an invariant formula at state s. In the evaluation, the operators $\neg, \vee, \wedge$ work in the same way as the logical *not*, *or*, *and*, respectively.

Definition 2.10 Let I be a invariant formula and $x_1, ..., x_n$ $(x_i \in V)$ be variables appearing in I. Let s be a state and let $I\theta$ be a formula obtained by applying a substitution $\theta = \langle x_1|a_1, ..., x_n|a_n\rangle$ $(a_i \in U)$ to I.

Then, I is $satisfied$ at s for θ, denoted by $s, \theta \vdash I$, iff $I\theta$ takes a true value by applying the truth valuation of s to $I\theta$. I is satisfied at s, denoted by $s \vdash I$, iff $s, \theta \vdash I$ for all θ.

Although the invariant formula is a logical formula consisting of predicates (without quantifiers), its expression power is equivalent to formulas in propositional logic.

Remark 2.11 From Definition 2.10, it is obvious that $s \vdash I$ holds s iff $I' = I\theta_1 \wedge I\theta_2 \wedge ... \wedge I\theta_l$ takes a true value at s, where l is the number of all possible substitutions. The true or false of any instance of predicates in I' is uniquely determined by s, that is, I' is a propositional formula. Therefore, the evaluation of I at each state s is not a difficult task.

Example 2.12 Let us give a simple invariant formula for POTS. The property "If user x is idle, then x is not busy at any time", can be described as

$$I = \neg idle(x) \vee \neg busytone(x).$$

Let us evaluate I in the following state s. We assume here $U = \{A, B\}$.

$$s = idle(A), busytone(B)$$

For a substitution $\theta_1 = \langle x|A\rangle$, we have

$$I\theta_1 = \neg idle(A) \vee \neg busytone(A)$$

Now, based on s, we evaluate $I\theta_1$,

$$I\theta_1 = \neg true \vee \neg false = false \vee true = true$$

That is, we have $s, \theta_1 \vdash I$. Similarly, for $\theta_2 = \langle x|B\rangle$,

$$\begin{aligned} I\theta_2 &= \neg idle(B) \vee \neg busytone(B) \\ &= \neg false \vee \neg true \\ &= true \end{aligned}$$

So, $s, \theta_2 \vdash I$. Consequently, $s \vdash I$, that is, I is satisfied at s.

In general, the invariant formula should be specified manually by the service (feature) designers. This is because the invariant property for a service must originate from the *semantics* of the service, such as requirements and functionalities. Therefore, we assume that the invariant formula is given by the designer and it will be input of the interaction detection problem.

2.4 Class of Feature Interactions

In this paper, we focus primarily on the following three types of interactions. These are very typical cases of interactions and are discussed in many papers(e.g., [21, 24, 37, 27, 51]):

deadlock: Functional conflicts of two or more services cause a mutual prevention of their service execution, which result in a deadlock.

loop: The service execution is trapped into a loop from which the service execution never returns to the initial state.

non-determinism: An event can simultaneously activate two or more functionalities of different services. As a result, it cannot be determined exactly which functionality should be activated.

violation of invariant: The invariant property, which is asserted by each service, is violated by the service combination.

2.5 Problem Formulation

First, we define the following undesirable states. Each of them corresponds to a class of interactions discussed before.

Definition 2.13 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a given service specification, and let $I_{\mathcal{S}}$ be a given invariant formula for $\mathcal{S}$. Then, a state s is said to be a

deadlock state: iff $[s_0 \rightarrow^* s]$ and $[\neg en(s, r, \theta)$ for any $r \in R$ and $\theta]$.

loop state: iff $[s_0 \rightarrow^* s]$ and [there exists such a state s' that $s \rightarrow^* s' \rightarrow^* s$ and $\neg(s \rightarrow^* s_0)$].

non-deterministic state: iff $[s_0 \rightarrow^* s]$ and [there exists a pair of rules $r, r' \in R$ such that $en(s, r, \theta)$ and $en(s, r', \theta')$, and that $Ev[r\theta] = Ev[r'\theta']$].

violating state: iff $[s_0 \rightarrow^* s]$ and $[s \not\models I_{\mathcal{S}}]$.

The above four kinds of states are called *undesirable states*. The pair $(\mathcal{S}, I_{\mathcal{S}})$ is called *safe* iff there is no undesirable state for a service specification $\mathcal{S}$ and an invariant formula $I_{\mathcal{S}}$.

Example 2.14 We explain the non-deterministic state using an example. Consider the following two rules cw_4 and cf_{10}, and a state s.

$$cw_4: \quad CW(x), talk(x, y), dialtone(z) \quad [dial(z, x)] \quad CW(x), talk(x, y), CWcalling(z, x).$$

$$cf_{10}: \quad CF(y, z), dialtone(x), idle(z) \quad [dial(x, y)] \quad CF(y, z), calling(x, z).$$

$$s = CW(A), CF(A, D), talk(A, B), talk(B, A), dialtone(C), idle(D)$$

The rules cw_4 and cf_{10} respectively describe the functionality of CW and CF shown in Example 2.1. Rule cw_4 implies CW feature such that "Suppose that x subscribes CW

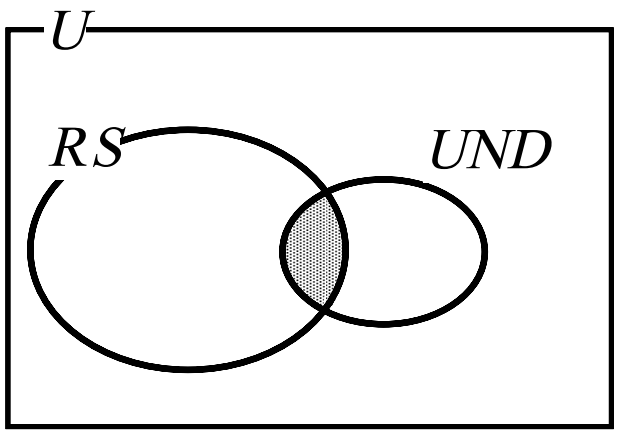


Figure 2.3: Classification of states

feature, x is talking with y and z receives a dialtone. In this situation, if z dials x, then x receives an additional call from a third party z".

On the other hand, rule cf_{10} implies CF feature such that "Suppose that y subscribes CF feature and pre-sets the forwarding address to z, z is idle, and x receives a dialtone. In this situation, if x dials y, then the call to y is forwarded to z and x is calling z".

Also, the state s means that: user A has both a CW feature and a CF feature with forwarding to D, A is talking with B, C receives a dialtone, and D is idle. Now, we suppose that s is reachable from the initial state s_0. Let $\theta_1 = \langle x|A, y|B, z|C\rangle$ and $\theta_2 = \langle x|C, y|A, z|D\rangle$. Then, $en(s, cw_4, \theta_1)$ and $en(s, cf_{10}, \theta_2)$, and $Ev[cw_4\theta_1] = Ev[cf_{10}\theta_2] = dial(C, A)$. This is exactly the non-deterministic behavior explained in Example 2.1.

Note that each condition of the undesirable states in Definition 2.13 is formed by a conjunction of the *reachable condition* $s_0 \rightarrow^* s$ and the *undesirable condition* such as $s \not\models I_{\mathcal{S}}$. Let UND be a set of all states satisfying such an undesirable condition, and let RS be a set of all reachable states from s_0. Also, let U be a set of all states. Figure 2.3 shows a classification of states schematically. Then, the set of undesirable states is just the intersection of UND and RS, depicted by the shaded part.

Next, we define a combined operator of two service specifications. Because of the good modularity of the rule-based specification, we can easily combine two specifications as follows.

Definition 2.15 For two specifications $\mathcal{S}_1 = \langle U_1, V_1, P_1, E_1, R_1, s_{10} \rangle$ and $\mathcal{S}_2 = \langle U_2, V_2, P_2, E_2, R_2, s_{20} \rangle$, we define a *combined specification* $\mathcal{S}_1 \oplus \mathcal{S}_2 = \langle U, V, P, E, R, s_0 \rangle$ such that $U = U_1 \cup U_2$, $V = V_1 \cup V_2$, $P = P_1 \cup P_2$, $E = E_1 \cup E_2$, $R = R_1 \cup R_2$ and $s_0 = s_{10} + s_{20}$ where $+$ denotes the addition operator in Definition 2.7.

Now, we are ready to define the feature interaction on the rule-based service specification.

Definition 2.16 Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be given service specifications, and let $I_{\mathcal{S}1}$ and $I_{\mathcal{S}2}$ be given invariant formulas for $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Then, we say $\mathcal{S}_1$ *interacts with* $\mathcal{S}_2$ iff

(a) both $(\mathcal{S}_1, I_{\mathcal{S}1})$ and $(\mathcal{S}_2, I_{\mathcal{S}2})$ are safe, and

(b) $(\mathcal{S}_1 \oplus \mathcal{S}_2, I_{\mathcal{S}1} \wedge I_{\mathcal{S}2})$ is not safe.

Consequently, the feature interaction detection problem is defined as follows:

Definition 2.17 Feature Interaction Detection Problem

Input: Two service specifications $\mathcal{S}_1$ and $\mathcal{S}_2$, and their invariant formulas $I_{\mathcal{S}1}$ and $I_{\mathcal{S}2}$.

Output: True (detected) or False (not detected) for "$\mathcal{S}_1$ *interacts with* $\mathcal{S}_2$"

In terms of the classification of states shown in Figure 2.3, the feature interaction detection problem is to identify the intersection of RS and UND for $\mathcal{S}_1 \oplus \mathcal{S}_2$ and $I_{\mathcal{S}1} \wedge I_{\mathcal{S}2}$.

Chapter 3

Conventional Exhaustive Search

3.1 Full Reachability Graph FRG

Most of the conventional feature interaction detection frameworks (e.g., [4, 16, 21, 27, 37, 43, 51, 54]) adopt the exhaustive search method, which explores all possible reachable states. An exhaustive search is generally performed by means of a *reachability graph*.

For a given service specification $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$, the rule applications from initial state s_0 construct a finite state machine(FSM) consisting of all reachable states from s_0, in which a state moves to the next state by the occurrence of an event. Since an FSM can be described by a labelled directed graph, here we directly define such an FSM as a directed graph [1]. In the following, we represent a directed edge from s to s' labelled by e as a triple (s, e, s').

Definition 3.1 A labelled directed graph is defined as $G = \langle N, L, T \rangle$ where:

(a) N is a set of *nodes*.

(b) L is a set of *labels* attached to the directed edges.

(c) $T \subseteq N \times L \times N$ is a set of *directed edges*.

For any directed graph G, a *directed path* ρ is a sequence of directed edges: $\rho = (s_1, e_1, s_2)$, (s_2, e_2, s_3), ..., (s_n, e_n, s_{n+1}). For this, the node s_1 is called the *head node* of ρ, while s_{n+1} is called the *tail node* of ρ. n is called a *length* of ρ. A directed path is a *directed cycle* iff its head node and tail node are identical. A node is called a *terminal* iff it has no outgoing edge.

Definition 3.2 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. A *full reachability graph* for a given $\mathcal{S}$ is a labelled directed graph $FRG(\mathcal{S}) = \langle N, L, T \rangle$ such that:

(a) $N = \{s | s_0 \rightarrow^* s\}$.

(b) L is a set of all instances of events.

(c) $T = \{(s, Ev[r\theta], s') |\ \ s - Ev[r\theta] \rightarrow s'\ \}$

In FRG, each node represents a reachable state s, and each arc outgoing from s represents a state transition which occurs at s. The algorithm in Figure 3.1 constructs an FRG for a service specification $\mathcal{S}$. In the algorithm, we define *waiting* as a set of nodes.

Example 3.3 Figure 3.2 shows a full reachability graph for the POTS specification in Figure 2.2. We can see that there are 12 reachable states (represented by ovals) and 30 state transitions (represented by directed arrows).

3.2 Interaction Detection Algorithm EXH

Using the full reachability graph FRG, we can easily identify the undesirable states in Definition 2.13 as follows.

Proposition 3.4 *The following properties are satisfied for $FRG(\mathcal{S})$ and given $I_{\mathcal{S}}$.*

(a) there exists a terminal s $\Leftrightarrow$ s is a deadlock state.

FRG Construction Algorithm

Input: $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$

Output: $FRG(\mathcal{S}) = \langle N, L, T \rangle$

Procedure:

$waiting = N := \{s_0\}; \quad L = T := \emptyset;$

repeat {

  select s from $waiting$;

  for any $r \in R$ s.t. $en(s, r, \theta)$ for some θ {

    generate the next state s' by applying r to s;

    add $Ev[r\theta]$ to L ;

    if $(s' \notin N)$ then {

      add s' to N;

      add s' to $waiting$;

    }

    add $(s, Ev[r\theta], s')$ to T ;

  }

  delete s from $waiting$;

}

Until $waiting = \emptyset$

Figure 3.1: FRG construction algorithm

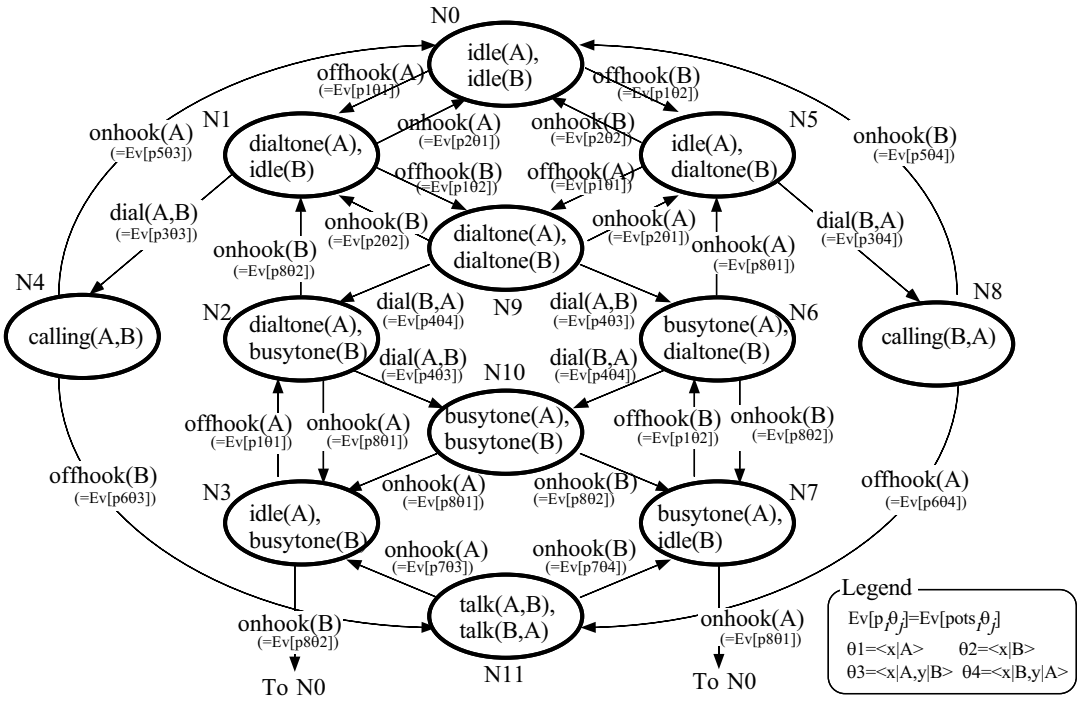


Figure 3.2: FRG for POTS specification

(b) there exists a directed cycle starting from s, and there exists no directed path from s to s_0 $\Leftrightarrow$ s is a loop state.

(c) there exists a node s which has a pair of outgoing edges (s, e_1, s') and (s, e_2, s'') such that $e_1 = e_2 \Leftrightarrow s$ is a non-deterministic state.

(d) there exists a node s such that $s \not\Vdash I_{\mathcal{S}} \Leftrightarrow s$ is a violating state.

Proof: Straightforward from Definitions 2.13 and 3.2. □

According to Proposition 3.4 undesirable states except for violating state in Definition 2.13 are easily identified from the structure of FRG. In order to detect the violating state, we have to evaluate $s \not\Vdash I_{\mathcal{S}}$, where $I_{\mathcal{S}} = I_{\mathcal{S}1} \wedge I_{\mathcal{S}2}$ for each reachable state s. For this, we recall Remark 2.11. From this, $s \not\Vdash I_{\mathcal{S}}$ holds iff

$$\begin{aligned} \neg I'_{\mathcal{S}} &= \neg(I_{\mathcal{S}}\theta_1 \wedge I_{\mathcal{S}}\theta_2 \wedge ... \wedge I_{\mathcal{S}}\theta_l) \\ &= \neg I_{\mathcal{S}}\theta_1 \vee \neg I_{\mathcal{S}}\theta_2 \vee ... \vee \neg I_{\mathcal{S}}\theta_l \end{aligned}$$

$s \not\models I_{\mathcal{S}}$ **Evaluation Procedure**

function eval$(s, I_{\mathcal{S}})$

/* $\neg I_{\mathcal{S}} = t_1 \vee t_2 \vee ... \vee t_h$ */

for each $t_i = p_1 \wedge p_2 \wedge ... \wedge p_q$

for each θ

if (some $p_j\theta$ is false at s) then break;

else return($true$);

return($false$);

Figure 3.3: Evaluation procedure

takes true value at s. Without loss of generality, we assume that $\neg I_{\mathcal{S}}$ is described as sum-of-product form [1]: $\neg I_{\mathcal{S}} = t_1 \vee t_2 \vee ... \vee t_h$, where $t_i = p_1 \wedge p_2 \wedge ... \wedge p_q$ and p_i is a predicate $p_i(x_1, ..., x_k)$ or its negation. Then, we have

$$\begin{aligned} \neg I'_{\mathcal{S}} &= \neg I_{\mathcal{S}}\theta_1 \vee \neg I_{\mathcal{S}}\theta_2 \vee ... \vee \neg I_{\mathcal{S}}\theta_l \\ &= \bigvee_i (t_1 \vee t_2 \vee ... \vee t_h)\theta_i \\ &= \bigvee_{i,j} t_j\theta_i \end{aligned}$$

$\neg I'_{\mathcal{S}}$ take true value at s iff at least one $t_j\theta_i$ takes a true value at s. So, $s \not\models I_{\mathcal{S}}$ holds iff $s, \theta_i \vdash t_j$ holds for some t_j and θ_i. Consequently, in order to detect the violating state s, we apply the procedure $eval(I_{\mathcal{S}}, s)$ in Figure 3.3. In the algorithm, we assume $\neg I_{\mathcal{S}} = t_1 \vee t_2 \vee ... \vee t_h$ is given.

Example 3.5 Consider again the service specification for POTS in Figure 2.2. Let us evaluate the following invariant expression I saying that "If x receives dialtone, then [x

is not busy] and [x is not idle]"

$$\begin{aligned} I &= \neg dialtone(x) \vee (\neg idle(x) \wedge \neg busytone(x)) \\ &= (\neg dialtone(x) \vee \neg idle(x)) \wedge (\neg dialtone(x) \vee \neg busytone(x)) \end{aligned}$$

Hence, we obtain $\neg I$ as follows.

$$\neg I = dialtone(x) \wedge idle(x) \vee dialtone(x) \wedge \neg busytone(x)$$

Consider the following state s:

$$s = idle(A), busytone(B), dialtone(A)$$

The term $dialtone(x) \wedge idle(x)$ of $\neg I$ is true for $\theta_1 = \langle x|A \rangle$. So, we can conclude $s \not\models I$. If s is reachable from the initial state s_0, then s is a violating state * . Note that it is not necessary to all possible substitutions and all predicates in I when $s \not\models I$ holds.

Thus, in order to detect the interactions between given two specifications $\mathcal{S}_1$ and $\mathcal{S}_2$, we first construct $FRG(\mathcal{S}_1 \oplus \mathcal{S}_2)$, then identify the undesirable states using Proposition 3.4.

The interaction detection algorithm based on exhaustive search, denoted by Algorithm EXH, is summarized in Figure 3.4. Consider again Fig 2.3. Since the set RS of reachable states is the set of nodes in FRG, we can see that EXH works as shown in Figure 3.5.

The following proposition characterizes Algorithm EXH.

Proposition 3.6 *The following property is satisfied for Algorithm EXH:*

$$\mathcal{S}_1 \textit{ interacts with } \mathcal{S}_2 \quad \Leftrightarrow \quad EXH \textit{ returns } \textit{“Detected”}.$$

Proof: Straight forward from Proposition 3.4. □

Although interaction detection using the FRG is quite simple and powerful, it may suffer from the *state explosion problem* [60]. That is, the size of the FRG (i.e., the size

* In fact, we can see that s is not reachable since it does not appear in the FRG shown in Figure 3.2.

of RS in Figure 3.5) exponentially grows when the number of users and the number of rules in the specification become large. For example, consider the POTS specification in Figure 2.2. As we vary the number of users from 2 to 5, then the number of nodes in the FRG grows as 12, 54, 270, 1458. For the number of edges, it increases exponentially as 30, 234, 1728, 12690. We will show more practical evaluation in Chapter 6.

Detection Algorithm EXH

Input: $\mathcal{S}_1, \mathcal{S}_2, I_{\mathcal{S}1}, I_{\mathcal{S}2}$

Output: "Detected" or "Not detected"

Procedure:

$\mathcal{S} = \mathcal{S}_1 \oplus \mathcal{S}_2$; $I_{\mathcal{S}} = I_{\mathcal{S}1} \wedge I_{\mathcal{S}2}$;

Phase 1: Construct FRG($\mathcal{S}_1 \oplus \mathcal{S}_2$);

Phase 2: Detect undesirable states using Proposition 3.4;

If (detected) then return("Detected");

else return("Not detected");

Figure 3.4: Detection Algorithm EXH

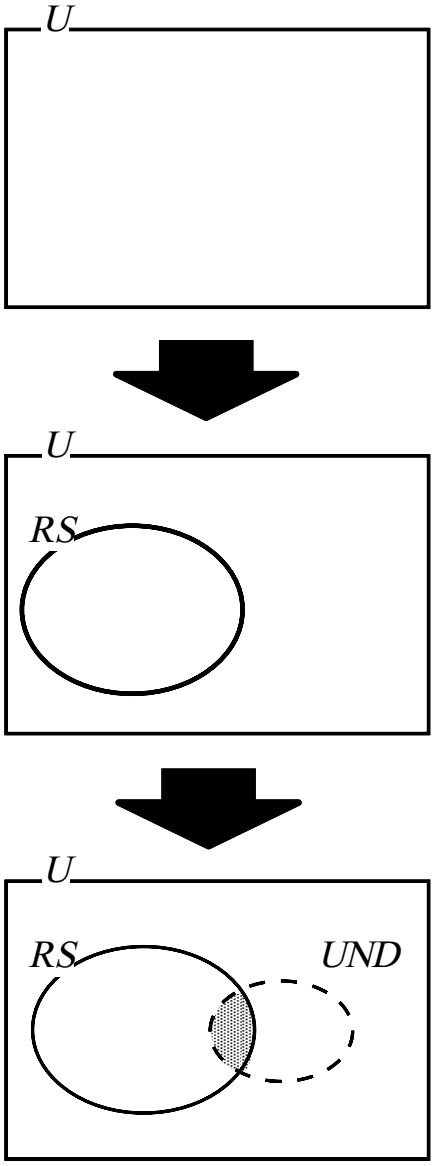


Figure 3.5: Conceptual overview of EXH

Chapter 4

Exploiting Symmetric Relation

4.1 Introduction

The straightforward way to circumvent the state explosion problem in Algorithm EXH is to reduce the size of the full reachability graph FRG (which is also called *state space*[31]) in a certain manner. In this Chapter, we try to propose a new interaction detection algorithm, called Algorithm SYM, which attains efficient state space reduction without losing any necessary information for interaction detection.

In order to achieve the reduction, we utilise an equivalence relation called *permutation symmetry* with respect to users. The basic idea of permutation symmetry is originally proposed in other research fields such as Petri net verification[32]. However, here we show that the state reduction is successfully achieved by means of a specific constraint on telecommunication services.

In telecommunication systems, there exists a specific constraint that "all subscribers of a service X are guaranteed to be able to use the same functionality of X". Under this constraint, suppose that both users A and B are subscribers of X. If we know A's possible behavior on X, then we can infer B's behavior on X from A's, because B can use X in the same way as A. Therefore, we can discard the state transitions for B's behaviors

since they can be reproduced from A's. Based on this idea, we define an equivalence relation *symmetrical* on the states and transitions for the state reduction. Then, we define a new reachability graph, called the *symmetric reachability graph*, SRG, in which all symmetrical states are grouped into one node. Finally, by using the SRG, we provide a new algorithm, SYM, for the detection of all four types of undesirable states: deadlock, loop, non-deterministic and violating states.

Permutation symmetry is well suited for the constraints of telecommunication service. Therefore, the state reduction by permutation symmetry preserves effectively all information stored in the FRG. As a result, Algorithm SYM attains the optimal (exact) interaction detection based on a necessary and sufficient condition in the same way as Algorithm EXH, with much smaller state space.

The remainder of this chapter is organised as follows. In the next section, we define the permutation symmetry after explanation of the key idea. Then, we define a particular permutation symmetry, called consistent symmetry, which guarantees the reachabiltiy of symmetrical states. In Section 4.3, we present a new reachability graph, SRG, and then we propose Algorithm SYM. Finally, Section 4.5 summarizes the related interaction detection methods using reduction techniques.

4.2 Exploiting Symmetric Relation

4.2.1 Key Idea

Here we explain the key idea in more detail by means of an example. The key idea to achieve this reduction is to utilise a relation between states called permutation symmetry. There exists the following specific constraint in telecommunication services:

Constraint: All subscribers of service X must be able to use functionality of X in the same way.

For example, let us consider that A uses POTS's dialing functionality to B: "Suppose that A receives a dialtone and B is idle. At this time, if A dials B, then A will be calling B". Similarly, when B uses this functionality, the situation is: "Suppose that B receives a dialtone and A is idle. At this time, if B dials A, then B will be calling A." We can easily convince that two situations are symmetrical with respect to users, that is, the one can be inferred from the other only by swapping A and B. In terms of our service specification, we can observe the symmetry on states and state transitions.

Example 4.1 Consider the following states s_1 and s_2:

$$\begin{aligned} s_1 &= dialtone(A), idle(B), busytone(C) \\ s_2 &= dialtone(C), idle(A), busytone(B) \end{aligned}$$

We see s_1 and s_2 are symmetrical, in the sense that s_2 is obtained from s_1 just by substituting A for C, B for A, C for B. In other words, letting $U = \{A, B, C\}$, there exists a permutation $\phi : U \to U$ such that $\phi(A) = C$, $\phi(B) = A$, $\phi(C) = B$ from s_1 to s_2. Now, let us apply the following rule to s_1 and s_2:

$$pots_3 : dialtone(x), idle(y)[dial(x, y)]calling(x, y).$$

Then, for $\theta_1 = \langle x|A, y|B\rangle$ and $\theta_2 = \langle x|C, y|A\rangle$, $en(s_1, pots_3, \theta_1)$ and $en(s_2, pots_3, \theta_2)$.

$$\begin{aligned} pots_3\theta_1 : &\quad dialtone(A), idle(B) \quad [dial(A, B)] \quad calling(A, B). \\ pots_3\theta_2 : &\quad dialtone(C), idle(A) \quad [dial(C, A)] \quad calling(C, A). \end{aligned}$$

As a result, the following next states s'_1 and s'_2 are obtained from s_1 and s_2, respectively.

$$\begin{aligned} s'_1 &= calling(A, B), busytone(C) \\ s'_2 &= calling(C, A), busytone(B). \end{aligned}$$

Now, we can observe that there also exists the same permutation ϕ from s'_1 to s'_2. Roughly speaking, this fact implies that state transition $s_2 - dial(C, A) \to s'_2$ can be reproduced as $\phi(s_1) - \phi(dial(A, B)) \to \phi(s'_1)$ from $s_1 - dial(A, B) \to s'_1$. Therefore, we need no longer store either states s_2, s'_2 nor transition $s_2 - dial(C, A) \to s'_2$ in the reachability graph.

From this example, we reach a hypothesis in the general case, that if we have a state transition $s{-}e{\rightarrow}s'$, then we also have $\phi(s){-}\phi(e){\rightarrow}\phi(s')$ for any permutation ϕ on U. If the hypothesis is true, then it is sufficient to have only $s{-}e{\rightarrow}s'$ in the reachability graph, since we can infer symmetric state transitions from $s{-}e{\rightarrow}s'$. As a result, the reduced reachability graph will preserve the all information of the original FRG.

4.2.2 Permutation Symmetry

In this section, we formally define the permutation symmetry. Throughout this section, we assume that a service specification $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ is given unless otherwise specified. First, we define all permutations with respect to users.

Definition 4.2 Let $Perm(U)$ denote a set of all permutations $\phi : U \rightarrow U$. Each element ϕ of $Perm(U)$ is called a *permutation symmetry*.

Example 4.3 Let $U = \{A, B, C\}$. Then, $Perm(U) =$
$\left\{ \begin{bmatrix} A & B & C \\ A & B & C \end{bmatrix} \begin{bmatrix} A & B & C \\ A & C & B \end{bmatrix} \begin{bmatrix} A & B & C \\ B & A & C \end{bmatrix} \begin{bmatrix} A & B & C \\ B & C & A \end{bmatrix} \begin{bmatrix} A & B & C \\ C & A & B \end{bmatrix} \begin{bmatrix} A & B & C \\ C & B & A \end{bmatrix} \right\}$
Each permutation symmetry of $Perm(U)$ specifies a bijection ϕ from U to U. For example, $\begin{bmatrix} A & B & C \\ C & A & B \end{bmatrix}$ specifies a bijection ϕ such that $\phi(A) = C$, $\phi(B) = A$, $\phi(C) = B$.

Next, we extend $\phi \in Perm(U)$ for states, rules and substitutions as follows.

Definition 4.4 Let $\phi \in Perm(U)$ be a permutation symmetry. Then, for any instance $p(a_1, ..., a_k)$ of a predicate with $p \in P, a_i \in U$, we define $\phi(p(a_1, ..., a_k)) = p(\phi(a_1), ..., \phi(a_k))$. At this time,

(a) For any state s, we define $\phi(s)$ to be a state obtained by applying ϕ to each instance of predicate in s.

(b) For any instances $r\theta$ of rule $r \in R$, we define $\phi(r\theta)$ to be an instance of rule obtained by applying ϕ to each instance of predicate in $r\theta$. Similarly, we define $\phi(Pre[r\theta])$, $\phi(Post[r\theta])$ and $\phi(Ev[r\theta])$.

(c) For any substitution $\theta = \langle x_1|a_1, ..., x_n|a_n\rangle$, $x_i \in V$, $a_i \in U$, we define $\phi(\theta) = \langle x_1|\phi(a_1), ..., x_n|\phi(a_n)\rangle$.

(d) For any instances $I\theta$ of invariant formula I, we define $\phi(I\theta)$ to be an instance of the invariant formula obtained by applying ϕ to each predicate in $I\theta$.

Two states s and s' are *symmetrical*, denoted by $s \approx s'$ iff $\phi(s) = s'$ for some $\phi \in Perm(U)$ * .

Example 4.5 Consider again Example 4.1. Then, for a permutation symmetry $\phi = \begin{bmatrix} A & B & C \\ C & A & B \end{bmatrix}$, we can see that (a) $\phi(s_1) = s_2$ and $\phi(s_1') = s_2'$, (b) $\phi(pots_3\theta_1) = pots_3\theta_2$, and (c) $\phi(\theta_1) = \langle x|\phi(A), y|\phi(B)\rangle = \langle x|C, y|A\rangle = \theta_2$.

Before showing the main theorem, we present the following lemma. Intuitively, Lemma 4.6 implies that it does not matter whether we use a permutation symmetry ϕ before or after the instantiation of the rule r (and invariant formula I) based on the substitution θ.

Lemma 4.6 *Let a permutation symmetry $\phi \in Perm(U)$ be given. Then, for any rule $r \in R$ and any substitution θ, $\phi(r\theta) = r\phi(\theta)$ holds. Also, $\phi(Pre[r\theta]) = Pre[r\phi(\theta)]$, $\phi(Post[r\theta]) = Post[r\phi(\theta)]$ and $\phi(Ev[r\theta]) = Ev[r\phi(\theta)]$ hold. For any invariant formula I, $\phi(I\theta) = I\phi(\theta)$*

Proof: Suppose that $\phi = \begin{bmatrix} a_1 & a_2 & ... & a_{|U|} \\ b_1 & b_2 & ... & b_{|U|} \end{bmatrix} \in Perm(U)$, $a_i, b_i \in U$ be given.

* The relation $\approx$ is an equivalence relation on states, since (i) $s \approx s$ (reflexive), (ii) $s \approx s'$ implies $s' \approx s$ (symmetric) and (iii) $s \approx s'$ and $s' \approx s''$ imply $s \approx s''$ (transitive).

Let $p = p_m(x_{m1}, x_{m2}, ..., x_{mk})$ be any predicate in rule r. If we apply $\theta = \langle x_1|a_{i1}, ..., x_n|a_{in}\rangle$ to r, then p in r is instantiated to an instance $p' = p_m(a_{im1}, a_{im2}, ..., a_{imk})$ in $r\theta$. Next, we apply ϕ to $r\theta$. Then p' in $r\theta$ is transformed into the following instance p'' in $\phi(r\theta)$: $p'' = p_m(\phi(a_{im1}), \phi(a_{im2}), ..., \phi(a_{imk})) = p_m(b_{im1}, b_{im2}, ..., b_{imk})$

On the other hand, if we first apply ϕ to θ, we obtain $\phi(\theta) = \langle x_1|\phi(a_{i1}), ..., x_n|\phi(a_{in})\rangle = \langle x_1|b_{i1}, ..., x_n|b_{in}\rangle$ Next, by applying $\phi(\theta)$ to r, then we get the following instance p^* of p in $r\phi(\theta)$: $p^* = p_m(b_{im1}, b_{im2}, ..., b_{imk}) = p''$ Thus, for any predicate p in $r \in R$, two instances p'' in $\phi(r\theta)$ and p^* in $r\phi(\theta)$ are identical. Hence we conclude that $\phi(r\theta) = r\phi(\theta)$. By the same discussions, we can prove that $\phi(Pre[r\theta]) = Pre[r\phi(\theta)]$, $\phi(Post[r\theta]) = Post[r\phi(\theta)]$ and $\phi(Ev[r\theta]) = Ev[r\phi(\theta)]$.

Also, for the invariant formula, by applying the same discussions to each predicate in I, we have $\phi(I\theta) = I\phi(\theta)$. □

Now, we are ready to provide the main theorem for a state transition.

Theorem 4.7 *For any permutation symmetry $\phi \in Perm(U)$,*

$$s\text{−}Ev[r\theta]\text{→}s' \quad \Leftrightarrow \quad \phi(s)\text{−}\phi(Ev[r\theta])\text{→}\phi(s').$$

Proof: ($\Rightarrow$) Assume that the state transition $s\text{−}Ev[r\theta]\text{→}s'$ is defined on $\mathcal{S}$. Since $en(s, r, \theta)$, all predicates of $Pre[r\theta]$ are included in s from Definition 2.7. From Definition 4.4, all predicates of $\phi(Pre[r\theta])$ are clearly included in $\phi(s)$. From Lemma 4.6, all predicates of $Pre[r\phi(\theta)]$ are included in $\phi(s)$. This implies $en(\phi(s), r, \phi(\theta))$. Hence, from Definition 2.7 and Lemma 4.6, the next state s^* of $\phi(s)$ is obtained as follows:

$$\begin{aligned} s^* &= \phi(s) - Pre[r\phi(\theta)] + Post[r\phi(\theta)] \\ &= \phi(s) - \phi(Pre[r\theta]) + \phi(Post[r\theta]) \\ &= \phi(s - Pre[r\theta] + Post[r\theta]) \\ &= \phi(s') \end{aligned}$$

Therefore, a state transition $\phi(s)-Ev[r\phi(\theta)]\rightarrow\phi(s')$ is defined on $\mathcal{S}$. From Lemma 4.6, $\phi(s)-Ev[r\phi(\theta)]\rightarrow\phi(s') = \phi(s)-\phi(Ev[r\theta])\rightarrow\phi(s')$, as required.

($\Leftarrow$) Assume that $s^*-e^*\rightarrow s'^* = \phi(s)-Ev[r\phi(\theta)]\rightarrow\phi(s')$ is defined on $\mathcal{S}$. Since ϕ is a bijection, there exists an inverse $\phi^{-1} \in Perm(U)$. By applying ϕ^{-1}, we have $\phi^{-1}(s^*)-\phi^{-1}(e^*)\rightarrow\phi^{-1}(s'^*)$ $= s-Ev[r\theta]\rightarrow s'$. Hence, we show that if a state transition $s^*-e^*\rightarrow s'^*$ is defined on $\mathcal{S}$, then $\phi^{-1}(s^*)-\phi^{-1}(e^*)\rightarrow\phi^{-1}(s'^*)$ is defined on $\mathcal{S}$. However, it directly follows from $\Rightarrow$. □

Theorem 4.7 clearly explains the hypothesis discussed in Section 5.1 is true. That is, if we have a state transition $s-e\rightarrow s'$ on $\mathcal{S}$, then we can confirm that all of its symmetric transitions $\phi(s)-\phi(e)\rightarrow\phi(s')$'s are defined on $\mathcal{S}$. Theorem 4.7 is easily extended for the sequences of state transitions.

Corollary 4.8 *For any permutation symmetry $\phi \in Perm(U)$, the following properties hold.*

$$(a) \quad s_1 - e_1 \rightarrow s_2 - e_2 \rightarrow ... \rightarrow s_n \quad \Leftrightarrow \quad \phi(s_1) - \phi(e_1) \rightarrow \phi(s_2) - \phi(e_2) \rightarrow ... \rightarrow \phi(s_n)$$

$$(b) \quad s_0\rightarrow^* s \quad \Leftrightarrow \quad \phi(s_0)\rightarrow^*\phi(s).$$

Proof: Property(a) follows by repeated use of Theorem 4.7. Property(b) is a direct consequence of Property(a). □

Thus, if we have only one transition sequence τ on $\mathcal{S}$, then we can confirm the existence of all sequences symmetrical with τ by Corollary 4.8.

4.2.3 Consistent Symmetry

For a state transition defined on $\mathcal{S}$, Theorem 4.7 and Corollary 4.8 guarantee the existence of its symmetrical transitions for all permutation symmetry $\phi \in Perm(U)$. However, they never guarantee the reachability of symmetric states from the initial state s_0 of $\mathcal{S}$. That is, even if $s_0\rightarrow^* s$, we cannot generally conclude $s_0\rightarrow^*\phi(s)$ since Corollary 4.8 only says

$\phi(s)$ is reachable from $\phi(s_0)$. In order to assure the reachability of symmetric states, we must put a restriction on ϕ so that $\phi(s_0) = s_0$. We define such a permutation symmetry ϕ as a consistent symmetry.

Definition 4.9 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a given service specification. A permutation symmetry ϕ is a *consistent symmetry* iff $\phi(s_0) = s_0$. Let $CS(U, s_0)$ denote a set of all consistent symmetries, i.e.,

$$CS(U, s_0) = \{\phi | \phi \in Perm(U) \wedge \phi(s_0) = s_0\}$$

Two states s and s' are *consistently symmetrical*, denoted by $s \approx_c s'$, iff $\phi(s) = s'$ for some $\phi \in CS(U, s_0)$. For a given state s, the *symmetric class* of s, denoted by $[s]$, is defined as $[s] = \{s' | s \approx_c s'\}$. For $[s]$, s is called a *representative* of $[s]$.

Remark 4.10 Note that all theories in the previous section are still valid for the consistent symmetry ϕ since $CS(U, s_0) \subseteq Perm(U)$, and that the relation $\approx_c$ is an equivalence relation on states as discussed in $\approx$.

Additionally, the consistent symmetry provides the following theorem, which guarantees the reachability from initial state s_0.

Theorem 4.11 *Let $\phi \in CS(U, s_0)$ be any consistent symmetry. Then, $s_0 \rightarrow^* s \Leftrightarrow s_0 \rightarrow^* \phi(s)$.*

Proof: It follows from Corollary 4.8(b) assuming $\phi(s_0) = s_0$. □

Example 4.12 Consider the following initial state s_0.

$$s_0 = CW(A), CW(B), idle(A), idle(B), idle(C)$$

Now, we suppose $U = \{A, B, C\}$ Then, the set of consistent symmetries is

$$CS(U, s_0) = \left\{ \begin{bmatrix} A & B & C \\ A & B & C \end{bmatrix} \quad \begin{bmatrix} A & B & C \\ B & A & C \end{bmatrix} \right\}$$

s_0 means all A, B and C are idle and A and B are subscribers of CW. Assume that the following state s is reachable from s_0.

$$s = CW(A), CW(B), talk(A, B), talk(B, A), CWcalling(C, A)$$

in which A is receiving CW calling from C. Now, we apply a permutation symmetry ϕ such that $\phi(A) = C, \phi(B) = B, \phi(C) = A$ to s. Then,

$$\phi(s) = CW(C), CW(B), talk(C, B), talk(B, C), CWcalling(A, C)$$

C was not a CW subscriber at s_0. Therefore, it is inconsistent that $\phi(s)$, in which C is receiving CW calling, is reachable from s_0. This is justified by a fact that ϕ is not a consistent symmetry, i.e., $\phi(s_0) \neq s_0$.

4.3 Symmetric Reachability Graph SRG

Based on theories for permutation symmetry presented in the previous section, we define a new reachability graph called the symmetric reachability graph, SRG.

Definition 4.13 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. A symmetric reachability graph for a given $\mathcal{S}$ is a labelled directed graph $SRG(\mathcal{S}) = \langle N, L, T \rangle$ such that:

(a) $N = \{[s] | s_0 \rightarrow^* s\}$ (that is, N is a partition of the set of all reachable states),

(b) L is a set of instances of events, and

(c) $T = \{([s], Ev[r\theta], [s']) |\ s - Ev[r\theta] \rightarrow s^* \wedge (s^* \approx_c s')\}$

In SRG, each node represents a symmetric class $[s]$ with a representative (state) s, and each arc outgoing from $[s]$ represents a state transition which occurs at the representative s. Also, from Theorem 4.11, each node $[s]$ in $SRG(\mathcal{S})$ implies that all states belonging

to the class $[s]$ are reachable from the initial state s_0. Figure 4.1 shows the construction algorithm for an SRG, for a service specification $\mathcal{S}$. In the algorithm, we define *waiting* as a set of nodes.

Example 4.14 Figure 4.2 shows a symmetric reachability graph, SRG for the POTS specification in Figure 2.2. Each of nodes $N4$, $N5$, $N6$ and $N7$ represents a symmetric class with two states, while each of other nodes represents a class with only one state. For example, $N4$ actually represents two symmetrical states $dialtone(A), idle(B)$ and $dialtone(B), idle(A)$, while $N0$ does one state $idle(A), idle(B)$. Compared with FRG in Figure 3.2, SRG has smaller number of nodes(8 nodes) and edges(20 edges) (though it may be a small reduction in this example).

As for $SRG(\mathcal{S})$, the following proposition holds:

Proposition 4.15 *The following properties are satisfied for $SRG(\mathcal{S})$.*

(a) Each directed path ρ in $SRG(\mathcal{S})$: $\rho = ([s_1], e_1, [s_2]), ([s_2], e_2, [s_3]), ..., ([s_n], e_n, [s_{n+1}])$ has, for each state $s_1^* \in [s_1]$, a corresponding sequence of state transitions τ on S: $\tau = s_1^* - e_1^* \rightarrow s_2^* - e_2^* \rightarrow s_3^* - ... \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^*$, where $s_i \approx_c s_i^* (1 \leq i \leq n+1)$.

(b) Each sequence of transitions τ on $\mathcal{S}$: $\tau = s_1 - e_1 \rightarrow s_2 - e_2 \rightarrow s_3 - ... \rightarrow s_n - e_n \rightarrow s_{n+1}$, where $s_0 \rightarrow^* s_1$, has a corresponding directed path in $SRG(\mathcal{S})$: $\rho = ([s_1^*], e_1^*, [s_2^*]), ([s_2^*], e_2^*, [s_3^*])), ..., ([s_n^*], e_n^*, [s_{n+1}^*])$, where $s_i \approx_c s_i^* (1 \leq i \leq n+1)$.

Proof: We prove this proposition by means of induction over the length n of ρ/τ. Property(a): When the length of ρ is zero, property(a) is clearly satisfied from Definition 4.13(a). Next, assume that property(a) is satisfied when the length of ρ is n, and assume that we have a directed path ρ' such that $\rho' = ([s_1], e_1, [s_2]), ..., ([s_n], e_n, [s_{n+1}]), ([s_{n+1}], e_{n+1}, [s_{n+2}])$. From the inductive hypothesis, it follows that, for each state $s_1^* \in [s_1]$,

SRG Construction Algorithm

Input: $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$

Output: $SRG(\mathcal{S}) = \langle N, L, T \rangle$

Procedure:

$waiting = N := \{[s_0]\}; \quad L = T := \emptyset;$

repeat

  select $[s]$ from $waiting$;

  for any $r \in R$ s.t. $en(s, r, \theta)$ for some θ {

    generate the next state s' by applying r to s;

    add $Ev[r\theta]$ to L ;

    if ($[s^*] \notin N$ s.t. $s^* \approx_c s'$) then {

      add $[s']$ to N ; add $[s']$ to $waiting$;

      add $([s], Ev[r\theta], [s'])$ to T ;

    }

    else add $([s], Ev[r\theta], [s^*])$ to T;

  }

  delete $[s]$ from $waiting$;

until $waiting = \emptyset$

Figure 4.1: SRG construction algorithm

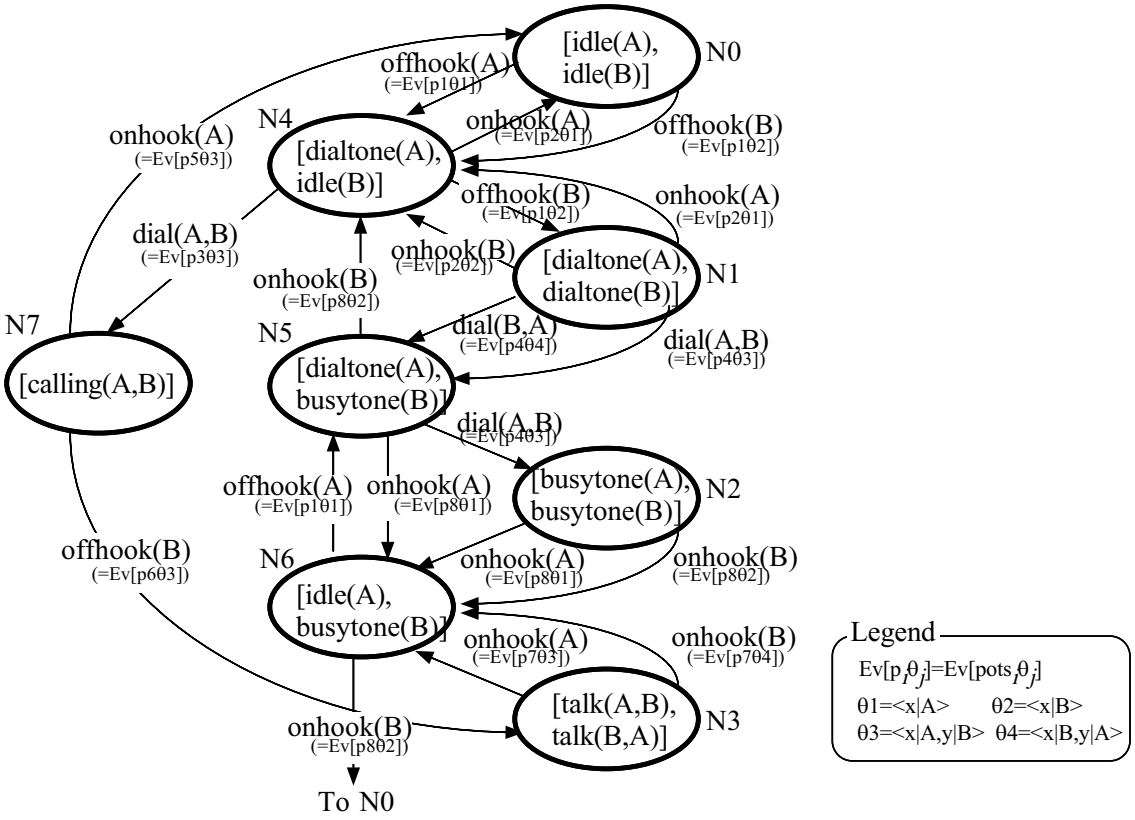


Figure 4.2: SRG for POTS specification

there exists a sequence of state transitions: $s_1^* - e_1^* \rightarrow s_2^* - ... \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^*$. From Definition 4.13(c) and the edge $([s_{n+1}], e_{n+1}, [s_{n+2}])$ of ρ', there exists a transition $s_{n+1}-e_{n+1}\rightarrow s'_{n+2}$, where $s'_{n+2} \in [s_{n+2}]$. From Theorem 4.7, for $\phi \in CS(U, s_0)$ such that $\phi(s_{n+1}) = s^*_{n+1}$, there exists $\phi(s_{n+1})-\phi(e_{n+1})\rightarrow\phi(s'_{n+2})$. Let $s^*_{n+2} = \phi(s'_{n+2})$ and $e^*_{n+1} = \phi(e'_{n+1})$. Since $s_{n+2}\approx_c s'_{n+2}$, $s_{n+2}\approx_c s^*_{n+2}$. Thus, there exists a transition $s^*_{n+1}-e^*_{n+1}\rightarrow s^*_{n+2}$. This means that $s_1^* - e_1^* \rightarrow s_2^* - ... \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^* - e_{n+1}^* \rightarrow s_{n+2}^*$ is a sequence of transitions with the requested properties.

Property(b): When the length of τ is zero, property(b) is clearly satisfied from Definition 4.13(a). Next, assume that property(b) is satisfied when the length of τ is n, and assume that we have a directed path τ' such that $\tau' = s_1 - e_1 \rightarrow s_2 - ... \rightarrow s_n - e_n \rightarrow s_{n+1} - e_{n+1} \rightarrow s_{n+2}$, where $s_0\rightarrow^* s_1$. From the inductive hypothesis, it follows that there exists a directed path: $([s_1^*], e_1^*, [s_2^*]), ([s_2^*], e_2^*, [s_3^*])), ..., ([s_n^*], e_n^*, [s_{n+1}^*])$ and $s_{n+1}\approx_c s^*_{n+1}$. From a transition $s_{n+1}-e_{n+1}\rightarrow s_{n+2}$ of τ', we know $s_0\rightarrow^* s_{n+2}$. From Definition 4.13(a), there exists a node $[s^*_{n+2}]$ in $SRG(\mathcal{S})$ such that $s^*_{n+2}\approx_c s_{n+2}$. Moreover, from Theorem 4.7, for such that $\phi(s_{n+1}) = s^*_{n+1}$, there exists a transition $s^*_{n+1}-\phi(e_{n+1})\rightarrow\phi(s_{n+2})$. Since $s_{n+2}\approx_c s^*_{n+2}$, $\phi(s_{n+2})\approx_c s^*_{n+2}$. Hence, from Definition 4.13(c), there exists an edge $([s^*_{n+1}], \phi(e_{n+1}), [s^*_{n+2}])$. This means that $([s_1^*], e1^*, [s_2^*]), ..., ([s_n^*], e_n^*, [s_{n+1}^*]), ([s_{n+1}^*], e_{n+1}^*, [s_{n+2}^*])$ is a directed path in $SRG(\mathcal{S})$ with the required properties. □

4.4 Interaction Detection Algorithm SYM

Using the symmetric reachability graph, SRG, we can identify the undesirable states in Definition 2.13 as follows:

Proposition 4.16 *The following properties are satisfied for $SRG(\mathcal{S})$ and given $I_{\mathcal{S}}$.*

(a) there exists a terminal $[s]$ $\Leftrightarrow$ each $s^* \in [s]$ is a deadlock state.

(b) there exists a directed cycle starting from $[s]$*, and there exists no directed path from* $[s]$ *to* $[s_0]$ $\Leftrightarrow$ *each* $s^* \in [s]$ *is a loop state.*

(c) there exists a node $[s]$ *which has a pair of outgoing edges* $([s], e_1, [s'])$ *and* $([s], e_2, [s''])$ *such that* $e_1 = e_2$ $\Leftrightarrow$ *each* $s^* \in [s]$ *is a non-deterministic state.*

(d) there exists a node $[s]$ *such that* $s \not\vdash I_{\mathcal{S}}$ $\Leftrightarrow$ *each* $s^* \in [s]$ *is a violating state.*

Proof:

Properties (a)(b): They directly follow from Proposition 4.15 and Theorem 4.11.

Property (c) : ($\Rightarrow$) Assume that there exists a pair of the edges $([s], e, [s'])$ and $([s], e, [s''])$ in $SRG(\mathcal{S})$. From Definition 4.13(c), there exists a pair of transitions $s{-}e{\rightarrow}ss'$, $s{-}e{\rightarrow}ss''$ such that $s_0{\rightarrow}^*s$, $ss' \in [s']$,$ss'' \in [s'']$. From Theorem 4.7, for any $\phi \in CS(U, s_0)$, there exists a pair of transitions $\phi(s){-}\phi(e){\rightarrow}\phi(ss')$, $\phi(s){-}\phi(e){\rightarrow}\phi(ss'')$. Since $s_0{\rightarrow}^*s$, $s_0{\rightarrow}^*\phi(s)$ from Theorem 4.11. Hence, from Definition 2.13, $s^* = \phi(s) \in [s]$ is a non-deterministic state, as required.

($\Leftarrow$) Since s^* is a non-deterministic state, there exists a pair of transitions $s^*{-}e{\rightarrow}ss'$, $s^*{-}e{\rightarrow}ss''$ such that $s_0{\rightarrow}^*s^*$. From Definition 4.13(a), there exists a node $[s]$ such that $s = \phi(s^*)$ in $SRG(\mathcal{S})$. From Theorem 4.7, there exists a pair of transitions $\phi(s^*){-}\phi(e){\rightarrow}\phi(ss')$, $\phi(s^*){-}\phi(e){\rightarrow}\phi(ss'')$ on $\mathcal{S}$. From Definition 4.13(c), $SRG(\mathcal{S})$ has a pair of edges $([\phi(s^*)], \phi(e), [s'])$, $([\phi(s^*)], \phi(e), [s''])$, where $\phi(ss') \in [s']$,$\phi(ss'') \in [s'']$. Thus, $SRG(\mathcal{S})$ has a pair of edges $([s], e^*, [s'])$, $([s], e^*, [s''])$, as required. □

Property (d): ($\Rightarrow$) For each $s^* = \phi(s) \in [s]$ with $\phi \in CS(U, s_0)$, $s_0 \rightarrow^* s^*$ holds by Theorem 4.11 and Proposition 4.15. Since $s \not\vdash I_{\mathcal{S}}$, there exists a substitution θ such that $s, \theta \vdash \neg I_{\mathcal{S}}$. This means $\neg I_{\mathcal{S}}\theta$ takes true value at s from Definition 2.10. Clearly, $\phi(\neg I_{\mathcal{S}}\theta)$ is true at $\phi(s)$ from Definition 4.4. From Lemma 4.6, $\neg I_{\mathcal{S}}\phi(\theta)$ is true at $\phi(s)$. Hence, $\phi(s), \phi(\theta) \vdash \neg I_{\mathcal{S}}$. This means $\phi(s) = s^*$ is a violating state.

($\Leftarrow$) Since s^* is a violating state, $s_0 \rightarrow^* s^*$ holds. So, there exists a node $[s]$ in SRG such that $s = \phi(s^*) \approx_c s^*$ from Definition 4.13(a). Since $s^* \not\Vdash I_{\mathcal{S}}$, $s \not\Vdash I_{\mathcal{S}}$ holds by the same discussion as $\Rightarrow$. □

Thus, in order to detect the interactions between two specifications $\mathcal{S}_1$ and $\mathcal{S}_2$, we first construct $SRG(\mathcal{S}_1 \oplus \mathcal{S}_2)$, then identify the undesirable states using Proposition 4.16.

The interaction detection algorithm using the symmetric reachability graph SRG, denoted by Algorithm SYM, is summarized in Figure 4.3. Again, we recall Fig 2.3. Now, let [RS] (and [UND]) be a partition of RS (and UND, respectively) based on the relation symmetrical $\approx_c$. Then, the set [RS] corresponds to the set of nodes in the SRG. Therefore, we can see that SYM works as shown in Figure 4.4.

The following proposition characterizes Algorithm SYM.

Proposition 4.17 *The following property is satisfied for Algorithm SYM:*

$$\mathcal{S}_1 \textit{ interacts with } \mathcal{S}_2. \quad \Leftrightarrow \quad SYM \textit{ returns } \text{“}Detected\text{”}$$

Proof: Straight forward from Proposition 4.16. □

For a state s, the SRG groups all states symmetrical with s together by means of the relation $\approx_c$, which enables the state reduction of the FRG as shown in Example 4.14. Also, by using the constraint in telecommunication services, the backbone theories based on permutation symmetry succeed in preserving all information required for the interaction detection. As a result, the interactions are detected by both necessary and sufficient conditions. This fact means Algorithm SYM achieves the optimal interaction detection quality. As for the state reduction ratio, we look at a simple example here. As an example, consider the POTS specification in Figure 2.2. As we vary the number of users from 2 to 5, then the number of nodes in the SRG grows as 8, 16, 30, 50. For the

number of edges, it increases 20, 72, 204, 482. These are linear in the number of users. The more practical evaluation will be presented in Chapter 6.

Detection Algorithm SYM

Input: $\mathcal{S}_1, \mathcal{S}_2, I_{\mathcal{S}1}, I_{\mathcal{S}2}$

Output: "Detected" or "Not detected"

Procedure:

$\mathcal{S} = \mathcal{S}_1 \oplus \mathcal{S}_2$; $I_{\mathcal{S}} = I_{\mathcal{S}1} \wedge I_{\mathcal{S}2}$;

Phase 1: Construct SRG($\mathcal{S}_1 \oplus \mathcal{S}_2$);

Phase 2: Detect undesirable states using Proposition 4.16;

If (detected) then return("Detected");

else return("Not detected");

Figure 4.3: Detection algorithm SYM

4.5 Related Works

There are several interaction detection methods exploiting some kinds of reduction techniques.

Cameron et al. [15] proposed the tool CADRES-FI which utilizes a *state abstraction technique*. This method abstracts the details of users behavior not concerning to the interaction detection by means of heuristics. Due to its heuristical characteristics, detection quality and coverage deeply depends on knowledge of the verifier.

Lin et al. [44] also proposed an industrial service creation environment and its fundamental framework, which is an extension work from [15]. In this method, they define an equivalence relation similar to ours by focusing input/output (READ/WRITE) relations

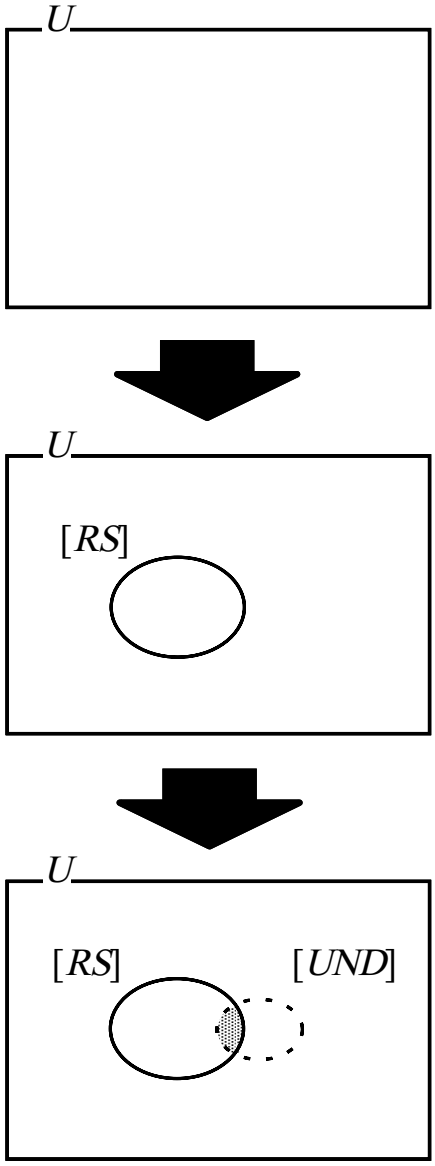


Figure 4.4: Conceptual overview of SYM

among events. Since this relation is, however, based on heuristical observation, it does not achieve the optimal detection as is in Algorithm SYM. As a result, it needs a decision from subjective experts to identify, following the detection process.

Nitsche [50] demonstrated how the homomorphisms of FSM can be used for interaction detection. In this method, the homomorphism replaces certain events in FSM with empty word ϵ, which results in the abstraction of state space. However, since the homomorphism is given manually, the optimal interaction detection is not always guaranteed.

In other research fields, there are also several techniques for the reduction of state space such as stubborn sets [59, 60], partial order [30, 31] and symbolic model checking [47]. They are expected to apply to further improvement of Algorithm SYM, as discussed in our future works.

Chapter 5

Static Algorithm using Petri Net Structure

5.1 Introduction

The interaction detection algorithms presented so far enumerate the reachable states first by using their own reachability graphs, and then check for each state producing an undesirable condition (such as deadlock). This type of verification, using a reachability graph is generally called *dynamic verification* [32], in the sense that it traces the system's reachable behaviors in all execution sequences, based on a system description such as the service specification. Dynamic verification has an advantage, in that it can verify any property defined on the reachable states (a so-called dynamic property). In this sense, the properties satisfied on the undesirable states are dynamic properties (see Definition 2.13). However, the cost for exploring all reachable states by means of a reachablity graph may be too expensive when the verified system becomes very large.

On the other hand, verification using some extra information but not using reachable state exploration is generally called *static verification*. Since it does not require costly reachability analysis, it can be applied to large-scale problems. However, static verification

cannot always cover all dynamic properties in general.

Both verification approaches have their own advantage and disadvantage, and the relationship is clearly trade-off. Therefore, these cannot be directly compared with each other and should be chosen for different purposes.

In this Chapter, we propose a new static interaction detection algorithm, called Algorithm PINV, which extensively utilises the structural information of the service specification, instead of exploring the reachable states. Concretely, we first extract the structure of the service specification as a Petri net structure, and then apply the *P-invariant method* [48, 32, 33] of Petri net to the interaction detection.

The outline of Algorithm PINV is as follows. We first construct a logically equivalent Petri net model for a given rule-based service specification, then determine the set of *undesirable candidate states* based on only the structure of the rules. Next, we identify all candidates in the set which are not reachable from the initial state using the P-invariant of the Petri-net, and delete them from the set. Let us see again the classification of states shown in Figure 2.3 in Chapter 2. Intuitively, Algorithm PINV first identifies the set UND as a set of candidates, then tries to obtain the intersection of that set by means of the P-invariant method.

Due to characteristics of static algorithms mentioned above, there are two limitations on Algorithm PINV. Firstly, since the definitions of the undesirable states are based on dynamic properties, Algorithm PINV cannot cover all types of undesirable states. Therefore, we focus on only non-deterministic and violating states. Secondly, P-invariant gives only the necessary condition for reachability. Thus, undesirable candidates may not actually be reachable, theoretically speaking. This means that Algorithm PINV may detect some interactions which do not actually occur, but it will never miss interactions which actually occur.

However, even incorporating the above limitations, Algorithm PINV achieves great

gains. First, since the P-invariant method works in a static way without costly reachable state exploration, Algorithm PINV is expected to achieve the drastic cost reduction for the interaction detection. Second, it is applicable to large-scale services with many users, to which the dynamic verification cannot be applied. It is also used to narrow the scope of possible interactions since Algorithm PINV never misses the actual interactions.

This rest of this chapter is organised as follows. In the next section, we present a Petri net model onto which the rule-based service specification is mapped. Section 5.3 shows the P-invariant method of the Petri net model. Then, in Section 5.4, we explain how to apply the P-invariant method to interaction detection, and propose Algorithm PINV. Finally, Section 5.5 concludes this chapter with related works on static methods and Petri net based methods for feature interaction detection.

5.2 Petri Net Model

5.2.1 Labelled Pr/T Net

In this section, we define a kind of Petri-Net which is a simple extension of *predicate transition nets(Pr/T Nets)*[48]. However, it is still a subclass of High Level Petri net (or Coloured Petri net).

Definition 5.1 A labelled Pr/T net $\mathcal{N}$ is defined by $\mathcal{N} = \langle U, V, P, E, T, F, H, L_a, L_t, M_0 \rangle$, where

(a) U is a set of constants.

(b) V is a set of variables ranging over U.

(c) P is a set of *places*.

(d) E is a set of predicates, and each element is represented by $e(x_1, ..., x_n)$, $x_i \in V$.

(e) T is a set of *transitions*, and $P \cap T = \phi$.

(f) $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. Each element of F is called *arc*.

(g) $H \subseteq (P \times T)$ is a set of *inhibitor arcs*.

(h) L_a is the *arc labelling function* which attaches a label $\langle x_1, ..., x_k \rangle$, where each $x_i \in V$, to the arc or the inhibitor arc, and k is called arity of the arc. For any input/output arc of each place $p \in P$, its arity k is a unique constant associated with p.

(i) L_t is the *transition labelling function* which attaches the element of E to each transition.

(j) M_0 is an *initial marking* (Marking will be defined in Definition 5.5).

$In(t) = \{p|(p,t) \in H \cup (F \cap (P \times T))\}$ and $Out(t) = \{p|(t,p) \in F \cap (T \times P)\}$ are called *input places* and *output places* of transition t, respectively.

Remark 5.2 The differences between our labelled Pr/T net and Pr/T net are that (1) our model includes the inhibitor arcs to represent the negation of predicates(see Definition 2.4) and (2) the label is attached to each transition.

Example 5.3 Figure 5.1(a) shows a labelled Pr/T net. The arities of places *idle*, *dialtone*, *calling* are 1,1,2, respectively. Figure 5.1(b) shows schematic representation of the labelled Pr/T net.

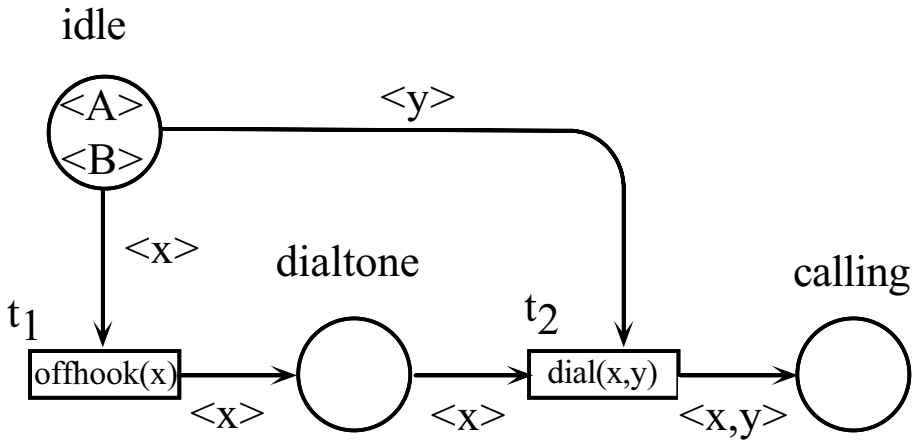
Remark 5.4 In the following, we use schematic representation(as shown in Figure 5.1(b)) rather than algebraic representation(as shown in Figure 5.1(a)).

Next, we define the marking of labelled Pr/T net.

Definition 5.5 *Colour set of place* p, denoted by $C(p)$, is the set of all constant k-tuples $\langle a_1, ..., a_k \rangle$, where each $a_i \in U$ and k is the arity of p. Each element of $C(p)$ is called a

U = {A, B}
V = {x, y}
P = {idle, dialtone, calling}
E = {offhook(x), dial(x,y)}
T = {t1, t2}
F = {(idle, t1), (t1, dialtone), (dialtone, t2),
(t2, calling), (idle, t2) }
H = {}
La: La(idle, t1) = La(t1, dialtone)= La(dialtone, t2)= <x>,
La(t2, calling)=<x,y>, La(idle, t2)=<y>
Lt: Lt(t1) = offhook(x), Lt(t2) = dial(x,y)
M_0: M_0(idle) = {<A>, <B>}

(a)



(b)

Figure 5.1: An example of labelled Pr/T net

coloured token(or simply a token) and it can be allocated to a place p. The allocation of the tokens to each place $p \in P$ is called *marking* and it is defined as a mapping function from P to the multiset over $C(p)$.

A marking M can be also expressed in terms of a vector: $M = (M(p_1), ..., M(p_m))$.

Let $Q(t)$ be a set of variables that occur at the incident arcs of t and at the predicate on t. Let $x_1, ..., x_l \in V$ be an arbitrary (but fixed) sequence of all variables in $Q(t)$. Then, the *colour set of transition* t, denoted by $C(t)$, is the set of all constant l-tuples $\langle a_1, ..., a_l \rangle$ obtained by substituting each x_i in the sequence by a constant in U. Thus, each colour $c = \langle a_1, ..., a_l \rangle \in C(t)$ can be interpreted as a substitution such that $\langle x_1 | a_1, ..., x_l | a_l \rangle$. We represent this substitution by $\theta(c)$.

Example 5.6 Consider again the labelled Pr/t net shown in Figure 5.1. At first, we explain the colour set of place. Since the arities of both places *idle* and *dialtone* are 1, $C(idle) = C(dialtone) = \{\langle A \rangle, \langle B \rangle\}$. Next, since the arity of place *calling* is 2, $C(calling) = \{\langle A, A \rangle, \langle A, B \rangle, \langle B, A \rangle, \langle B, B \rangle\}$. The tokens $\langle A \rangle$ and $\langle B \rangle$ are allocated in *idle* and no token is allocated in any other place. This marking M_0 is denoted by

$$\begin{array}{lccc} & idle & dialtone & calling \\ M_0 = & (\{\langle A \rangle, \langle B \rangle\}, & \emptyset, & \emptyset \ ). \end{array}$$

Next, we explain the colour set $C(t)$ of transition t. Consider the transition t_2. Since two variables x and y are attached on the incident arcs $(idle, t_2)$, $(dialtone, t_2)$,$(t_2, calling)$ and on the transition label $dial(x, y)$, $Q(t_2) = \{x, y\}$. Consider an ordering x, y for $Q(t_2)$. Then, $C(t_2) = \{\langle A, A \rangle, \langle A, B \rangle, \langle B, A \rangle, \langle B, B \rangle\}$. For example, consider $\langle A, B \rangle \in C(t_2)$. This colour represents a substitution $\theta\langle A, B \rangle = \langle x | A, y | B \rangle$. (If we consider the order of y,x, then the substitution is considered as $\theta\langle A, B \rangle = \langle y | A, x | B \rangle$).

Definition 5.7 Consider $t \in T$, $c \in C(t)$, and a marking M. For $L_a(p, t)$, define $L_a(p, t)\theta(c)$ be a constant tuple obtained by substituting the variables in $L_a(p, t)$ according

to $\theta(c)$. Then, t is *enabled* for $\theta(c)$ under M iff

$$\begin{aligned} \forall p \in In(t) \quad \{L_a(p,t)\theta(c)\} \subseteq M(p) \quad \cdots \quad \textbf{if}((p,t) \in F \cap (P \times T)) \\ \{L_a(p,t)\theta(c)\} \notin M(p) \quad \cdots \quad \textbf{if}((p,t) \in H) \end{aligned}$$

If $t \in T$ is enabled for $\theta(c)$ under M, then t can fire. Firing of t changes the current marking M into the *next marking* M' as follows:

$$\forall p \in P; \quad M'(p) = M(p) - \{L_a(p,t)\theta(c)\} + \{L_a(t,p)\theta(c)\}$$

where $+$ and $-$ are the union and difference operations defined on multisets[48].

A marking M is called *reachable* from M_0 iff $M = M_0$ or there exists at least one sequence of marking $M_0, M_1, ..., M_n = M$ such that M_{i+1} is a next marking of M_i.

Example 5.8 We explain the firing of transitions using Figure 5.2. Consider the marking M in Figure 5.2(a), which is also specified by

$$\begin{array}{rccc} & idle & dialtone & calling \\ M = & (\{\langle B\rangle\}, & \{\langle A\rangle\}, & \emptyset \ ). \end{array}$$

For example, take transition t_2 and $\theta\langle A, B\rangle = \langle x|A, y|B\rangle$. Then $In(t_2) = \{dialtone, idle\}$, and $\{L_a(dialtone, t_2)\theta\langle A, B\rangle\} = \{\langle A\rangle\} = M(dialtone)$ and $\{L_a(idle, t_2)\theta\langle A, B\rangle\} = \{\langle B\rangle\} = M(idle)$. Thus, t_2 is enabled for $\theta\langle A, B\rangle$ under M.

Now, suppose that t_2 fires for $\theta\langle A, B\rangle$ under M. Then, tokens $\langle A\rangle$ and $\langle B\rangle$ are respectively removed from places *dialtone* and *idle*, since $L_a(dialtone, t_2)\theta\langle A, B\rangle = \langle A\rangle$ and $L_a(idle, t_2)\theta\langle A, B\rangle = \langle B\rangle$. Moreover, a new token $\langle A, B\rangle$ is allocated to place *calling*, because $L_a(t_2, calling)\theta\langle A, B\rangle = \langle A, B\rangle$. As the result, M is transformed into the following next marking M', which is also shown in Figure 5.2(b).

$$\begin{array}{rccc} & idle & dialtone & calling \\ M' = & (\ \emptyset, & \emptyset, & \{\langle A, B\rangle\} \ ) \end{array}$$

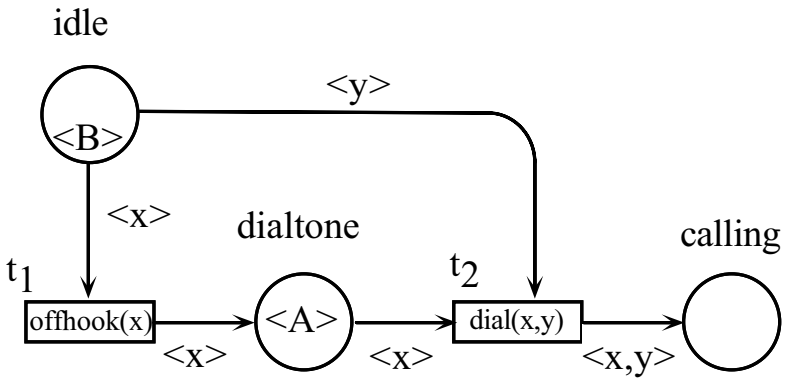


Figure 5.2: An explanation of firing

5.2.2 Service Specification Net

Here, we define the particular labelled Pr/T net for a service specification $\mathcal{S}$.

Definition 5.9 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. Then, a *service specification net* $\mathcal{N}(\mathcal{S}) = \langle U', V', P', E', T, F, H, L_a, L_t, M_0 \rangle$ for a given service specification $\mathcal{S}$ is a labelled Pr/T net which satisfies the following conditions.

(a) $U' = U$

(b) $V' = V$

(c) $P' = P$

(d) E' is a set of events $e(x_1, ..., x_k)$ with $e \in E, x_i \in V$.

(e) For each rule $r_i \in R$, there is exactly one transition $t_i \in T$ such that $L_t(t_i) = Ev[r_i]$.

(f) For each predicate $p_{ij}(x_{i1}, ..., x_{im})$ in pre-condition of rule $r_i \in R$, exactly one arc with a label $\langle x_{i1}, ..., x_{im} \rangle$ exists from place p_{ij} to transition t_i.

(g) For each predicate $\neg p_{ij}(x_{i1}, ..., x_{im})$ in pre-condition of rule $r_i \in R$, exactly one inhibitor arc with a label $\langle x_{i1}, ..., x_{im} \rangle$ exists from place p_{ij} to transition t_i.

(h) For each predicate $p_{ij}(x_{i1}, ..., x_{im})$ in post-condition of rule $r_i \in R$, exactly one arc with a label $\langle x_{i1}, ..., x_{im} \rangle$ exists from transition t_i to place p_{ij}.

(i) If the initial state s_0 includes $p(c_1, ..., c_m)$, then the initial marking $M_0(p) = \langle c_1, ..., c_m \rangle$.

According to Definition 5.9, we can easily understand that (1) the pre(post)-condition of a rule corresponds to the input(output) places of a transition, (2) the event of a rule corresponds to the predicate attached to a transition, (3) the initial state corresponds to the initial marking.

Note that any state s of $\mathcal{S}$ uniquely described as a marking M on $\mathcal{N}(\mathcal{S})$. That is, if a predicate $p(a_1, ..., a_n)$ holds(that is, takes a true value) on state s, then place p has a token $\langle a_1, ..., a_n \rangle$ under M. Also, from the structure of $\mathcal{N}(\mathcal{S})$, we can describe an application of rule r_i as a firing of the corresponding transition t_i.

Example 5.10 Consider a service specification $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ in which U, V, P, E, s_0 are the same as the POTS specification in Figure 2.2, but R contains only $pots_1$ and $pots_4$.

$$\begin{array}{lllll} pots_1: & & idle(x) & [offhook(x)] & dialtone(x) \\ pots_4: & dialtone(x), idle(y) & & [dial(x,y)] & calling(x,y) \end{array}$$

Then a labelled Pr/T net shown in Figure 5.1 is a service specification net for the service specification $\mathcal{S}$. Now we explain in detail the correspondence between rule $pots_4$ and transition t_2. Since $pots_4$ has two predicates $idle(y)$ and $dialtone(x)$ in $Pre[pots_4]$, t_2 has two input places *idle* and *dialtone*, and has two arcs, $(idle, t_2)$ with a label $\langle y \rangle$ and $(dialtone, t_2)$ with a label $\langle x \rangle$. Similarly, for a predicate $calling(x, y)$ in $Post[pots_4]$, t_2 has an output place *calling*, and has an arc $(t_2, calling)$ with a label $\langle x, y \rangle$. Next, for $dial(x, y)$ of $Ev[pots_4]$, t_2 is attached with a label $dial(x, y)$.

Next, consider the following two states:

$$\begin{aligned} s &= dialtone(A), idle(B) \\ s' &= calling(A, B) \end{aligned}$$

Then, markings M and M' in Example 5.8 (thus two markings in Figure 5.2(a) and (b)) respectively represent these two states s and s', that is, $M \equiv s$ and $M' \equiv s'$. The state transition from s to s' by rule $pots_4$ is already explained in Example 2.8. For this state transition, we can correspond a firing of t_2 for $\theta\langle A, B \rangle = \langle x|A, y|B \rangle$ which transforms M into M' (this firing is already explained in Example 5.8).

From the above observations, we can see that the following lemma holds.

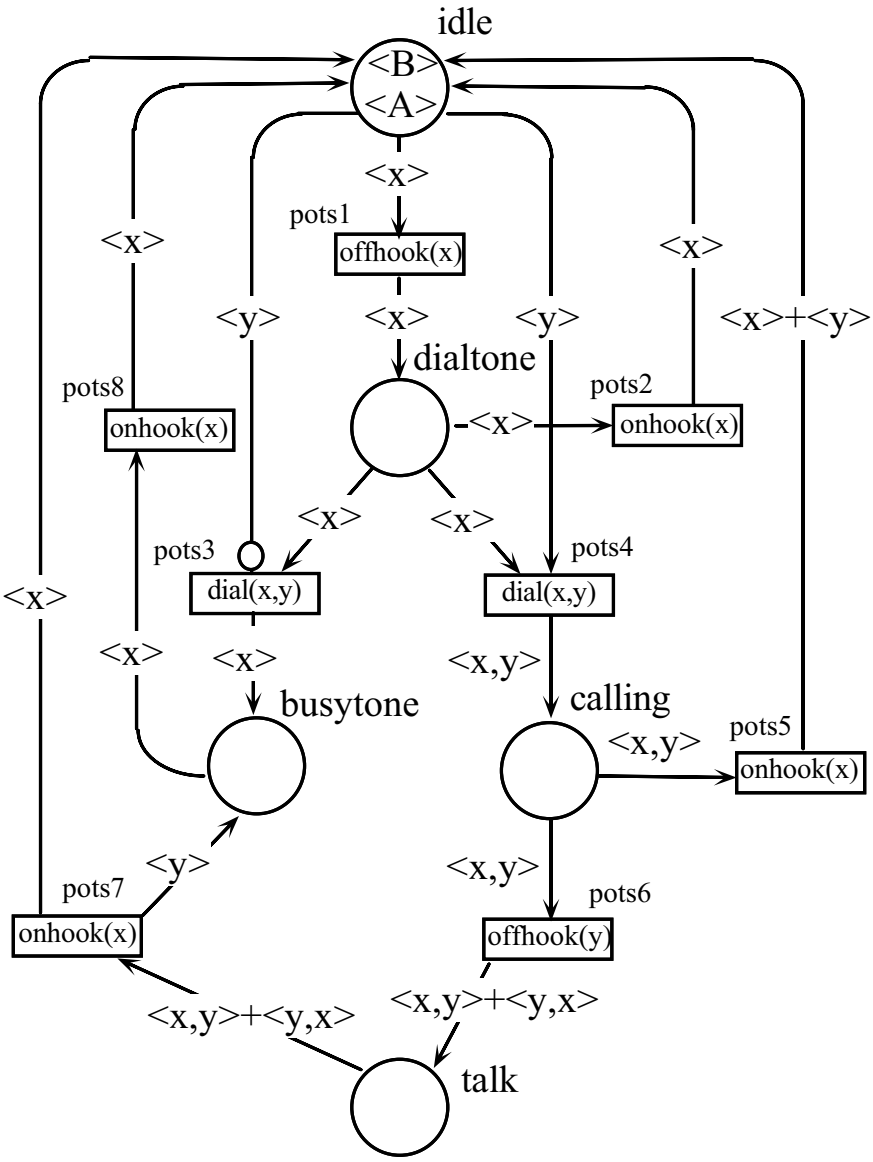


Figure 5.3: Service specification net for POTS specification

Lemma 5.11 *For a given service specification $\mathcal{S}$ and a service specification net $N(\mathcal{S})$ for $\mathcal{S}$, there exists one-to-one correspondence between a set of reachable markings of $\mathcal{N}(\mathcal{S})$ and a set of reachable states of $\mathcal{S}$.*

Proof: Let SS be a set of all states on $\mathcal{S}$ and MS be a set of all markings on $\mathcal{N}(\mathcal{S})$. Then, define a bijection $\tau : SS \rightarrow MS$ such that for $s \in SS$ and $M \in MS$, $\langle a_1, ..., a_k \rangle \in M(p)$ iff s includes $p(a_1, ..., a_k)$. Now, we prove Lemma 5.11 by induction.

By Definition 5.9(i), it is clear that $\tau(s_0) = M_0$.

Then, suppose that there exists a pair of sequences $s_0, s_1, ..., s_n$ of reachable states on $\mathcal{S}$ and $M_0, M_1, ..., M_n$ of reachable markings on $\mathcal{N}(\mathcal{S})$ such that $\tau(s_i) = M_i (0 \leq i \leq n)$. Next, let s_{n+1} be the next state of s_n generated by an application of rule r_i based on a substitution θ to s_n. Since $en(s, r_i, \theta)$ and $\tau(s_n) = M_n$, for any instance $p(c_1, ..., c_k)$ in $Pre[r_i\theta]$, $\langle c_1, ..., c_k \rangle \in M_n(p)$ holds. So, according to the enable condition of Definition 5.7 and Definition 5.9, transition t_i of $\mathcal{N}(\mathcal{S})$ is surely enabled for θ under M_n. Let $p(c_1, ..., c_k)$ and $q(d_1, ..., d_m)$ be any instances in the $Pre[r_i\theta]$ and $Post[r_i\theta]$, respectively. Suppose that t_i fires for θ under M_n and that the next marking M_{n+1} is generated. From Definition 5.7 and Definition 5.9, $M_{n+1}(p) = M_n(p) - \{\langle c_1, ..., c_k \rangle\}$ $M_{n+1}(q) = M_n(q) + \{\langle d_1, ..., d_m \rangle\}$, so $\langle c_1, ..., c_k \rangle \notin M_{n+1}(p)$ and $\langle d_1, ..., d_m \rangle \in M_{n+1}(q)$. On the other hand, by Definition 2.7, the application of r_i to s_n based on θ removes $p(c_1, ..., c_k)$ from s_n, and adds $q(d_1, ..., d_m)$ to s_n. Considering this, it is clear that s_{n+1} includes $p(a_1, ..., a_k)$ iff $\langle a_1, ..., a_k \rangle \in M_{n+1}(p)$. Hence, $\tau(s_{n+1}) = M_{n+1}$. By the induction, for any reachable state s_n, there exists exactly one marking M_n such that $\tau(s_n) = M_n$. □

Remark 5.12 For a pair of a state s and a marking M, we use the notation $s \equiv M$ iff the relation $\tau(s) = M$ holds.

Lemma 5.11 implies that $\mathcal{N}(\mathcal{S})$ is logically equivalent with respect to reachability analysis. We can completely simulate the behavior of the service specification on this net model.

Thus, we succeed in extracting the structural information of rule-based service specification as a Petri net structure.

Example 5.13 Figure 5.3 shows a service specification net obtained from the service specification of POTS in Figure 2.2.

5.3 P-invariant Method

After mapping the rule-based description into the Petri net, we can utilise the powerful analysis method of Petri net, called the P-invariant method * . Intuitively, the P-invariant method is used to find equations that are satisfied for all reachable markings (i.e., states) of a considered Petri net. The basic idea is that we first assign a *weight* to each place, and then make a weighted sum of tokens on all places. If the weights are nicely chosen, the weighted sum of the tokens is equally reserved before and after the transition firing. This forms an *invariant* which always holds on all reachable states. In the coloured Petri nets, the weight on each place is specified in terms of linear function.

Before explaining the P-invariant method, we introduce the notion of weighted-sets[32] over the colour sets.

Definition 5.14 [32] Let p be a place of $\mathcal{N}(\mathcal{S})$, and $C(p)$ be a colour set of p. Then a *weighted-set* w over $C(p)$ is defined as $w = \bigcup_{c \in C(p)} w(c)c$, where $w(c)$ is an integer specified for each c. The set of all weighted-sets of $C(p)$ is denoted by $C(p)_{ws}$. For $w_1, w_2 \in C(p)_{ws}$, $w_1 = w_2$ iff $\forall c \in C(p) \;\; w_1(c) = w_2(c)$.

Addition and *subtraction* of weighted-sets are defined in the following way, for any $w_1, w_2 \in C(p)_{ws}$:

* We must distinguish the term P-invariant from the invariant property or the invariant formula defined in Chapter 2. The invariant formula should be given manually based on the knowledge of the designer, while the P-invariant is mathematically calculated from the Petri net structure.

(i) $w_1 + w_2 = \bigcup_{c \in C(p)}(w_1(c) + w_2(c))c$

(ii) $w_1 - w_2 = \bigcup_{c \in C(p)}(w_1(c) - w_2(c))c$

For transition t, the weighted-set over $C(t)$ and $C(t)_{ws}$ are similarly defined.

Example 5.15 Consider place *idle* in Figure 5.3. Then $C(idle) = \{\langle A\rangle, \langle B\rangle\}$. A weighted-set over $C(idle)$ can be described as $w_1 = \{2\langle A\rangle, -3\langle B\rangle\}$, or $w_2 = \{-3\langle A\rangle, 3\langle B\rangle\}$. Then, $w_1 + w_2 = \{-\langle A\rangle\}$, and $w_1 - w_2 = \{5\langle A\rangle, -6\langle B\rangle\}$.

Definition 5.16 [32] Let $\mathcal{N}(\mathcal{S})$ be a service specification net with u places and v transitions, and let $p \in P$, $t \in T$ with $(p,t) \notin H$. Then, $W(p,t)$ (or $W(t,p)$) is a linear function $[C(t) \to C(p)]$ such that $\forall c = \langle x_1, ..., x_l\rangle \in C(t)$, $W(p,t)(c) = L_a(p,t)\theta(c)$ (or, $W(t,p)(c) = L_a(t,p)\theta(c)$, respectively). In our discussion, we consider only four kinds of linear functions defined as follows:

(a) identity function *id*: $id\langle\alpha,\beta\rangle = \langle\alpha,\beta\rangle$ and $id\langle\alpha\rangle = \langle\alpha\rangle$.

(b) zero function *o*: $o\langle\alpha,\beta\rangle = 0$ and $o\langle\alpha\rangle = 0$.

(c) projection functions p_1 and p_2: $p_1\langle\alpha,\beta\rangle = \langle\alpha\rangle$, $p_2\langle\alpha,\beta\rangle = \langle\beta\rangle$.

(d) reverse function r: $r\langle\alpha,\beta\rangle = \langle\beta,\alpha\rangle$

We assume that for any linear function f except for zero function, $f \pm o = o \pm f = f$, $f \cdot o = o \cdot f = o \cdot o = o$, $id \cdot f = f \cdot id = f$. Also, we assume that for the reverse function r, $p_1 \cdot r = p_2$ and $p_2 \cdot r = p_1$, since $p_1(r(\langle\alpha,\beta\rangle)) = p_1(\langle\beta,\alpha\rangle) = \langle\beta\rangle = p_2(\langle\alpha,\beta\rangle)$, and vice versa.

Then the *incident matrix* A of $\mathcal{N}(\mathcal{S})$ is the $u \times v$ matrix defined as follows.

$$A[p,t] = W(t,p) - W(p,t)$$

Then u-dimensional vector Y such that $Y * A = \mathbf{0}$ is called *P-invariant* of $\mathcal{N}(\mathcal{S})$, where $*$ is a formal product operation of matrix[48, 32].

Example 5.17 Consider the service specification net shown in Figure 5.3. Then the incident matrix A of this net is

$$A = \begin{array}{c} \\ idle \\ dialtone \\ calling \\ busytone \\ talk \end{array} \begin{array}{c} \begin{array}{cccccccc} pots_1 & pots_2 & pots_3 & pots_4 & pots_5 & pots_6 & pots_7 & pots_8 \end{array} \\ \begin{bmatrix} -id & id & o & -p_2 & p_1+p_2 & o & p_1 & id \\ id & -id & -p_1 & -p_1 & o & o & o & o \\ o & o & o & id & -id & -id & o & o \\ o & o & p_1 & o & o & o & p_2 & -id \\ o & o & o & o & o & r+id & -r-id & o \end{bmatrix} \end{array}$$

For example, let us consider place $idle$ and transition $pots_1$. For any colour, for example $\langle A\rangle \in C(pots_1)$, it is clear that $L_a(idle, pots_1)\theta\langle A\rangle = \langle A\rangle$, and that $L_a(pots_1, idle)\theta\langle A\rangle = 0$. Therefore, $W(idle, pots_1) = id$ and $W(pots_1, idle) = o$, and thus $A[idle, pots_1] = W(pots_1, idle) - W(idle, pots_1) = o - id = -id$. Next, consider place $dialtone$ and transition $pots_4$. For any colour, for example $\langle B, D\rangle \in C(pots_4)$, $L_a(dialtone, pots_4)\theta\langle B, D\rangle = \langle B\rangle$, and $L_a(pots_4, dialtone)\theta\ \langle B, D\rangle = 0$. Hence, $W(dialtone, pots_4) = p_1$ and $W(pots_4, dialtone) = o$, and thus $A[dialtone, pots_4] = W(pots_4, dialtone) - W(dialtone, pots_4) = o - p_1 = -p_1$.

The following vector Y is a P-invariant of $\mathcal{N}(\mathcal{S})$ since a relation $Y * A = \mathbf{0}$ holds.

$$Y = \begin{array}{ccccc} idle & dialtone & calling & busytone & talk \\ (id & id & p_1+p_2 & id & p_1) \end{array}$$

$$\begin{aligned} Y * A &= (id \cdot -id + id \cdot id + (p_1+p_2)\cdot o + id \cdot o + p_1 \cdot o, \\ &\quad id \cdot id + id \cdot -id + (p_1+p_2)\cdot o + id \cdot o + p_1 \cdot o, \\ &\quad \cdots\cdots \\ &\quad , id \cdot id + id \cdot o + (p_1+p_2)\cdot o + id \cdot -id + p_1 \cdot o) \\ &= (-id + id + o + o + o, id - id + o + o + o, \cdots, id + o + o - id + o) \\ &= (o, o, \cdots, o) = \mathbf{0} \end{aligned}$$

We should note that the P-invariant can be obtained from the incident matrix that depends on only the net structure(i.e., it can be obtained independently of the marking).

Remark 5.18 The the calculation of the P-invariants is not an easy task[3, 32] for a general coloured Petri net. However, as for our service specification net which is a subclass of the general one, there exists a relatively simple calculation method. The brief algorithm can be found in Appendix A.

The following theorem for the P-invariant is a well-known theorem to be used for checking reachability.

Theorem 5.19 *[32][48] Let Y be a P-invariant of the service specification net $\mathcal{N}(\mathcal{S})$. If a marking M is reachable from the initial marking M_0, then $Y * M^t = Y * M_0^t$.*

In $Y * M^t$, P-invariant Y makes a weighted sum of tokens by assigning a weight specified by a linear function to tokens on each corresponding place under M. With respect to this weighted sum, Theorem 5.19 implies that Y prescribes an equation which always holds for any reachable marking M from M_0.

Using the contraposition of Theorem 5.19, we can check the reachability of any arbitrary state s by the following rule:

Proposition 5.20 *Let s and s_0 be states of the service specification $\mathcal{S}$, and let M and M_0 be markings of the service specification net $\mathcal{N}(\mathcal{S})$, where $s \equiv M$ and $s_0 \equiv M_0$. Then,*

$$Y * M^t \neq Y * M_0^t \qquad \Rightarrow \qquad \neg(s_0 \rightarrow^* s)$$

Remark 5.21 The equation $Y * M^t = Y * M_0^t$ is only a necessary condition for any reachable marking M. Hence, even if $Y * M^t = Y * M_0^t$ holds, we cannot conclude, in general, that M is reachable from M_0.

Using Proposition 5.20, we can check the reachability of any arbitrary state s. That is, if the P-invariant equation does not hold, then we can conclude that s is *not* reachable

from s_0. Note that this checking can be performed in a *static* way without generating any reachability graph.

Example 5.22 Consider again the service specification net in Figure 5.3 and its P-invariant Y.

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ Y = & (id & id & p_1 + p_2 & id & p_1) \end{array}$$

Also, consider the following markings M_0 and M. Note that M is reachable from M_0. We can easily demonstrate this in the same way performed in Example 5.8.

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ M_0 = & (\{\langle A\rangle, \langle B\rangle\} & \emptyset & \emptyset & \emptyset & \emptyset) \end{array}$$

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ M = & (\emptyset & \emptyset & \{\langle A, B\rangle\} & \emptyset & \emptyset) \end{array}$$

Let us evaluate $Y * M_0^t$. In the evaluation, we make a weighted sum of tokens by applying each linear function to tokens in the corresponding place [†] .

$$\begin{aligned} Y * M_0^t &= id\{\langle A\rangle, \langle B\rangle\} + id\emptyset + (p_1 + p_2)\emptyset + id\emptyset + p_1\emptyset \\ &= \{id\langle A\rangle, id\langle B\rangle\} + \emptyset + \emptyset + \emptyset + \emptyset \\ &= \{\langle A\rangle, \langle B\rangle\} \end{aligned}$$

Similarly, as for $Y * M^t$,

$$\begin{aligned} Y * M^t &= id\emptyset + id\emptyset + (p_1 + p_2)\{\langle A, B\rangle\} + id\emptyset + p_1\emptyset \\ &= \emptyset + \emptyset + \{p_1\langle A, B\rangle\} + \{p_2\langle A, B\rangle\} + \emptyset + \emptyset \\ &= \{\langle A\rangle\} + \{\langle B\rangle\} \\ &= \{\langle A\rangle, \langle B\rangle\} = Y * M_0^t \end{aligned}$$

[†] In [32], the definition of linear functions is extended for the weighted-set. For any linear function f, we define $f(\{\langle \alpha_1, \beta_1\rangle, ..., \langle \alpha_k, \beta_k\rangle\}) = \{f\langle \alpha_1, \beta_1\rangle, ..., f\langle \alpha_k, \beta_k\rangle\}$.

Next, we consider the following marking M':

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ M' = & (\{\langle A\rangle, \langle B\rangle\} & \emptyset & \emptyset & \langle A\rangle & \emptyset) \end{array}$$

Then,

$$\begin{aligned} Y * M'^t &= id\{\langle A\rangle, \langle B\rangle\} + id\emptyset + (p_1 + p_2)\emptyset + id\{\langle A\rangle\} + p_1\emptyset \\ &= \{\langle A\rangle, \langle B\rangle\} + \emptyset + \emptyset + \{\langle A\rangle\} + \emptyset \\ &= \{2\langle A\rangle, \langle B\rangle\} \neq Y * M_0^t \end{aligned}$$

So, we can conclude M' is not reachable from M_0 according to Theorem 5.19.

5.4 Interaction Detection Algorithm PINV

5.4.1 Outline

Here, we discuss how to apply the P-invariant method to interaction detection. As mentioned before, the P-invariant method allows the reachability checking of a state. Therefore, if it is possible to determine the *undesirable candidates* in Definition 2.13. Then we can apply the P-invariant method to the candidate states to check their reachability. Based on this scheme, we try to implement the detection algorithm called PINV.

Figure 5.4 shows the approach, schematically. In the figure, U, RS and UND are the same as those in Figure 2.3, and the set PEQ denotes the set of all states (i.e., marking) M satisfying the P-invariant equation $Y * M^t = Y * M_0^t$. Note that PEQ is a superset of RS (See Remark 5.21). First, we construct the set UND as the candidates of undesirable states. Then, we delete states not satisfying $Y * M^t = Y * M_0^t$ from UND since such states are guaranteed to be unreachable by the P-invariant. This approach requires no state space generation using a reachability graph, therefore, we expect a drastic cost reduction for interaction detection.

However, there are two drawbacks. Firstly, the detection result will not be optimal, whereas Algorithms EXH and SYM guarantee optimal detection. That is, since the P-invariant method gives us only a necessary condition, Algorithm PINV may identify some interactions which do not actually occur. Secondly, it is not always possible to determine undesirable candidate states UND nicely. This means that Algorithm PINV may not cover all classes of interactions.

Taking these issues into account, we propose Algorithm PINV for the following two classes of interactions: non-determinism and violation of invariant (See Chapter 2).

5.4.2 Candidate Non-deterministic States

We consider how to construct the candidate non-deterministic states. It is not necessary here to care if the candidate states are reachable from the initial state. Instead, we pay an attention to the structure of the rules.

The non-deterministic interactions arise when two or more rules are simultaneously enabled on a state with respect to an identical instance of event. Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification, and let $r_i, r_j \in R$ be a pair of rules which have the same event symbol $e \in E$. By Definition 2.13, if a state s is a non-deterministic state, s requires the following condition with respect to r_i and r_j.

Condition NDT: There exists a pair of substitution θ_i and θ_j such that

(a) $Ev[r_i\theta_i] = Ev[r_j\theta_j]$ and

(b) s includes both $Pre[r_i\theta_i]$ and $Pre[r_j\theta_j]$ (i.e., both $Pre[r_i\theta_i]$ and $Pre[r_j\theta_j]$ are true at s.).

By means of Condition NDT, we generate s as a candidate in the following way.

Procedure P1 :

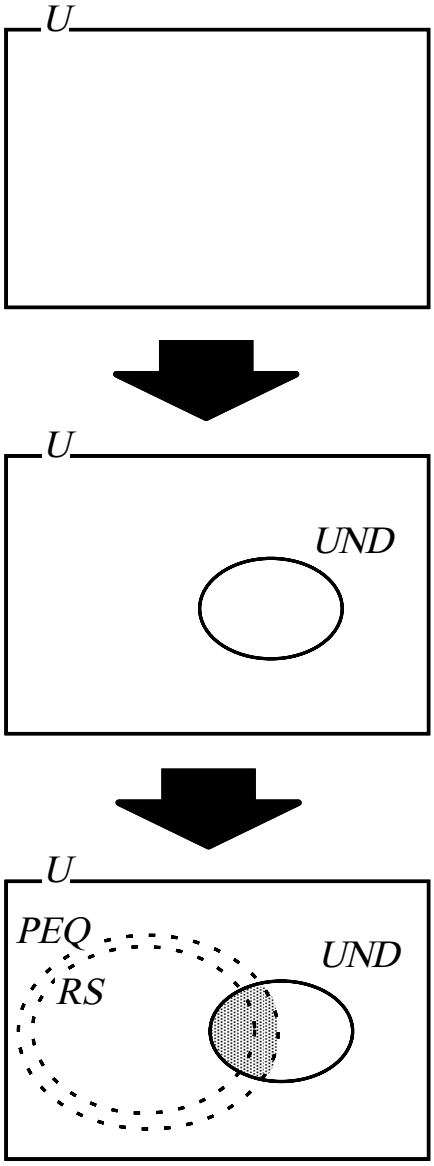


Figure 5.4: Conceptual overview of PINV

Step1: Select any pair of rules r_i and r_j which have the same event symbol.

Step2: Apply a pair of substitution θ_i and θ_j such that $Ev[r_i\theta_i] = Ev[r_j\theta_j]$ to r_i and r_j, respectively.

Step3: Let s be a conjunction of $Pre[r_i\theta_i]$ and $Pre[r_j\theta_j]$, that is $s := Pre[r_i\theta_i] \wedge Pre[r_j\theta_j]$.

The resultant s clearly satisfies Condition NDT.

Example 5.23 Let us apply Procedure P1 to rules $pots_1$ and $pots_6$ in Figure 2.2 whose event symbols are identical to $offhook$:

$$\begin{array}{lllll} pots_1: & & idle(x) & [offhook(x)] & dialtone(x) \\ pots_6: & calling(x,y) & [offhook(y)] & talk(x,y), talk(y,x) & \end{array}$$

In order to make both instances of the event identical, we apply $\theta_1 = \langle x|A\rangle$ to $pots_1$ and $\theta_2 = \langle x|B, y|A\rangle$ to $pots_6$. Then, by combining both pre-conditions, we obtain a candidate non-deterministic state s, as follows:

$$s = idle(A), calling(B, A)$$

This situation is: " Suppose that A is idle, and B is calling A. At this time, if A offhooks, then should A receive dialtone or talk with B?" If s is reachable from s_0, s is a non-deterministic state.

Due to the negations in pre-condition, some rules r_i and r_j cannot become enabled simultaneously. We define such a pair of rules r_i and r_j to be mutually exclusive.

Definition 5.24 Let r_i and r_j be rules with the same event symbol, and let θ_i and θ_j be substitutions such that $Ev[r_i\theta_i] = Ev[r_j\theta_j]$. Then, r_i and r_j are *mutually exclusive* iff for all θ_i and θ_j, there exists an instance of predicate $p(a_1, ..., a_k)$ such that $p(a_1, ..., a_k)$ is in $Pre[r_i\theta_i]$ and that $\neg p(a_1, ..., a_k)$ is in $Pre[r_j\theta_j]$.

The rules r_i and r_j which are mutually exclusive never become enabled simultaneously with the same instance of event. Thus, Condition NDT never holds with respect to r_i and r_j. Hence, we need not generate the candidates from such r_i and r_j.

Example 5.25 Consider a pair of rules $pots_3$ and $pots_4$ in Figure 2.2:

$$\begin{aligned} pots_3: \quad & dialtone(x), \neg idle(y) \quad [dial(x,y)] \quad busytone(x) \\ pots_4: \quad & dialtone(x), idle(y) \quad [dial(x,y)] \quad calling(x,y) \end{aligned}$$

No matter how nicely we choose the substitutions θ and θ' such that $Ev[pots_3\theta] = Ev[pots_4\theta'] = dial(a,b)$, where $a, b \in U$, both $Pre[pots_3\theta]$ and $Pre[pots_4\theta']$ never take true value simultaneously. Since $\neg idle(b) \in Pre[pots_3\theta]$ and $idle(b) \in Pre[pots_4\theta]$, $pots_3$ and $pots_4$ are mutually exclusive.

Although any state s obtained by Procedure P1 satisfies Condition NDT, all other states s' in which s is true also satisfy Condition NDT. Let us consider again $s = idle(A), calling(B,A)$ in Example 5.23. Then, all of the following states also satisfy Condition NDT.

$$\begin{aligned} s &= idle(A), calling(B,A) \\ s_1 &= idle(A), calling(B,A), dialtone(A) \\ s_2 &= idle(A), calling(B,A), idle(B), dialtone(B) \\ s_3 &= idle(A), calling(B,A), calling(A,B) \\ s_4 &= idle(A), calling(B,A), talk(A,B), talk(A,B) \\ &\dots \end{aligned}$$

In order to deal with such states efficiently, we define the notion of *wildcard*. Intuitively, it regards instances of predicates other than those in s to be *don't care*.

Definition 5.26 Let $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. For each predicate symbol $p \in P$, a *wildcard* of p, denoted by $\Sigma p(x_1, ..., x_k), x_i \in V$, is defined to be a set of

any instances $p(a_1, ..., a_k), a_i \in U$'s. A *wildcard extension* of a state s, denoted by $WX(s)$ is defined as a state obtained by combining s and $\Sigma p(x_1, ..., x_k), x_i \in V$ for all $p \in P$.

Example 5.27 Let us consider states s, s_1, s_2, s_3 and s_4 in the above example. We can describe all states in which s is true by means of the wildcard extension:

$$WX(s) = idle(A), calling(B, A), \Sigma idle(x), \Sigma dialtone(x), \Sigma calling(x, y), \Sigma busytone(x), \Sigma talk(x, y)$$

For example, using $WX(s)$, we can represent s_4 by regarding that

$$\Sigma talk(x, y) = \{talk(A, B), talk(B, A)\}$$

and others are empty sets.

Finally, we present the candidates decision procedure in Figure 5.5. The output of Procedure NDT is the set C_{NDT} of candidate non-deterministic states.

5.4.3 Candidate Violating States

The candidate violating states are easily determined as the states at which the given invariant formula $I_{\mathcal{S}}$ is not respected. In other words, it is identified as a set of states satisfying $\neg I_{\mathcal{S}}$. As shown in Chapter 3, we assume that $\neg I_{\mathcal{S}}$ is given by the sum-of-product form:

$$\neg I_{\mathcal{S}} = t_1 \vee t_2 \vee ... \vee t_h$$

where $t_i = p_1 \wedge p_2 \wedge ... \wedge p_q$ and p_i is a predicate $p_i(x_1, ..., x_k)$ or its negation. If at least one of terms t_i's is satisfied at s for some θ, then we can conclude $s \not\models I_{\mathcal{S}}$. Therefore, we determine a candidate for each t_i. By applying possible substitution θ's to each t_i, we can obtain a set of candidates s's such that $s = p_1\theta, p_2\theta, ..., p_q\theta$. After that, we apply the wildcard extension to s as in the candidate non-deterministic states.

We present the candidates decision procedure in Figure 5.6. The output of Procedure VLT is the set C_{VLT} of candidates violating states.

Candidate Decision Procedure NDT

Input: $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$

Output: $C_{NDT} = \{WX(s) |\ s$ satisfies Condition NDT $\}$

Procedure:

$C_{NDT} := \emptyset$;

for each $r_i, r_j \in R$ with the same event symbol e {

  if (r_i and r_j are mutual-exclusive) then break;

  else

    for all θ_i, θ_j s.t. $Ev[r_i\theta_i] = Ev[r_j\theta_j]$ {

      $s := Pre[r_i\theta_i], Pre[r_j\theta_j]$;

      add $WX(s)$ to C_{NDT};

    }

}

Figure 5.5: Candidate decision procedure NDT

Example 5.28 Consider again the invariant formula I in Example 2.12.

$$I = \neg idle(x) \vee \neg busytone(x)$$

Then we have its negation of I.

$$\neg I = idle(x) \wedge busytone(x)$$

By applying possible substitutions $\theta_1 = \langle x|A\rangle$ and $\theta_2 = \langle x|B\rangle$ to $\neg I$, we can obtain two candidates s and s' as follows:

$$\begin{aligned} s &= idle(A), busytone(A) \\ s' &= idle(B), busytone(B) \end{aligned}$$

Clearly, $s \nvdash I$ and $s' \nvdash I$ hold.

5.4.4 Algorithm PINV

Now, we are ready to present Algorithm PINV. For given service specifications $\mathcal{S}_1, \mathcal{S}_2$ and invariant formulas $I_{\mathcal{S}1}, I_{\mathcal{S}2}$, we first construct the service specification net $\mathcal{N}(\mathcal{S})$ according to Definition 5.9, and then calculate P-invariant Y [‡] . Next, we obtain candidates of non-deterministic and violating states by means of Procedures NDT and VLT.

After that, for each candidate, we check its reachability by evaluating the P-invariant equation. Note that each candidate forms a wildcard extension $WX(s)$ of a state s. So, we first define the wildcard extension of a marking M.

Definition 5.29 Let $\mathcal{N}(\mathcal{S}) = \langle U, V, P, E, T, F, H, L_a, L_t, M_0\rangle$ be a service specification. For each place $p \in P$, a *wildcard* of p, denoted by $\Sigma\langle x_1, ..., x_k\rangle, x_i \in V$, is defined to be

[‡] In general, there may be several *basis* Y's of some P-invariants from which any P-invariant can be constructed by means of their linear combinations[32][48]. In such a case, apply the following for each basis Y of some P-invariants.

Candidate Decision Procedure VLT

Input: $\mathcal{S} = \langle U, V, P, E, R, s_0 \rangle$, $I_{\mathcal{S}}$

Output: $C_{VLT} = \{WX(s) |\ s \not\models I_{\mathcal{S}}\ \}$

Procedure:

/* $\neg I_{\mathcal{S}} = t_1 \vee t_2 \vee ... \vee t_h$ */

$C_{VLT} := \emptyset$;

for each $t_i = p_1 \wedge p_2 \wedge ... \wedge p_q$ {

  for all θ {

    $s := p_1\theta, p_2\theta, ..., p_q\theta$;

    add $WX(s)$ to C_{VLT};

  }

}

Figure 5.6: Candidate decision procedure VLT

a set of any tokens $\langle a_1, ..., a_k \rangle, a_i \in U$'s. A *wildcard extension* of a marking M, denoted by $WX(M)$ is defined as a marking M' such that $M'(p) = M(p) \cup \{\Sigma\langle x_1, ..., x_k \rangle\}$ for all $p \in P$.

Then, for each $WX(M)$ such that $M \equiv s$, we explain how to evaluate the P-invariant equation $Y * WX(M)^t = Y * M_0^t$. Let ρ be any function which assigns a set of tokens to each wildcard of $WX(M)$. If we can find no such ρ that makes the equation $Y * WX(M)^t = Y * M_0$ hold, we conclude $Y * WX(M)^t \neq Y * M_0$ by Proposition 5.20. That is, any state s characterized by $WX(s)$ is not reachable from s_0. So, we can conclude that interaction is *not detected* for such states. Therefore, we delete $WX(s)$ from the candidates.

Otherwise, if we find such ρ, we cannot derive any decision on reachability of s's characterized by $WX(s)$ (See Remark 5.21). So, we should say that interaction is *suspected* for some states characterized by $WX(s)$.

Example 5.30 Let us evaluate $WX(s)$ in Example 5.27, which is a candidate non-deterministic state. Based on this, we get the following $W(M)$:

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ WX(M) = & (\{\langle A \rangle, \Sigma\langle x_1 \rangle\}, & \{\Sigma\langle x_2 \rangle\}, & \{\langle B, A \rangle, \Sigma\langle x_3, x_4 \rangle\}, & \{\Sigma\langle x_5 \rangle\}, & \{\Sigma\langle x_6, x_7 \rangle\}) \end{array}$$

Now, let us apply the following P-invariant Y in Example 5.17

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & talk \\ Y = & (id & id & p_1 + p_2 & id & p_1) \end{array}$$

We recall from Example 5.17:

$$Y * M_0^t = \{\langle A \rangle, \langle B \rangle\}$$

Next, evaluate $Y * WX(M)^t$ as follows:

$$Y * WX(M)^t = id(\{\langle A \rangle, \Sigma\langle x_1 \rangle\}) + id(\{\Sigma\langle x_2 \rangle\}) + (p_1 + p_2)(\{\langle B, A \rangle, \Sigma\langle x_3, x_4 \rangle\}) +$$

$$
\begin{aligned}
& id(\{\Sigma\langle x_5\rangle\}) + p_1(\{\Sigma\langle x_6, x_7\rangle\}) \\
= \; & \{\langle A\rangle, \Sigma\langle x_1\rangle\} + \{\Sigma\langle x_2\rangle\} + p_1(\{\langle B, A\rangle, \Sigma\langle x_3, x_4\rangle\}) + \\
& p_2(\{\langle B, A\rangle, \Sigma\langle x_3, x_4\rangle\}) + \{\Sigma\langle x_5\rangle\} + \{\Sigma\langle x_6\rangle\} \\
= \; & \{\langle A\rangle, \Sigma\langle x_1\rangle\} + \{\Sigma\langle x_2\rangle\} + \{\langle B\rangle, \Sigma\langle x_3\rangle\} + \{\langle A\rangle, \Sigma\langle x_4\rangle\} + \\
& \{\Sigma\langle x_5\rangle\} + \{\Sigma\langle x_6\rangle\} \\
= \; & \{2\langle A\rangle, \langle B\rangle, \Sigma\langle x_1\rangle, \Sigma\langle x_2\rangle, \Sigma\langle x_3\rangle, \Sigma\langle x_4\rangle, \Sigma\langle x_5\rangle, \Sigma\langle x_6\rangle\}
\end{aligned}
$$

For this, no matter how nicely we choose the assignment of tokens such as $\langle A\rangle, \langle B\rangle$ to $\Sigma\langle x_i\rangle$'s, the equation $Y * WX(M)^t = Y * M_0^t$ never holds. Therefore, we can conclude that no state characterized by $WX(s)$ is reachable from s_0, and that they are not non-deterministic states, actually. Note that this checking is performed without generating a reachablity graph. Thus, Algorithm PINV achieves a static verification.

As for the interaction of non-deterministic states and violating states, the following proposition characterizes Algorithm PINV.

Proposition 5.31 *The following properties are satisfied for Algorithm PINV:*

$$
\begin{array}{rcl}
\mathcal{S}_1 \textit{ interacts with } \mathcal{S}_2. & \Rightarrow & PINV \textit{ returns ``Suspected''} \\
\mathcal{S}_1 \textit{ does not interact with } \mathcal{S}_2. & \Leftarrow & PINV \textit{ returns ``Not detected''}
\end{array}
$$

Proof: Straight forward from Lemma 5.11, Theorem 5.19 and Proposition 5.20. □

Remark 5.32 Note that Algorithm PINV never misses any interaction which actually occurs. However, it may identify some interactions which do not actually occur.

The evaluation of Algorithm PINV will be presented in Chapter 6.

5.5 Related Works

There are several static methods proposed for interaction detection. Kimbler proposed a notion of *interaction filtering* [38] that makes a rough pre-evaluation of the interaction-

Detection Algorithm PINV

Input: $\mathcal{S}_1, \mathcal{S}_2, I_{\mathcal{S}1}, I_{\mathcal{S}2}$

Output: "Suspected" or "Not detected" (only for non-deterministic and violating states)

Procedure:

$\mathcal{S} = \mathcal{S}_1 \oplus \mathcal{S}_2$; $I_{\mathcal{S}} = I_{\mathcal{S}1} \wedge I_{\mathcal{S}2}$;

Phase 0: Construct $\mathcal{N}(\mathcal{S}_1 \oplus \mathcal{S}_2)$;

Calculate P-invariant Y of $\mathcal{N}(\mathcal{S}_1 \oplus \mathcal{S}_2)$;

Phase 1: Obtain C_{NDT} and C_{VLT} by Procedures NDT and VLT;

$C := C_{NDT} \cup C_{VLT}$

Phase 2: For each $WX(s) \in C$

If $Y * WX(M)^t \neq Y * M_0^t$ s.t. $M \equiv s$;

then delete $WX(s)$ from C;

If $(C = \emptyset)$ return("Not detected");

else return("suspected");

Figure 5.7: Detection algorithm PINV

prone service combination before the detection process. Keck [35] presented a tool that identifies interaction-prone service scenarios based on heuristical criteria. Thomas [55] and Heisel [28] also proposed heuristical methods to find overlapping guards for non-deterministic interactions. All of the static methods above are heuristic-based methods. Hence, the detection quality and coverage of the methods deeply depend on the knowledge of the verifier or the knowledge database in the verification environment. Yoneda et al. [62] presented a guideline of static verification using description method State Transition Rules (STR)[29][27]. However, no evaluation has yet been conducted for detection coverage and redundancy.

Also, there are some Petri net based methods for interaction detection. Capellman et al.[16][17] proposed a service description method by means of Product Nets, a high-level variant of Petri net. In order to avoid the state explosion, Nitsche [50] proposed an abstraction method of reachable state space by means of homomorphisms for Capellman's framework. Kawarazaki et al. [34] proposed a method using T-invariants of Petri nets [48] for the guided search of the reachable state exploration. Note that these Petri net based approaches are dynamic algorithms that examine the reachability analysis.

Chapter 6

Experimental Evaluation

6.1 Introduction

In order to evaluate the effectiveness of the proposed algorithms, we have conducted the experimental evaluation through interaction detection for practical services.

We evaluate the algorithms by the following three metrics: (a) detection quality, (b) performance and (c) scalability. In the detection quality measurement, we investigate whether the algorithms can detect the interactions correctly or not. Then, in performance measurement, we measure the space and time needed for each algorithm. Finally, scalability measurement is conducted in order to evaluate how many users and features the algorithms can scale.

The rule-based service specifications are prepared based on ITU-T recommendations [64] and Bellcore's documents [66]. We have selected eight services, and perform the interaction detection for any pair of services.

Also, we have developed a software, called *Service specification VALidator*— SVAL in short, for the experiment. Software SVAL is written in the C language, comprises about 7000 lines of code, and can execute automatic interaction detection based on any of the algorithms EXH, SYM and PINV for a given rule-based service specification. When the

interaction occurs, that is, some undesirable states are detected, SVAL returns not only "detected" (or "suspected") but also the execution trace to the undesirable state. It also outputs the number of states, the number of edges, and amount of execution time elapsed.

SVAL can quit its execution as soon as an undesirable state is identified, (i.e., so-called on-the-fly verification [30]), however, we do not use the on-the-fly mode. We want to perform the worst case analysis, which is independent of the searching strategy and so on.

The rest of this chapter is organised as follows: In the next section, we explain the service specifications prepared for the experiment. In Section 6.3, the evaluation experiment is conducted. In Section 6.4, we summarize the results and discuss characteristics of the three algorithms EXH, SYM and PINV.

6.2 Service Specifications

We prepare the rule-based service specifications for the experiment. From ITU-T recommendation [64] (*ITU-T Recommendations Q.1200 Series* — Intelligent Network Capability Set 1 (CS1)) and Bellcore's feature standards [66] (*Bellcore* — LSSGR Features Common to Residence and Business Customers I, II, III), we have selected the following eight services (features):

CW: Call Waiting

CF: Call Forwarding

DC: Direct Connect

DO: Denied Origination

DT: Denied Termination

OCS: Originating Call Screening

TCS: Terminating Call Screening

EMG: Emergency call

For each of the eight services, we have created a rule-based service specification (We present them written in the SVAL language in Appendix B). In the following, we describe the fundamental functionality of each service, and then we attempt to provide a reasonable invariant property intended to be satisfied.

Call Waiting (CW): This service allows the subscriber to receive an additional call while he is talking. Suppose that x subscribes to CW. Even when x is busy talking with y, x can receive an additional call from a third party z.

From this fundamental functionality, there is no invariant property respected for CW. Therefore, we give an invariant formula $I_{CW} = true$.

Call Forwarding (CF): This service allows the subscriber to have his incoming calls forwarded to another number. Suppose that x subscribes to CF and that x specifies z to be a forwarding address. Then, any incoming call to x is automatically forwarded to z.

From this fundamental functionality, there is no invariant property respected for CF. Therefore, we give an invariant formula $I_{CF} = true$.

Originating Call Screening (OCS): This service allows the subscriber to specify that outgoing calls be either restricted or allowed according to a screening list. Suppose that x subscribes to OCS and that x puts y in the OCS screening list. Then, any outgoing call to y from x is restricted, while any other call to z from x is allowed. Suppose that x receives dialtone. At this time, even if x dials y, x receives busytone instead of calling y.

From this fundamental functionality, a reasonable invariant property is considered to be "If x puts y in the OCS screening list, x is never calling y at any time". Therefore, we give an invariant formula $I_{OCS} = \neg OCS(x, y) \vee \neg calling(x, y)$.

Terminating Call Screening (TCS): This service allows the subscriber to specify that incoming calls be either restricted or allowed according to a screening list. Suppose that x subscribes to TCS and that x puts y in the TCS screening list. Then, any incoming call from y to x is restricted, while any other call from z to x is allowed. Suppose that y receives dialtone. At this time, even if y dials x, y receives busytone instead of calling x.

From this fundamental functionality, a reasonable invariant property is considered to be "If x puts y in the TCS screening list, y is never calling x at any time". Therefore, we give an invariant formula $I_{TCS} = \neg TCS(x, y) \vee \neg calling(y, x)$.

Denied Origination (DO): This service allows the subscriber to disable any call originating from the terminal. Only terminating calls are permitted. Suppose that x subscribes to DO. Then, any outgoing call from x is restricted. Even if x offhooks when the terminal is idle, x receives busytone instead of dialtone.

From this fundamental functionality, a reasonable invariant property is considered to be "If x subscribes to DO, x never receives dialtone at any time". Therefore, we give an invariant formula $I_{DO} = \neg DO(x) \vee \neg dialtone(x)$.

Denied Termination (DT): This service allows the subscriber to disable any call terminating at the terminal. Only originating calls are permitted. Suppose that x subscribes to DT. Then, any outgoing call from x is restricted. Even if another user y dials x, y receives busytone without calling x.

From this fundamental functionality, a reasonable invariant property is considered to be "If x subscribes to DT, y is never calling x at any time". Therefore, we give

an invariant formula $I_{DT} = \neg DT(x) \vee \neg calling(y, x)$.

Direct Connect (DC) This service is a so-called *hot line* service. Suppose that x subscribes to DC and that x specifies y as the destination address. Then, by only offhooking, x is directly calling y. It is not necessary for x to dial y.

From this fundamental functionality, there is no invariant property respected for DC. Therefore, we give an invariant formula $I_{DC} = true$.

Emergency call (EMG): This service is usually deployed on police and fire stations. In the case of an emergency incident, the call will be held even when the caller mistakenly onhooks. Suppose that x is a police station on which EMG is deployed, and that y has made a call to x and is now busy talking with x. Then, even when y onhooks, the call is on hold without being disconnected. Followed by that, if y offhooks, the held line reverts to a connected line and y can talk with x again. In order to disconnect the call, x has to onhook.

From this fundamental functionality, there is no invariant property respected for DC. Therefore, we give an invariant formula $I_{DC} = true$.

In the experiment, we put the following assumption.

Assumption 6.1 *The following properties are assumed in the experiment.*

(a) All users can subscribe to all services.

(b) At the initial state, all users are idle and no user subscribes to any service yet.

This assumption is quite reasonable for telecommunication services. In order to achieve Assumption 6.1(a), a pair of rules for the subscription registration and its withdrawal is added to each service specification. Also, as for Assumption 6.1(b), we specify the initial state as follows:

$$s_0 = idle(a_1), idle(a_2), ..., idle(a_n), yetX(a_1), yetX(a_2), ..., yetX(a_n),$$

where n is the number of users verified, and the predicate $yetX(a_1)$ means that user a_i does not subscribes to supplementary service X yet.

6.3 Experimental Evaluation

For the service specifications prepared in the previous section, we perform the feature interaction detection using three algorithms, EXH, SYM and PINV. The experimental evaluation is performed from the following three viewpoints.

Detection Quality: whether the algorithms can exactly identify all interactions or not.

Performance: how much time and space are needed for the algorithms.

Scalability: how many users and features can be scaled by the algorithms.

The experiments have been performed by the software package SVAL on the UNIX workstation Sun SS-UA1 with 334Mb memory.

6.3.1 Detection Quality

The primary objective here is to evaluate the *detection quality*. That is, we see the proposed algorithms can exactly identify all interactions.

As shown in Chapter 3, it is clear that Algorithm EXH surely detects all interactions since it enumerates all possible reachable states. Hence, we can regard the interactions detected by Algorithm EXH as the *correct answer*. By comparing the detection result obtained by each of the algorithms SYM and PINV with those by Algorithm EXH, we evaluate the detection quality of the algorithms. The closer to the correct answers the result is, the higher quality the algorithm has.

According to Proposition 4.17, the detection quality of Algorithm SYM is expected to be optimal. That is, it is expected that Algorithm SYM detects all interactions correctly.

On the other hand, Algorithm PINV performs the interaction detection based on only a necessary condition of the P-invariant method. According to Proposition 5.31, Algorithm PINV identifies all correct interactions as *suspected interactions*, however, it may identify some interactions which do not actually occur, theoretically speaking. So, our interest here is how many such interactions are wrongly suspected by Algorithm PINV.

Now, we start the evaluation. First, we check if each of eight specifications prepared in the previous section is safe by applying Algorithm EXH. As a result, we have found that all services except EMG are safe, while EMG contains the loop states as shown in Example 2.3 which is interaction of EMG with itself. Next, we have combined each pair of the remaining seven services, then tried to detect the interactions between any two services by means of the three specified algorithms. For all specifications, the number of users is assumed to be three.

Table 6.1 summarises the result. In the table, the columns DLK, LOP, NDT and VLT respectively show whether deadlock, loop, non-deterministic or violating states are identified (*detected* or *suspected*) or not (*none*). Since Algorithm PINV does not work for deadlock and loop states, the corresponding columns appear as *not available*(N/A).

As is expected, Algorithm SYM achieves the optimal interaction detection. That is, it detects all undesirable states detected by Algorithm EXH. Moreover, it returns *none* for any states not detected by Algorithm EXH.

It is interesting that all undesirable states suspected by Algorithm PINV are actual ones. It never wrongly suspects interactions which do not actually occur. This means that the necessary condition for Algorithm PINV works as if it is a necessary and sufficient condition for these practical service specifications. From this, we conclude that Algorithm PINV attains semi-optimal detection quality.

Table 6.1: Result for detection quality

Service	Algorithm EXH				Algorithm SYM				Algorithm PINV			
Spec. S	DLK	LOP	NDT	VLT	DLK	LOP	NDT	VLT	DLK	LOP	NDT	VLT
CW+CF	None	None	Detected	None	None	None	Detected	None	N/A	N/A	Suspected	None
CW+DC	None	None	None	None	None	None	None	None	N/A	N/A	None	None
CW+DT	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
CW+DO	None	None	None	None	None	None	None	None	N/A	N/A	None	None
CW+OCS	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
CW+TCS	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
CF+DC	None	None	None	None	None	None	None	None	N/A	N/A	None	None
CF+DT	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
CF+DO	None	None	None	None	None	None	None	None	N/A	N/A	None	None
CF+OCS	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
CF+TCS	None	None	Detected	Detected	None	None	Detected	Detected	N/A	N/A	Suspected	Suspected
DC+DT	None	None	None	Detected	None	None	None	Detected	N/A	N/A	None	Suspected
DC+DO	None	None	Detected	None	None	None	Detected	None	N/A	N/A	Suspected	None
DC+OCS	None	None	None	Detected	None	None	None	Detected	N/A	N/A	None	Suspected
DC+TCS	None	None	None	Detected	None	None	None	Detected	N/A	N/A	None	Suspected
DT+DO	None	None	None	None	None	None	None	None	N/A	N/A	None	None
DT+OCS	None	None	Detected	None	None	None	Detected	None	N/A	N/A	Suspected	None
DT+TCS	None	None	Detected	None	None	None	Detected	None	N/A	N/A	Suspected	None
DO+OCS	None	None	None	None	None	None	None	None	N/A	N/A	None	None
DO+TCS	None	None	None	None	None	None	None	None	N/A	N/A	None	None
OCS+TCS	None	None	Detected	None	None	None	Detected	None	N/A	N/A	Suspected	None
EMG	None	Detected	None	None	None	Detected	None	None	N/A	N/A	None	None

6.3.2 Performance

Next, we evaluate the performance of the algorithms. For each of the three algorithms, we investigate how much space and time is needed to perform the interaction detection. The measurement is performed in the same setting as in the previous experiment of detection quality.

As the metric of space for algorithms EXH and SYM, we adopt the numbers of nodes and edges of the reachability graphs FRG and SRG, respectively. This is because the size of the reachability graph is the dominant factor of space for both algorithms. As for Algorithm PINV, we select the number of undesirable candidate states determined in Phase 1 of Algorithm PINV.

On the other hand, the time is measured as the execution time of software SVAL. Since the execution time may depend on the implementation of SVAL, we specifically devised the programming so that Algorithm EXH can give the best performance, by means of several techniques such as data compression and hash searching [1, 30].

Table 6.2 shows the result. In the table, $|N|$ and $|T|$ respectively represent the number of nodes and the number of edges in the reachability graph. $|C|$ shows the number of candidates determined in Phase 1 of Algorithm PINV.

Firstly, we discuss the result of Algorithm SYM. It is seen that Algorithm SYM attains about 80% reduction over Algorithm EXH in both space and time (e.g., as for the number of nodes in $CW+CF$, $1-17610/102746 = 0.83$). In general, the construction of the SRG needs more overhead than that of FRG. Because for each generation of the next state s, it may check all its symmetric states to determine if it already exists or not. However, this overhead appears to be cancelled since the total number of states to be explored is reduced by the SRG. From these, we can say that Algorithm SYM attains adequate performance.

Secondly, we investigate the result of Algorithm PINV. Since it needs no costly reach-

Table 6.2: Result for performance

Service	Algorithm EXH			Algorithm SYM			Algorithm PINV	
Spec. S	| N |	| T |	Time (s)	| N |	| T |	Time (s)	|C|	Time (s)
CW+CF	102746	446124	4706.4	17610	76732	913.8	423	3.4
CW+DC	9592	38424	270.4	1684	6838	55.6	297	1.8
CW+DT	7120	39036	187.2	1344	7470	43.9	285	1.7
CW+DO	3480	16560	89.3	668	3234	20.7	285	1.7
CW+OCS	23472	126996	705	4032	21912	151.7	282	1.7
CW+TCS	23472	127092	710.6	4032	21930	150.8	282	1.9
CF+DC	65410	243876	1851.5	6662	35902	213.2	99	0.6
CF+DT	38584	206868	1006.1	3087	14079	93.5	105	0.6
CF+DO	17775	80538	447.6	954	4956	17	75	0.5
CF+OCS	130113	704700	3980.2	21863	118545	821.7	102	0.6
CF+TCS	130113	704682	3982.6	21863	118540	838.9	102	0.7
DC+DT	5390	27510	69.6	954	4956	17	48	0.2
DC+DO	4654	23490	57.9	820	4202	14.2	57	0.2
DC+OCS	17325	91212	268.9	2932	15519	61.7	45	0.2
DC+TCS	17325	91296	273.3	2932	15528	63.8	45	0.2
DT+DO	1450	9180	15.7	300	1936	4.9	33	0.1
DT+OCS	7074	54402	102.8	1242	9594	27	33	0.1
DT+TCS	7074	54534	104.8	1242	9616	27.7	33	0.1
DO+OCS	4410	28854	59.7	780	5142	15.3	30	0.1
DO+TCS	4410	28920	60.5	780	5153	15.3	30	0.1
OCS+TCS	22518	175176	367.3	3804	29623	91.5	30	0.1
EMG	522	2766	3.5	116	640	1.3	51	0.1

ability graph generation, it attains extremely effective performance. For example, let us see the case of $CW + CF$. Total time needed for (1)construction of Petri net, (2)calculation of P-invariant, (3)generation of candidates and (4)evaluation of P-invariant equation costs only 3.5 seconds. The performance is about 1340 times that of Algorithm EXH, and about 260 times that of Algorithm SYM. Moreover, space is also reduced by several orders of magnitude. This is justified by the following two reasons. The first one is that the P-invariant calculation and the candidate generation are performed from only structure of rules (i.e., Petri net structure) independently of states. The second reason is that the characterising several states by wildcard extension of one state works efficiently.

6.3.3 Scalability

In order to investigate the applicability of the algorithms to complex services with many users, we evaluate the scalability of the algorithms. We discuss the following two kinds of scalability:

(a)*scalability w.r.t. # of users*: impact of the number of users on the algorithms.

(b)*scalability w.r.t. # of features*: impact of the number of features on the algorithms.

As the metric of scalability, we adopt state space needed for each algorithm, since it is the most dominant factor in determining the cost of the algorithms. For algorithms EXH and SYM, we measure the numbers of nodes in the FRG and SRG, respectively. For Algorithm PINV, we measure the number of undesirable candidate states determined in Phase 1.

First we compare the scalability w.r.t. # of users. Concretely, for a fixed service specification, we observe the growth of state space by varying the number of users. In order to measure as many cases as possible, we select the POTS specification in Figure 2.2. , which is the simplest one. Then, we vary the number of users from 2 to 8.

Figure 6.3.3 shows the result. In the graph, the vertical axis shows the number of states, while the horizontal axis represents the number of users. Note the logarithmic scale on the vertical axis.

Firstly, we discuss the results of Algorithm SYM. It can be seen that, for the increase of the number of users, the state space of Algorithm EXH exponentially grows. And finally, we could not construct FRG with 8 users due to the memory overflow. On the other hand, the state space of Algorithm SYM grows much more slowly than that of Algorithm EXH (almost linear in this case). The larger the number of users is, the more Algorithm SYM attains the significant reduction. From this, we can say that Algorithm SYM is much more scalable than Algorithm EXH with respect to the number of users.

Secondly, we discuss the results of Algorithm PINV. Similarly to the results of Algorithm SYM, the number of states grows almost linearly according to the number of users. It is not surprising that the two curves for Algorithms SYM and PINV are crossed in the small number of users, since the number of candidates determined by Algorithm PINV has nothing to do with the number of reachable states in the SRG. As the number of users increases, Algorithm PINV achieves further reductions than Algorithm SYM. Even for the case of 100 users, Algorithm PINV completes in a very short time (86.44 sec.) and smaller space (800 states). This fact implies that Algorithm PINV has more scalability than Algorithm SYM with respect to the number of users.

Next, we evaluate the scalability w.r.t.# of features. For this, we first prepare the POTS specification and fix the number of users (here we set it to be three). Then, by incrementally adding the specifications DO, DT, DC, EMG to POTS in this order, we observe how the state space grows for each addition of the service.

Figure 6.2 shows the result. Unfortunately, for the increase of the number of features, the state space of Algorithm SYM exponentially grows in line with that of Algorithm EXH. However, it can be seen that Algorithm SYM constantly achieves more than an

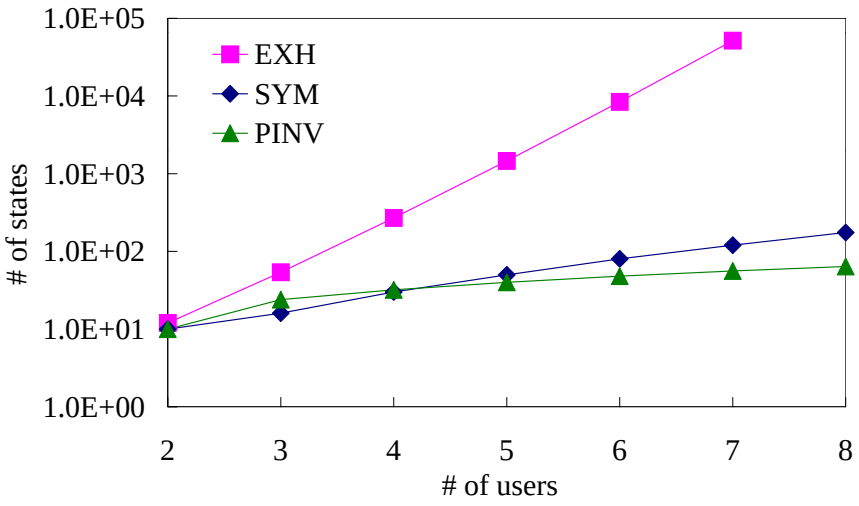


Figure 6.1: Result for scalability w.r.t. # of users

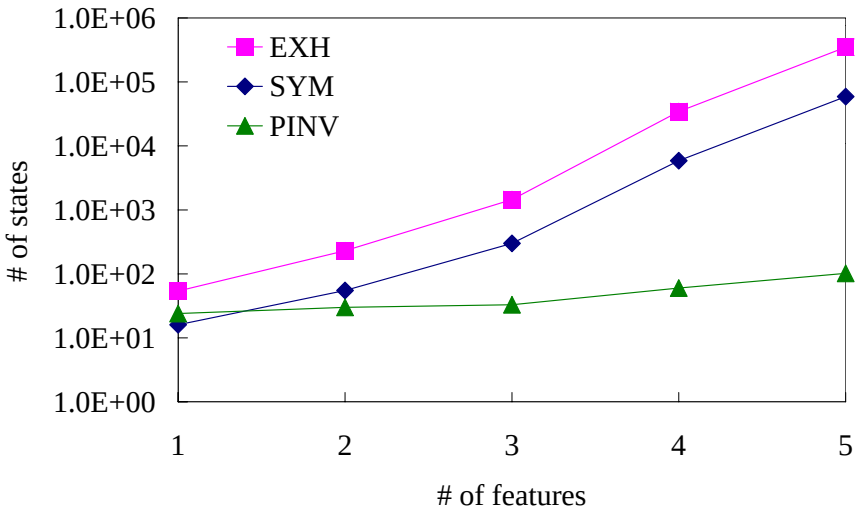


Figure 6.2: Result for scalability w.r.t. # of features

80% reduction over Algorithm EXH (e.g., as for the number of nodes with 5 features, $1 - 58832/348868 = 0.83$). This is justified by the fact that permutation symmetry does not contribute to the different services since it is defined only for the permutation of *users*. As a result, Algorithm SYM is slightly more scalable than Algorithm EXH with respect to the number of features.

As for Algorithm PINV, the state space grows comparably slowly to the scalability with respect to the number of users. So, we can conclude that Algorithm PINV is much more scalable than other two algorithms.

6.4 Discussion

According to the results, we summarise the characteristics of the three algorithms in Table 6.3.

Although the conventional Algorithm EXH has an advantage in its simplicity and optimal detection quality, it is difficult to apply it to practical interaction detection. For example, we recall that we set the number of users in the interaction detection to be only three. Actually, in the case of four users, Algorithm EXH cannot work because of the state explosion. Due to its poor performance and scalability, the application of Algorithm EXH is limited to relatively simple services with a small number of users.

Algorithm SYM is a suitable example which achieves good state reduction by focusing on a constraint of telecommunication services. That is, all users of a service X can use X in the same way. Based on the backbone theory of permutation symmetry, Algorithm SYM succeeds in improving the performance while completely preserving the optimal detection quality of Algorithm EXH. Also, it is very scalable for an increase in the number of users. The only disappointing aspect is that the scalablity with respect to the number of features is poor. However, it is not surprising, since permutation symmetry is defined only on users, not on features. Thus, it does not contribute any state reduction when new

features are added. The other reason is that we assume in the experiment that all users can subscribe to any service dynamically, as in Assumption 6.1. The dynamic subscription and withdrawal may cause combinational explosion of the user's subscription cases. One of the promising ways for improving the scalability is to divide the original problem into smaller subproblems, in each of which the users' subscription cases is *statically fixed*. We are now studying it as future work.

Algorithm PINV takes a completely different approach from other two algorithms in the sense that it performs a static verification. Based on mathematical foundation of the P-invariant method of a Petri net, it extensively utilises the structural information of rule-based service specifications. As a result, Algorithm PINV achieves excellent performance and scalability, which are well applicable to practical settings. For example, even in the case of $CW\&CFV$, which is the most difficult case in the experiment, with 40 users, we confirm that Algorithm PINV definitely works, and its execution is finished within an hour. This is sufficient enough even for industrial settings.

However, we cannot directly compare Algorithm PINV with the other two algorithms, because Algorithm PINV does not solve the exact interaction detection problem, strictly speaking. In compensation for its excellent performance and scalability, there are mainly two drawbacks. Firstly, the detectable interaction classes are limited. This is because we do not always have nice techniques for generating undesirable candidate states, automatically. Although there is a deadlock detection method using the P-invariant method in [3], the proof of deadlock free is performed by manually for each given system or specification, so it is difficult to implement. In our work, we specifically select two classes of non-deterministic states and violating states, and propose the automated candidate generation procedures for them. We are now studying the generation procedures for other types of undesirable states.

The second drawback is that the detection quality achieved by Algorithm PINV should

Table 6.3: Characterictics of Algorithms

Metrics	Algorithm EXH	Algorithm SYM	Algorithm PINV
Verification Type	dynamic	dynamic	static
Backbone Theory	exhaustive search	permutation symmetry	Petri net P-invariant
Detectable FI Classes	all	all	partial
Quality	optimal	optimal	semi-optimal
Performance	very poor	good	excellent
Scalability w.r.t.			
(# of users)	very poor	very good	excellent
(# of features)	very poor	poor	excellent

be semi-optimal, but not optimal theoretically speaking. This is clearly explained because the P-invariant method guarantees the reachability of states by only a necessary condition. However, the experimental evaluation shows that the detection quality is comparative to optimal for the practical interaction detection. The reason why the necessary condition essentially worked as both a necessary and sufficient condition is now being investigated. This is one of the most interesting and important issues for future research.

After all, although it can be from Table 6.3 seen that the proposed algorithms SYM and PINV are involved in a kind of trade-off relation, we believe that both of the proposed algorithms successfully extend the applicability of formal verification methods to interaction detection. As a result, we conclude that both algorithms are applicable to the interaction detections of more practical services with many users.

Chapter 7

Conclusion

7.1 Achievements

In this dissertation, we addressed research on algorithms for efficient feature interaction detection. Also, we have conducted experimental evaluation through application to practical services. We formulated interaction detection problems for the four classes of interactions: deadlock, loop, non-determinism and violation of invariants.

We first proposed Algorithm SYM. By focusing on a constraint of telecommunication services that "All users of service X can utilise X in the same way", we define a relation called permutation symmetry of users. Based on this relation, we have defined a reachability graph, called symmetric reachability graph (SRG). By means of the SRG, we have successfully reduced the state space needed for the conventional exhaustive search (Algorithm EXH), while completely preserving necessary information for all four classes of interactions. As a result, Algorithm SYM performs the optimal interaction detection based on the necessary and sufficient conditions with a smaller space and time.

Next, we proposed Algorithm PINV which extensively utilises the P-invariant method of Petri nets for interaction detection. Algorithm PINV is a static verification method in the sense that it does not employ the reachability analysis by means of graphs, which lim-

its the detectable interaction classes to only non-determinism and violation of invariants. However, because Algorithm PINV works in a static manner without expensive reachability analysis, significant cost reduction for the interaction detection can be achieved.

The experimental evaluation of the proposed algorithms demonstrated their respective effectiveness. We have prepared service specifications for practical services, and evaluated the proposed two algorithms by the following metrics: detection quality, performance and scalability.

From the results, we have confirmed that Algorithm SYM achieves the optimal detection quality as shown in backbone theories. Also, it was shown that Algorithm SYM attains relatively good performance and significantly good scalability with respect to the number of users. As for Algorithm PINV, it was interesting that it has attained semi-optimal detection quality in the sense that all interactions suspected by Algorithm PINV are correct ones. The performance and scalability of Algorithm PINV were excellent for the practical settings.

From these, we conclude that both of the proposed algorithms successfully extend the applicability of formal verification methods to feature interaction detection.

7.2 Future Research

Several important issues are left open for future study.

The first issue concerns the extension and improvement of the detection algorithms. Concerning the dynamic algorithm such as the proposed Algorithm SYM, we consider there to be room for improvement. As mentioned before, a potentially promising direction for scalability improvement with respect to the number of features is to divide the original problem into smaller subproblems based on the user's subscription conditions. Also, there are powerful verification methods for dynamic verification, such as *stubborn sets*[59, 60], *partial order*[30, 31], and *symbolic model checking*[12, 47], in other research fields.

Introducing these methods to the interaction detection will produce further improvement of dynamic analysis.

Also, as for the Petri net based algorithm PINV, we have to continue further study. The most important thing is to extend the feasible interaction classes. To do this, we have to develop good candidate generation procedures for other classes. Also, in the Petri net field, there are still other excellent structural methods such as *T-invariants*, *siphons and traps* [48, 49], which are expected to be applicable to static verification. Examination of other structural methods is an interesting issue.

A secondary issue is with respect to the numbers of users and features such like:

(a) "How many users should be taken into account in order to guarantee an interaction-free pair of services?"

(b) "How many combinations should be investigated for given n services."

There are two ways to tackle the above (a). The problem is that if we confirm that the case with n users is interaction-free, then can we also assert the case with $n+1$ users to be the interaction-free? From a theoretical viewpoint, we have to employ the proof with induction [47]. The induction is not fully automatic — some human input is required in general. From a practical viewpoint, we should set some upper bound with respect to the number of users. To determine the upper bound, extensive knowledge by subjective experts is necessary. The problem (b) can be viewed as follows. Suppose that three services $\mathcal{S}_A$, $\mathcal{S}_B$ and $\mathcal{S}_C$ are given, and that we have confirmed the combinations $\mathcal{S}_A \oplus \mathcal{S}_B$, $\mathcal{S}_B \oplus \mathcal{S}_C$ and $\mathcal{S}_C \oplus \mathcal{S}_A$ to be interaction-free. Then, should we perform the interaction detection for $\mathcal{S}_A \oplus \mathcal{S}_B \oplus \mathcal{S}_C$? This kind of proof is a challenging issue for the significant cost reduction of the interaction detection process. The fundamental framework for this kind of problem is recently proposed by LaPorta et al. [41].

In this dissertation, we addressed only four classes of interactions, that is, deadlock, loop, non-determinism, violation of invariants. However, there still exist some other

classes of interactions. For example, Ohta et al. [52] presented other classes of interactions called semantics interactions, such as appearance of abnormal states, and disappearance of normal transitions. Also, there are some other interactions which we may be unable to define explicitly within the state transition model, such as those caused by billing, terminology [51, 61]. Examination of these other classes is an important issue as a future work.

As is discussed to date in the previous Feature Interaction Workshops [8, 18, 22, 40], the feature interaction problem has been most thoroughly documented in public telecommunication systems. However, it occurs in other systems as well. Magill and Tsang et al. formulate the feature interaction problem in multimedia systems [45, 57]. Interesting examples include such interaction between an Email service and vacation programs, and interaction between video-on-demand and video-conference. The application of the proposed algorithms to multimedia services is also a challenging issue.

Bibliography

[1] Aho, A.V., Hopcroft, J.E. and Ullman, J.D., "The Design and Analysis of Computer Algorithms," Addison-Wesley, 1974.

[2] Aggoun, I. and Combes, P., "Observers in the SCE and the SSE to detect and resolve service interactions," *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.1-23, IOS Press 1997.

[3] Alla, H., Ladet, P., Martinez and J., Silva-Suarez, M., "Modelling and validation of complex systems by coloured Petri-nets: Application to a flexible manufacturing system," *Lecture Notes in Computer Science,* Vol 188, Springer-Verlag, pp.215-233, 1985.

[4] Au, P.K. and Atlee, J.M., "Evaluation of a state-based model feature interactions," *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.153-167, July, 1997.

[5] Blom J., Bol, R. and Kempe, L., "Automatic detection of feature interactions in temporal logic", *Proc. of Third Workshop on Feature Interactions in Telecommunications Systems*, pp.1-19, IOS Press 1995.

[6] Blom J., Bol, R. and Kempe, L., "Using temporal logic for modular specification of telephone services," *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.197-216, IOS Press 1994.

[7] Blom J., "Formalization of requirements with emphasis on feature interaction detection", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.61-77, July, 1997.

[8] Bouma, W. and Velthuijsen, H. eds. "Feature Interactions in Telecommunications Systems II", IOS Press, 1992.

[9] Bouma, W., Levelt, W., Melisse, A., Middelburg, K., amd Verhaard, L., "Formalization of properties for feature interaction detection: Experience in a real-life situation," *Lecture Notes in Computer Science*, Vol.851, Springer-Verlag, pp. 393-405, 1994.

[10] Boumerzbeur, R. and Logrippo, L., "Specifying telephone systems in LOTOS", *IEEE Communication Magazine*, Vol.31,No.8, pp.38-45, 1993.

[11] Bowen, T.F., et al., "The feature interaction problem in telecomunication systems," *Proc. 7th Int'l. Conf. on Software Eng. for Telecommun. Switching Sys.*, pp.59-62, July 1989.

[12] Cabodi, G. and Camurati. "Symbolic FSM Traversals based on the transition relation", *IEEE Trans. on Computer-Aided Design of Integrated Circuit and Systems,* Vol.16, No.5, May 1997.

[13] Cameron, E.J. and Velthuijsen, H., "Feature interactions in telecommunications systems," *IEEE Communication Magazine,* Vol.31, No.8, pp.18-23, 1993.

[14] Cameron, E.J., Griffeth, N.D., Lin, Y-J., Nilson, M.E., Schnure W.K. and Velthuijsen, H., "A feature interaction benchmark for IN and Beyond," *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.1-23, IOS Press 1994.

[15] Cameron, E.J., Cheng K., Lin, F-J, Liu, H. and Pinheiro B, "A formal AIN service creation, feature interactions analysis and management environment: an industrial appliction", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.342-346, July, 1997.

[16] Capellmann, C., Combes, P., Pettersson., J., Renard, B. and Ruiz, J.L., "Consistent interaction detection - A comprehensive approach integrated with service creation", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.183-197, July, 1997.

[17] Capellmann, C., Demandt, R., Galvez-Estrada, R., Nithsche, U., Ochsenschlager," "Case study: Service interaction detection by formal verification under behavior abstraction", *Proc. of Computer Aided Verification*, pp.466-469, 1996.

[18] Cheng, K.E., and Ohta, T. eds. "Feature Interaction in Telecommunications III", IOS Press, 1993.

[19] Cortadella, J., "Combining structual and symbolic methods for the verification of concurrent systems", *Proc. of Int'l conf. on Application of Concurrency to System Design*, pp.2-7, 1998.

[20] Desel, J., Neuendorf, K. -P. and Radola, M. -D, "Proving nonreachability by modulo-invariants", *Theoretical Computer Sciences*, Vol. 153, pp.49-64, 1996.

[21] Dssouli, R., Some, S., Guillery, J.W., and Rico, N., "Detection of feature interactions with REST", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.271-283, July, 1997.

[22] Dini, P., Bautaba, R. and Logrippo, L. eds. "Feature Interaction in Telecommunication Networks IV", IOS Press, 1997.

[23] Fristsche, N. "Runtime resolution of feature interactions in architectures with separated call and feature control", *Proc. of Third Workshop on Feature Interactions in Telecommunications Systems*, pp.43-64, 1995.

[24] Gammelgaard, A. and Kristensen E.J., "Interaction detection, a logical approach," Proc. of Second Workshop on Feature Interactions in Telecommunications Systems, pp.178-196, 1994.

[25] Griffeth, N.D. and Lin, Y-J, "Extending telecommunications systems: the feature-interaction problem", *IEEE Computer*, Vol.26, No.8, pp.14-18, 1993.

[26] Hall, R. J, "Feature combination and interaction detection via foreground/background models", *Proc. of Fifth Workshop on Feature Interactions and Software Systems*, pp.232-246, IOS Press 1998.

[27] Harada, Y., Hirakawa, Y., Takenaka, T. and Terashima, N., "A conflict detection support method for telecommunication service descriptions", *IEICE Trans. Commun.*, Vol. E75-B, No.10, Oct., 1992.

[28] Heisel, M. and Souquires, J., "A heuristic approach to detect feature interactions in requirements", *Proc. of Fifth Workshop on Feature Interactions and Software Systems*, pp.165-171, IOS Press 1998.

[29] Hirakawa, Y. and Takenaka, T., "Telecommunication service description using state transition rules," *Proc. of IEEE Int'l Workshop on Software Specification and Design,* pp.140-147, Oct. 1991.

[30] Holzmann, G.J., "The model cheker SPIN", *IEEE Trans. on Software Engineering*, Vol.23, No.5, pp.279-295, May 1997.

[31] Holzmann, G.J., Godefroid P. and Pirottin, D., "Coverage preserving reduction strategies for reachability analysis", *Proc. of IFIP Protocol Specification, Testing and Verification , XII*, pp.349-363, 1992.

[32] Jensen, K., "Coloured Petri Nets," *EATCS Monographs on Theoretical Computer Science*, Vol1-2, Springer Verlag, 1992.

[33] Jensen, K. and Rozenberg, G., "High-level Petri Nets", Springer-Verlag, 1991.

[34] Kawarazaki, Y. and Ohta, T., "A new proposal for feature interaction detection and elimination," *Proc. of Third Workshop on Feature Interactions in Telecommunications Systems*, pp.127-139, IOS Press 1995.

[35] Keck, D.O. and Kuehn, P.J., "The feature interaction problem in telecommunications systems: A survey," *IEEE Trans. on Software Engeering*, Vol.24, No.10, pp.779-796, 1998.

[36] Keck, D.O., "A tool for the identification of interaction-prone call scenarios", *Proc. of Fifth Workshop on Feature Interactions and Software Systems*, pp.276-290, IOS Press 1998.

[37] Khoumsi, A., "Detection and resolution of interactions between services of telephone networks", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.78-92, July, 1997.

[38] Kimbler, K., "Addressing the interaction problem at the enterprise level", *Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems*, pp.13-22, July, 1997.

[39] Kimbler, K., Capellmann, and Velthuijsen, H., "Comprehensive approach to service interaction handling", *Computer Networks and ISDN Systems*, Sept. 1998.

[40] Kimbler, K. and Bouma, L.G. eds. "Feature Interaction in Telecommunications and Software System V", IOS Press, 1993.

[41] LaPorta, T.F., Lee, D. and Lin, Y-J., "Protocol feature interactions", *Proc. of IFIP FORTE XI/ PSTV XVIII*, pp. 59-74, Nov. 1998.

[42] Lee, A., "Formal specification — a key to service interaction analysis", *Proc. of SETSS*, pp.62-66, 1992.

[43] Lin, F. J. and Lin,Y-J., "A building block approach to detecting and resolving feature interactions", *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.86-191, 1994.

[44] Lin, F. J., Liu, H. and Ghosh, A., "A methodology for feature interaction detection in the AIN 0.1 framework", *IEEE Trans. on Software Engeering*, Vol.24, No.10, pp.797-817, 1998.

[45] Magill, E.H., Tsang S. and Kelly, B., "The feature interaction problem in networked multimedia services: past, present and feature", *http://www.comms.eee.strath.ac.uk/ fi/fimna.html*, Oct. 1996.

[46] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S. and Feanceshinis G., "Modelling with Generalized Stochastic Petri Nets", John Wiley & Sons, 1995,

[47] McMillan, K.L., "Symbolic Model Checking", Kluwer Academic Publishers, 1993.

[48] Murata, T. and Zhang, D., "A predicate-transition net model for parallel interpretation of logic programs," *IEEE Trans. on Software Engineering*, Vol.14, No.4, pp.481-497, Apr. 1988.

[49] Murata, T., "Petri nets: properties, analysis and applications," *Proc. of IEEE*, Vol.77, No.4, pp.541-580, Apr.1988.

[50] Nitsche, U., "Application of formal verification and behavior abstraction to the service interaction problem in intelligent networks", *J. Systems Software*, North-Holland, Vol.40, pp.227-248, 1998.

[51] Ohta, T. and Harada Y., "Classification, detection and resolution of service interactions in telecommunication services," *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.60-72, 1994.

[52] Ohta, T. and Krischen, F., "Formal definitions of feature interactions in telecommunication software", *IEICE Trans.*, Vol.E81-A, No.4, pp.635-638.

[53] Plath, M. and Ryan, M., "Plug-and-play features", *Proc. of Fifth Workshop on Feature Interactions and Software Systems*, pp.150-164, IOS Press 1998.

[54] Stepien, B., and Logrippo, L., "Representing and verifying interactions in telecommunication features using abstract data types", *Proc. of Third Workshop on Feature Interactions in Telecommunications Systems*, pp.141-155, IOS Press 1995.

[55] Thomas, M., "Modelling and analyzing users vires of telecommunication services," *Proc. of Second Workshop on Feature Interactions in Telecommunications Systems*, pp.168-182, IOS Press 1994.

[56] Tomioka, M., Tanaka, Y. and Mizuno, O., "Service interaction prediction method for customized specification", *Proc. of second Asia-Paciffic Conference on Communications*, pp.123-127, June 1995.

[57] Tsang, S., Magill, E.H. and Kelly, B., "An investigation of the feature interaction problem in networked multimedia services", *Proc. of Third Communication Networks Symposium*, pp.58-61, 1996.

[58] Ueda, Y., Takura, A. and Ohta, T., "A fast verification method of the reachability set on discrete event systems" (in Japanese), *Technical Report of IEICE*, CST95-10, pp.29-34, 1995.

[59] Valmari A., "Stubborn sets for reduced state space generation", *Lecture Notes in Computer Science*, Vol.483, Springer-Verlag, pp. 491-515, 1991.

[60] Valmari A., "The state explosion problem", *Lecture Notes in Computer Science*, Vol.1491, Springer-Verlag, 1998 (to appear).

[61] Wakahara, Y., Fujioka, M., Kikuta, H., Yagi, H. and Sakai, S., "A method for detecting service interactions," *IEEE Communication Magazine,* Vol.31, No.8, pp.32-37, 1993.

[62] Yoneda, T. and Ohta, T., "A formal approach for definition and detection of feature interactions", *Proc. of Fifth Workshop on Feature Interactions and Software Systems*, pp.165-171, IOS Press 1998.

[63] Zave, P., "Feature interactions and formal specifications in telecommunications," *IEEE Computer*, Vol.26, No.8, pp.20-30, 1993.

[64] ITU-T Recommendations Q.1200 Series., "Intelligent Network Capability Set 1 (CS1)", Sept. 1990.

[65] ITU-T Recommendations Z.100, "Specification and Description Language (SDL)", 1993.

[66] Bellcore, "LSSGR Features Common to Residence and Business Customers I, II, III," Issue 2, July 1987.

[67] Bellcore, "Advanced Intelligent Network (AIN) Release 1, Switching Systems Generic Requirements," Bellcore Technical Advisory TA-NWT-001123, May 1991.

Appendix A

Calculating P-invariants

It is not an easy task to obtain a P-invariant of general coloured Petri nets. However, regarding our service specification net, there exists a relatively simple method for easy calculation of the P-invariant.

The service specification net possesses a specific structure, that is, each transition in the net always move the *same number* of tokens along a given arc, independently of the firing substitution. This kind of Petri net is called a *uniform coloured Petri net* [32].

A uniform coloured net has an underlying (colourless) Petri net structure [58, 48]. Since the P-invariant of colourless Petri nets is easily calculated by integer matrix operations (e.g., [20]), we extensively utilise the underlying colourless net for P-invariant computation of the service specification net. There are many articles and text books of the colourless Petri nets (e.g., [48, 46]), hence, the details are omitted here.

Intuitively, the underlying colourless net $UPT(\mathcal{S})$ of $\mathcal{N}(\mathcal{S})$ is obtained by removing all arc labels $\langle x_1, ..., x_k \rangle$, and by counting the number of tokens in each place p (regardless of tokens' colour) [58]. Here, we use the following notations.

- For any weighted set(or multi set) w, $|w|$ denotes the cardinal number of w over weighted sets (or multi sets, respectively). For example, for $w = \{2\langle A \rangle, 3\langle B \rangle\}$, $|w| = 5$.

- For any linear function f, $|f|$ denotes the multiplicity of f. For example, $|o| = 0$, $|id| = |p_1| = |p_2| = |rv| = 1$, $|5 \cdot p_1 + 3 \cdot p_2| = 8$ and so on.

The incident matrix of $UPN(\mathcal{S})$ is obtained from $A[p,t]$ of $\mathcal{N}(\mathcal{S})$ by replacing each entry $A[p,t]$ by $|A[p,t]|$, thus we let it be $|A|$. Then, a P-invariant of $UPN(\mathcal{S})$ is an integral vector X such that $X * |A| = 0$ [48]. At this time, the following theorem holds.

Theorem A.1 [32]

$$Y \textit{ is a P-invariant of } \mathcal{N}(\mathcal{S}) \quad \Rightarrow \quad |Y| \textit{ is a P-invariant of } UPN(\mathcal{S}).$$

Theorem A.1 allows us to narrow the scope for finding unknown Y's, that is, we can discard any vector Y such that $|Y| \neq X$ from candidates of the P-invariant of $\mathcal{N}(\mathcal{S})$.

After all, in order to obtain the P-invariant of $\mathcal{N}(\mathcal{S})$, we perform the following steps.

Step1: Calculate the P-invariant X of $UPT(\mathcal{S})$.

Step2: Construct candidate vectors Y's such that $|Y| = X$

Step3: Check if each of candidates is the P-invariant or not by evaluating $Y * A = \mathbf{0}$.

Step 1 is carried out by a method like Gaussian Elimination [20] in polynomial time, and Step 3 is a trivial task. To implement Step 2, we use a heuristical method. Suppose that a P-invariant X of $UPT(\mathcal{S})$ is given as follows. $X = \begin{matrix} p_1 & p_2 & \cdots & p_u \\ (x_1 & x_2 & \cdots & x_u) \end{matrix}$, where $\begin{matrix} p_1 & p_2 & \cdots & p_u \end{matrix}$

$p_i \in P$ and x_i is a non-negative integer. Also, suppose that $Y = \begin{matrix} (y_1 & y_2 & \cdots & y_u) \end{matrix}$ is the target P-invariant of $\mathcal{N}(\mathcal{S})$, where y_i is some linear function. Let $ari(p_i)$ be the arity of place p_i. Then, for requirement of $|Y| = X$, we use the following rule for linear function assignments.

$$y_i := \begin{cases} x_i \cdot id & \cdots (\text{if } \ ari(p_i) = 1) \\ e_1 \cdot proj_1 + e_2 \cdot proj_2 + \ldots + e_k \cdot proj_k \quad (e_1 + \ldots + e_k = x_i) & \cdots (\text{if } \ ari(p_i) = k \geq 2) \end{cases}$$

where e_i is a non-negative integer, id is the identity function, and $proj_i$ is a projection function such that $proj_i\langle a_1, a_2, ..., a_k\rangle = \langle a_i\rangle$. By the above rule, clearly we have $|Y| = X$. Although there may exist other rules to determine vectors Y's such that $|Y| = X$, we have confirmed heuristically that the above rule works well for obtaining *good* P-invariants of $\mathcal{N}(\mathcal{S})$.

To illustrate the technique, let us compute a P-invariant of $\mathcal{N}(\mathcal{S})$ in Figure 5.3. From the incident matrix A of $\mathcal{N}(\mathcal{S})$ in Example 5.17, we have the following $|A|$, which is the incident matrix of $UPN(\mathcal{S})$.

$$|A| = \begin{array}{r} \\ idle \\ dialtone \\ calling \\ busytone \\ talk \end{array} \begin{array}{c} \begin{array}{cccccccc} ps_1 & ps_2 & ps_3 & ps_4 & ps_5 & ps_6 & ps_7 & ps_8 \end{array} \\ \left[\begin{array}{cccccccc} -1 & 1 & 0 & -1 & 2 & 0 & 1 & 1 \\ 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 \end{array}\right] \end{array}$$

where $ps_i = pots_i$. Then, by using a computation method in [20], we have the following P-invariant X of $UPT(\mathcal{S})$.

$$X = \begin{array}{c} \begin{array}{ccccc} idl & dlt & clg & bst & tlk \end{array} \\ (1 \quad 1 \quad 2 \quad 1 \quad 1) \end{array}$$

According to the rule of linear function assignments, we have the following P-invariant candidates. Note that arities of *idle*, *dialtone*, *calling*, *busytone* and *talk* are 1, 1, 2, 1 and 2, respectively.

$$\begin{aligned} Y_1 &= (id \quad id \quad 2 \cdot p_1 \quad id \quad p_1) \\ Y_2 &= (id \quad id \quad 2 \cdot p_1 \quad id \quad p_2) \\ Y_3 &= (id \quad id \quad p_1 + p_2 \quad id \quad p_1) \\ Y_4 &= (id \quad id \quad p_1 + p_2 \quad id \quad p_2) \end{aligned}$$

$$
\begin{aligned}
Y_5 &= (id \quad id \quad 2 \cdot p_2 \quad id \quad p_1) \\
Y_6 &= (id \quad id \quad 2 \cdot p_2 \quad id \quad p_2)
\end{aligned}
$$

where p_j represents a projection function $proj_j$. Clearly, we can see that $|Y_i| = X$. Finally, by evaluating $Y_i * A = \mathbf{0}$ for each Y_i, we confirm that Y_3 and Y_4 are the P-invariants of $\mathcal{N}(\mathcal{S})$.

Appendix B

Service Specifications

We present service specifications prepared for the experiment in Chapter 6, which are expressed in the SVAL language.

B.1 Plain Ordinary Telephone Service (POTS)

```
Specification POTS;
User:      A, B, C;
Var:       x, y;
Predicate: idle(x), dialtone(x), calling(x,y), path(x,y), busytone(x);
Event:     onhook(x), offhook(x), dial(x,y);
Init:      idle(A), idle(B), idle(C);
Rule:
   pots1:  idle(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y) [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
```

```
   pots6:  calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x).
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
```

B.2 Call Waiting (CW)

```
Specification CW;
User:       A, B, C;
Var:        x, y, z;
Predicate: idle(x),dialtone(x),calling(x,y),busytone(x),path(x,y),
           m-cw(x),cw-calling(x,y), path-passive(x,y), cw-regconfirm(x),
           RS-cw(x),passive-state(x), CW(x), cw-mode(x);
Event:      offhook(x), onhook(x), dial(x,y), flash(x), reg-cw(x), wdraw-cw(x) ;
Init:       idle(A), idle(B), idle(C), RS-cw(A), RS-cw(B), RS-cw(C);
Rule:
   pots1:  idle(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y)  [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) & ~CW(y)  [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:  calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) & ~cw-mode(x) & ~cw-mode(y)
                          [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x)
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
     cw1:  CW(x) & path(x,y)& dialtone(z) & m-cw(x) & ~cw-mode(x) & ~cw-mode(y)
```

```
            [dial(z,x)] CW(x) & cw-calling(z,x) & path(x,y) & cw-mode(x).

 cw2:  CW(x) & cw-calling(z,x)& path(x,y)& path(y,x) & cw-mode(x)

            [onhook(x)] CW(x) & calling(z,x)& busytone(y) & m-cw(x).

 cw3:  CW(x) & cw-calling(z,x)& path(x,y)& path(y,x) & cw-mode(x)

            [onhook(y)] CW(x) & calling(z,x) & idle(y) & m-cw(x).

 cw4:  CW(x) & cw-calling(z,x) & cw-mode(x)

            [onhook(z)] CW(x) & idle(z) & m-cw(x).

 cw5:  CW(x) & cw-calling(z,x) & path(x,y) & path(y,x)

            [flash(x)] CW(x) & path(x,z) & path(z,x) & path-passive(x,y).

 cw6:  CW(x) & path(x,z) & path(z,x) & path-passive(x,y)

            [flash(x)] CW(x) & path(x,y) & path(y,x) & path-passive(x,z).

 cw7:  CW(x) & path-passive(x,y) & passive-state(x) & cw-mode(x)

            [flash(x)] CW(x) & path(x,y) & path(y,x) & m-cw(x).

 cw8:  CW(x) & path(x,z) & path-passive(x,y) & cw-mode(x)

            [onhook(y)] CW(x) & path(x,z) & idle(y)& m-cw(x).

 cw9:  CW(x) & path(x,z) & path(z,x) & path-passive(x,y) & cw-mode(x)

            [onhook(x)] CW(x) & calling(y,x) & busytone(z) & m-cw(x).

cw10:  CW(x) & path(z,x) & path(x,z) & path-passive(x,y) & cw-mode(x)

            [onhook(z)] CW(x) & passive-state(x) & idle(z)

           & path-passive(x,y) & cw-mode(x).

cw11:  CW(x) & path-passive(x,y) & passive-state(x) & cw-mode(x)

            [onhook(x)] CW(x) & calling(y,x) & m-cw(x).

cw12:  dialtone(x) & RS-cw(x)

            [reg-cw(x)] cw-regconfirm(x) & m-cw(x) & CW(x).

cw13:  dialtone(x) &  m-cw(x) & CW(x)

            [wdraw-cw(x)] cw-regconfirm(x) & RS-cw(x).
```

```
    cw14:  cw-regconfirm(x) [offhook(x)]   idle(x).
```

B.3 Call Forwarding (CF)

```
Specification CFV;

User:      A, B, C;

Var:       x, y, z;

Predicate: idle(x), dialtone(x), calling(x,y), busytone(x), path(x,y),
           m-cfv(x), confirmation(x),
           pingring(y,x), cfv-regconfirm(x), m-regcfv(x,y),
           RS-cfv(x), CFV(x);

Event:     dial(x,y), offhook(x), onhook(x), ccfv-code(x),
           cfv-code(x), reg-cfv(x), wdraw-cfv(x),
           timeover_m-Rcfv(x,y), timeover_pingring(x,y),
           timeover_m-Rcfv2(x,y);

Init:      idle(A), idle(B), idle(C),
           RS-cfv(A), RS-cfv(B), RS-cfv(C);

Rule:

   pots1:  idle(x) [offhook(x)] dialtone(x).

   pots2:  dialtone(x) & ~CFV(x) [onhook(x)] idle(x).

   pots3:  dialtone(x) & idle(y) & ~CFV(x) & ~CFV(y) [dial(x,y)] calling(x,y).

   pots4:  dialtone(x) & ~idle(y) & ~CFV(x) & ~CFV(y) [dial(x,y)] busytone(x).

   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).

   pots6:  calling(x,y) & ~CFV(x) [offhook(y)] path(x,y) & path(y,x).

   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).

   pots8:  busytone(x) [onhook(x)] idle(x).

   pots9:  dialtone(x) [dial(x,x)] busytone(x).
```

```
cfv1:  CFV(x) & dialtone(x) & idle(y) & m-cfv(x) & ~m-regcfv(y,*)
            [dial(x,y)] CFV(x) & calling(x,y) & m-cfv(x).
cfv2:  CFV(x) & calling(x,y)& m-cfv(x)
            [offhook(y)] CFV(x) & path(x,y) & path(y,x) & m-regcfv(x,y).
cfv3:  CFV(x) & dialtone(x) & m-cfv(x) & m-regcfv(y,z)
            [dial(x,y)] CFV(x) & busytone(x) & m-cfv(x) & m-regcfv(y,z).
cfv4:  CFV(x) & dialtone(x) & m-cfv(x) & ~idle(y) & ~m-regcfv(y,*)
            [dial(x,y)] CFV(x) & busytone(x) & m-cfv(x).
cfv5:  CFV(x) & dialtone(x) & m-regcfv(x,y)
            [ccfv-code(x)] CFV(x) & confirmation(x) & m-cfv(x).
cfv6:  CFV(x) & confirmation(x) [onhook(x)] CFV(x) & idle(x).
cfv7:  CFV(y) & dialtone(x) & idle(y) & idle(z) & m-regcfv(y,z) & ~m-cfv(x)
            [dial(x,y)] CFV(y) & calling(x,z) & pingring(y,x) & m-regcfv(y,z).
cfv8:  CFV(y) & dialtone(x) & ~idle(y) & idle(z) & m-regcfv(y,z)& ~m-cfv(x)
            [dial(x,y)] CFV(y) & calling(x,z) & m-regcfv(y,z).
cfv9:  CFV(x) & pingring(x,y) [timeover_pingring(x,y)] CFV(x) & idle(x).
cfv10: CFV(y) & dialtone(x) & m-regcfv(y,z) & ~idle(z) & ~m-cfv(x)
            [dial(x,y)] CFV(y) & busytone(x) & m-regcfv(y,z).
cfv11: dialtone(x) & RS-cfv(x)
            [reg-cfv(x)] CFV(x) & m-cfv(x) & cfv-regconfirm(x).
cfv12: CFV(x) & dialtone(x) & m-cfv(x)
            [wdraw-cfv(x)] cfv-regconfirm(x) & RS-cfv(x).
cfv13: cfv-regconfirm(x) [onhook(x)] idle(x).
```

B.4 Originating Call Screening (OCS)

```
Specification OCS;
```

```
User:       A, B, C;
Var:        x, y;
Predicate:  idle(x), dialtone(x), calling(x,y), path(x,y),
            busytone(x), OCS(x,y), RS-OCS(x);
Event:      onhook(x), offhook(x), dial(x,y), reg-ocs(x,y), wdraw-ocs(x);
Init:       idle(A), idle(B), idle(C),
            RS-OCS(A), RS-OCS(B), RS-OCS(C);
Rule:
   pots1:   idle(x) [offhook(x)] dialtone(x).
   pots2:   dialtone(x) [onhook(x)] idle(x).
   pots3:   dialtone(x) & idle(y) & ~OCS(x,y) [dial(x,y)] calling(x,y).
   pots4:   dialtone(x) & ~idle(y) & ~OCS(x,y) [dial(x,y)] busytone(x).
   pots5:   calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:   calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:   path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:   busytone(x) [onhook(x)] idle(x).
   pots9:   dialtone(x) [dial(x,x)] busytone(x).
    ocs1:   idle(x) & RS-OCS(x) [reg-ocs(x,y)] idle(x) & OCS(x,y).
    ocs2:   idle(x) & OCS(x,y)  [wdraw-ocs(x)] idle(x) & RS-OCS(x).
    ocs3:   dialtone(x) & OCS(x,y) [dial(x,y)] busytone(x) & OCS(x,y).
```

B.5 Terminating Call Screening (TCS)

```
Specification TCS;
User:       A, B, C;
Var:        x, y;
Predicate:  idle(x), dialtone(x), calling(x,y), path(x,y),
```

```
           busytone(x), TCS(x,y), RS-TCS(x);
Event:     onhook(x), offhook(x), dial(x,y), regist-tcs(x,y), wdraw-tcs(x);
Init:      idle(A), idle(B), idle(C),
           RS-TCS(A), RS-TCS(B), RS-TCS(C);
Rule:
   pots1:  idle(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y) & ~TCS(y,x) [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) & ~TCS(y,x) [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:  calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x).
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
    tcs1:   idle(x) & RS-TCS(x) [reg-tcs(x,y)] idle(x) & TCS(x,y).
    tcs2:   idle(x) & TCS(x,y)  [wdraw-tcs(x)] idle(x) & RS-TCS(x).
    tcs3:   dialtone(y) & TCS(x,y) [dial(y,x)] busytone(y) & TCS(x,y).
```

B.6 Denied Orgination (DO)

```
Specification DO;
User:      A, B, C;
Var:       x, y;
Predicate: idle(x), dialtone(x), calling(x,y), path(x,y),
           busytone(x), DO(x), RS-do(x);
Event:     onhook(x), offhook(x), dial(x,y), reg-do(x), wdraw-do(x);
Init:      idle(A), idle(B), idle(C),
```

```
           RS-do(A), RS-do(B), RS-do(C);
Rule:
   pots1:  idle(x) & ~DO(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y) [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:  calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x).
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
     do1:  idle(x) & RS-do(x) [reg-do(x)] idle(x) & DO(x).
     do2:  idle(x) & DO(x) [wdraw-do(x)] idle(x) & RS-do(x).
     do3:  idle(x) & DO(x) [offhook(x)] busytone(x) & DO(x).
```

B.7 Denied Termination (DT)

```
Specification DT;
User:      A, B, C;
Var:       x, y;
Predicate: idle(x), dialtone(x), calling(x,y), path(x,y),
           busytone(x), DT(x), RS-dt(x);

Event:     onhook(x), offhook(x), dial(x,y), reg-dt(x), wdraw-dt(x);
Init:      idle(A), idle(B), idle(C),
           RS-dt(A), RS-dt(B), RS-dt(C);
Rule:
```

```
   pots1:  idle(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y) & ~DT(y) [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) & ~DT(y) [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:  calling(x,y) & ~CFV(x) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x).
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
     dt1:  idle(x) & RS-dt(x) [reg-dt(x)] idle(x) & DT(x).
     dt2:  idle(x) & DT(x) [wdraw-dt(x)] idle(x) & RS-dt(x).
     dt3:  dialtone(x) & DT(y) [dial(x,y)] busytone(x) & DT(y).
```

B.8 Direct Connect (DC)

```
Specification DC;
User:       A, B, C;
Var:        x, y, z;
Predicate:  idle(x), dialtone(x), calling(x,y), path(x,y),
            busytone(x), DC(x,y), RS-DC(x);
Event:      onhook(x), offhook(x), dial(x,y), reg-dc(x,y), wdraw-dc(x);
Init:       idle(A), idle(B), idle(C),
            RS-DC(A), RS-DC(B), RS-DC(C);
Rule:
   pots1:  idle(x) & ~DC(x,*) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y) [dial(x,y)] calling(x,y).
```

```
   pots4:  dialtone(x) & ~idle(y) [dial(x,y)] busytone(x).
   pots5:  calling(x,y) [onhook(x)] idle(x) & idle(y).
   pots6:  calling(x,y) [offhook(y)] path(x,y) & path(y,x).
   pots7:  path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).
   pots8:  busytone(x) [onhook(x)] idle(x).
   pots9:  dialtone(x) [dial(x,x)] busytone(x).
     dc1:  idle(x) & idle(y) & RS-DC(x) [reg-dc(x,y)] idle(x) & idle(y) & DC(x,y).
     dc2:  idle(x) & idle(y) & DC(x,y) [wdraw-dc(x)] idle(x) & idle(y) & RS-DC(x).
     dc3:  idle(x) & idle(y) & DC(x,y) [offhook(x)] calling(x,y) & DC(x,y).
     dc4:  idle(x) & ~idle(y) & DC(x,y) [offhook(x)] busytone(x) & DC(x,y).
```

B.9 Emergency Call (EMG)

```
Specification DT;
User:       A, B, C;
Var:        x, y;
Predicate: idle(x), dialtone(x), calling(x,y), path(x,y),
           busytone(x), EMG(x), RS-emg(x), emg-hold(x,y);
Event:      onhook(x), offhook(x), dial(x,y), reg-emg(x), wdraw-emg(x);
Init:       idle(A), idle(B), idle(C),
           RS-emg(A), RS-emg(B), RS-emg(C);
Rule:
   pots1:  idle(x) [offhook(x)] dialtone(x).
   pots2:  dialtone(x) [onhook(x)] idle(x).
   pots3:  dialtone(x) & idle(y)  [dial(x,y)] calling(x,y).
   pots4:  dialtone(x) & ~idle(y) [dial(x,y)] busytone(x).
   pots5:  calling(x,y)  [onhook(x)] idle(x) & idle(y).
```

```
pots6:  calling(x,y)  [offhook(y)] path(x,y) & path(y,x).

pots7:  ~EMG(y) & path(x,y) & path(y,x) [onhook(x)] idle(x) & busytone(y).

pots8:  busytone(x) [onhook(x)] idle(x).

pots9:  dialtone(x) [dial(x,x)] busytone(x).

 emg1:  idle(x) & RS-emg(x) [reg-emg(x)] idle(x) & EMG(x).

 emg2:  idle(x) & EMG(x) [wdraw-emg(x)] idle(x) & RS-emg(x).

 emg3:  path(x,y) & EMG(x) [onhook(y)] emg-hold(x,y) & EMG(x).

 emg4:  emg-hold(x,y) & EMG(x) [offhook(y)] path(x,y) & EMG(x).
```