Feature Interactions in Telecommunication Networks IV

Edited by

P. Dini, R. Boutaba and L. Logrippo

Sponsored by
IEEE Communications Society
Centre de recherche informatique de Montréal

With the participation of
University of Ottawa
Université de Montréal
Bell Laboratories, Lucent Technologies

IOS
Press
Ohmsha

Amsterdam, Berlin, Oxford, Tokyo, Washington, DC

Feature Interactions in Telecommunications
and Distributed Systems IV
P. Dini et al. (Eds.)
IOS Press, 1997

Petri-Net Based Detection Method for Non-Deterministic Feature Interactions and its Experimental Evaluation

Masahide Nakamura, Yoshiaki Kakuda and Tohru Kikuno
Department of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University
1-3, Machikaneyama-cho, Toyonaka-shi, Osaka 560, Japan
{masa-n, kakuda, kikuno}@ics.es.osaka-u.ac.jp

Abstract: Non-deterministic feature interaction is one of the most typical feature interactions such as the one between CW & CFV features. The conventional detection algorithm for this interaction generally requires the reachable state enumeration, which may cause the state explosion problem. As a result, it takes a lot of time and space when it is applied to complex services including many users.

To resolve this problem, we have already devised a new detection algorithm based on a Petri-Net model. The new method is characterized by the P-invariant of the Petri-Net which is utilized to check the reachability of the states. Since the checking is efficiently carried out without any state enumeration, we can reduce both the time and space drastically. However, the P-invariant generally gives only necessary condition and thus may detect non-determinism which does not actually occur.

So, in this paper, we evaluate the new algorithm from two viewpoints: detection quality and scalability, through experiments. In the experiments, we have prepared five service specifications for the practical telecommunication services. The experimental results show that the new algorithm achieves such high detection quality that all detected non-determinisms actually occur, and also has a good scalability for complex services such as teleconference including many users. Thus, the Petri-Net based detection method enables us to verify non-deterministic feature interactions in the more complex communication service specification.

1. Introduction

Along with the enrichment of new telecommunication services in Intelligent Network(IN)[14], feature interaction becomes quite a serious problem which prevents the rapid creation of the new services[2]. As a result, it is strongly required to develop efficient methods for the detection and elimination of feature interaction from a given service specification. In order to attain the efficiency, it is desirable that the feature interactions are detected and eliminated in the early stage of service development (that is, service design process) rather than later stages (coding and execution processes)[8].

Therefore, we try to detect and resolve feature interaction at the service specification level. In this paper we adopt the rule-based method for the description of service specifications since it is widely studied toward the practical use such as STR[6] and declarative transition rule[4]. The rule-based description specifies the service as a set of rules, each of which describes a feature of the service. Feature interactions on the service specification have been classified into several types of interactions: deadlock, livelock, non-determinism, transition to abnormal states, etc.[3][8][12]. In this paper, we especially focus on "Non-determinism", and discuss the detection of feature interaction caused by non-determinism.

As studied in [12], telecommunication services can be often modeled by a state transition machine, in which a state consisting of local states of the service users successively moves to a next state by a trigger of user's event. If multiple transitions are allowed to be executed for a certain pair of a state and a user's event, then a non-deterministic transition occurs and it may cause an illegal state change against the user's intention. The interaction between Call Forwarding & Call Waiting and the interaction between Answer Calling & Call Waiting are typical examples of non-deterministic interaction.

So far, Harada et al. proposed an interaction detection algorithm[5], denoted by Algorithm EXH, for the rule-based specifications. This algorithm consists of the following two phases:
(*Phase 1*)Enumerate all possible reachable states by the state enumeration, then
(*Phase 2*)By checking each enumerated state, detect a pair of rules which non-deterministically conflict with each other.
This method can detect the non-deterministic interactions based on necessary-sufficient condition. However, it takes a lot of time and space, because the number of reachable states exponentially increases (so-called state explosion problem occurs) depending on the complexity of the service(i.e., the number of rules) and the number of users in the service. For complex services with many users, such as teleconference, the state explosion problem is quite a serious problem, and thus Algorithm EXH cannot be applied to analyze it.

In order to overcome this problem, we have devised an alternative new detection algorithm, denoted by Algorithm Ω based on the P-invariant of a Petri-Net model[11], which requires no state enumeration. Kawarazaki et al. also indicated an analysis method using the T-invariant of Petri-Net (instead of our P-invariant)[8] to reduce the state space. But this method is also based on the state enumeration, and thus unfortunately it gives no practical evaluation method. In Algorithm Ω, we at first transform the given service specification to the Petri-net, then obtain all pairs of rules which *may* cause the non-determinism. Next, based on the obtained rules, we determine a set of states at which the non-determinisms may occur. Finally, we check if the determined states are reachable from the initial state. For this checking, we extensively utilize the P-invariant of the Petri-net. Since this checking can be performed without any state enumeration, we can succeed in reducing time and space, drastically. However, the P-invariant provides a necessary condition for the reachability checking of the state. Thus, theoretically speaking, the detected pairs of rules do not always cause the non-determinism. So, we have to evaluate the quality of Algorithm Ω by applying it to the practical service specifications.

The goal of this paper is to show the effectiveness of Algorithm Ω through the experimental evaluation. We have performed two experiments. The first experiment evaluates Algorithm Ω from the view points of detection quality and performance. The second experiment examines the scalability, that is, the applicability of Algorithm Ω to the complex service specifications including many users. The experimental results show that Algorithm Ω can achieve high detection quality and scalability for the practical service specifications.

This paper is organized as follows: Section 2 defines the service specification, formalizes the non-deterministic interactions and reviews the previous algorithm EXH. Next, we explain a new algorithm Ω based on the P-invariant of the Petri-net model in Sections 3 and 4. Section 5 presents two experimental evaluations (Experiment 1 and Experiment 2) of Algorithm Ω. Finally Section 6 concludes this paper with future works.

2. Definitions

2.1 Service Specification

In this paper, a service specification is defined as a set of rules of service logic. This service description method is particularly relevant to the existing methods such as STR[6] and declarative transition rules[4] and it enables non-expert to design the service logic[12].

A *service specification* is defined by $S = \langle R, s_0 \rangle$, where R denotes a set of *rules* and s_0 is a *state* called *initial state*.

A *rule* $r \in R$ is defined by the following form:

r: pre-condition [event] post-condition

A *pre(post)-condition* of rule r is an AND-conjunction of predicates $p(x_1, \ldots, x_k)$'s, where p is a predicate symbol and x_i' s are *variables*. Especially, the pre-condition is allowed to have the negation $\neg p(x_1, \ldots, x_k)$ which implies predicate $p(x_1, \ldots, x_k)$ does not holds. Next, an *event* is a predicate $e(x_1, \ldots, x_k)$, where e is an event symbol.

A *global state* (or simply *state*) is also an AND-conjunction of all *instances* of predicates $p(a_1, \ldots, a_k)$'s, where a_i' s are *constants* representing the service users. For convenience, we list up only the instances that take true value.

A state s can be changed to the *next state* s' by applying a rule $r \in R$. Let $r\theta$ be an instanciation of a rule r based on the substitution θ. If any predicate in pre-conditions of $r\theta$ takes the true value at s, then we say r is *enabled* for θ at s. At this time, a next state s' can be generated by deleting all instances in pre-condition of $r\theta$ from s and then adding all instances in post-condition to s (in a similar way to the rewriting operation of production system). Then, we say that state s moved to s' by the trigger of the event of $r\theta$.

A state s is *reachable* iff there exists at least one sequence of states $s_0, s_1, \ldots, s_j = s$ such that s_i $(1 \le i \le j)$ is a next state of s_{i-1}.

Example 1: The following shows an example of service specification $S = \langle R, s_0 \rangle$ which specifies a simplified POTS (Plain Ordinary Telephone Service) with four users (A, B, C and D).

R = { 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: dialtone(x) [dial(x,x)] busytone(x).
pots6: calling(x,y) [offhook(y)] path(x,y) , path(y,x).
pots7: calling(x,y) [onhook(x)] idle(x) , idle(y).
pots8: path(x,y) , path(y,x) [onhook(x)] idle(x) , busytone(y).
pots9: busytone(x) [onhook(x)] idle(x). }

s_0 = idle(A) , idle(B) , idle(C) , idle(D)

At the initial state s0, rule pots1 is enabled for the substitutions <x=A>, <x=B>, <x=C> and <x=D>. For example, we apply pots1 for <x=C> to s_0. Then, idle(C) in s_0 is removed (because it is included in the pre-condition of pots1<x=C>) from s_0, and dialtone(C) is newly added (because it is in the post-condition of pots1<x=C>) to s_0. Thus, we obtain the next state

s_1 = idle(A), idle(B), dialtone(C), idle(D).

Next, let us apply pots3 for the substitution <x=C,y=D> to s_1. Then, we obtain the successive next state

s_2 = idle(A), idle(B), calling(C,D).

Both states s_1 and s_2 are reachable. The above sequence of the rule applications describes a typical scenario of POTS: [All users A, B, C and D are idle (s_0)], then [if user C offhooks, then C receives a dialtone (s_0-offhook(C)->s_1)], and finally [if C dials D, then C is calling D (s_1-dial(C,D)->s_2)].

2.2 Problem Formalization

Here, we formalize the non-deterministic interaction on the service specification. Intuitively, non-deterministic behavior arises when at least two different rules (called conflict rules) with the same event are simultaneously enabled at a certain reachable state.

For a given service specification $S = \langle R, s_0 \rangle$, we say that rules $r_i, r_j \in R$ *conflict* iff there exists such a state s that satisfies both of the following conditions P1 and P2.

Condition P1: s is reachable from s_0.

Condition P2: there exist two substitutions θ_i and θ_j such that r_i and r_j are enabled for θ_i and θ_j at s, respectively, and that the triggered events of $r_i\theta_i$ and $r_j\theta_j$ are identical.

Example 2: Consider the following pair of rules cw1 and cfv14 and state s.

cw1: CW(x), path(x,y), dialtone(z) [dial(z,x)] CW(x), path(x,y), CW-ringing(z,x).

cfv14: CFV(y), forward(y,z), ¬idle(y), idle(z), dialtone(x)
[dial(x,y)] CFV(y), forward(y,z), calling(x,z).

s = CW(A), CFV(A), forward(A,D), path(A,B), path(B,A), idle(D), dialtone(C).

The rule cw1 represents a CW feature saying that CW user x can receive an additional call from the third party z while x is talking with someone y. Next, cfv14 implies a CFV feature saying that CFV user y, who sets the forwarding address to z, can forward the incoming call from x to z. Then, state s means that A subscribes both CW and CFV features, A sets the forwarding address to D, A is talking with B, D is idle and C is ready to dial. Now, suppose that s is reachable.

By the definition of rule application, cw1 is enabled for <x=A,y=B,z=C> under s. Also, cfv14 is enabled for <x=C,y=A,z=D> under s. At this time, since the both events of cw1<x=A,y=B,z=C> and cfv14<x=C,y=A,z=D> are identically "dial(C,A)", Condition P2 is satisfied. So, rules cw1 and cfv14 conflict. This is exactly the well-known feature interaction between CW and CFV[3][12][13].

The **detection problem of the non-deterministic interaction** for the given service specification is defined as the detection of the pairs of conflict rules. In other words, the problem is to identify the states which satisfy both of the conditions P1 and P2. The following shows a schematic classification of states.

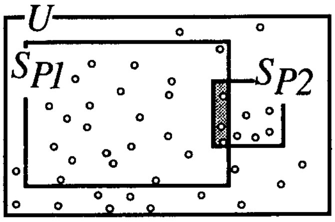

Figure 1. Classification of states

In Figure 1, U denotes a set of all global states, S_{P1} denotes a set of states satisfying Condition P1, and S_{P2} denotes a set of states satisfying Condition P2. Then, the intersection of S_{P1} and S_{P2} (depicted by the shaded part) is a set of states at which the non-deterministic interactions occur.

2.3 Detection Algorithm by Harada et al.

Harada et al. have proposed a detection algorithm[5], denoted by **Algorithm EXH** in the following, which intuitively consists of the following two phases.

Phase1: Enumerate all possible reachable states by exhaustive applications of rules.

Phase2: Check non-deterministic interaction for each state obtained at Phase1.

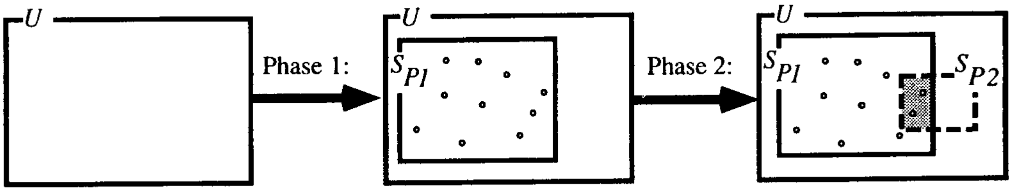

Figure 2. Outline of Algorithm *EXH*

Figure 2 describes the outline of Algorithm EXH. At first, Phase 1 identifies the set S_{P1} by the exhaustive state enumeration (so-called reachability analysis), and then Phase 2 checks Condition P2 for each state in S_{P1}. It is clear that this algorithm can extract the intersection of S_{P1} and S_{P2}. That is, all pairs of conflict rules are exactly determined for a given service specification. However, this algorithm takes a lot of time and space because the size of S_{P1} exponentially grows in accordance with the complexity of the service specification (i.e., the number of rules) and the number of users in the service specification. So, it may be impossible to apply Algorithm EXH to the complex services including many users (which will be mentioned in Section 5).

3. Petri-Net Model

3.1 Service Specification Net

This section presents a Petri-Net model which is logically equivalent to the rule-based service specification defined in Section 2. Here, we give an intuitive explanation of our Petri-Net model using some examples. As for the formal definition of the model, please refer to our earlier paper[11].

Basically, *Petri-net*[10] is a directed graph consisting of *places* (depicted by circles or rounded rectangles), *transitions* (depicted by boxes) and *arcs* (depicted by arrows). In addition to them, our model (called *labeled Pr/T net*) has two kinds of *labels*: one is attached to arcs, and the other is attached to transitions. An arc's label is represented by a tuple of variables (denoted by $\langle x_1, ..., x_k \rangle$), while the transition's label is represented by predicates (denoted by $e(x_1, ..., x_k)$). For a label $\langle x_1, ..., x_k \rangle$ on any arc which is incident on a place p, k is a specific number associated with p and is called *arity* of place p. Furthermore, the labeled Pr/T net can include *inhibitor arc* (depicted by arrows with a circle).

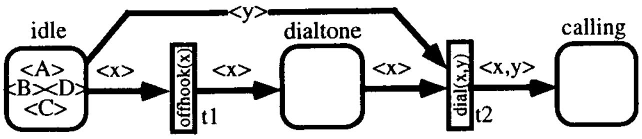

Figure 3. A labeled Pr/T Net

A place p is *input*(or *output*) *place* of a transition t iff there exists a directed arc from p to t (or t to p). For convenience, we denote an arc from a to b by (a,b). An arc is called *surrounding arc* of transition t iff it is specified between t and input/output places of t.

Figure 3 shows an example of a labeled Pr/T net. This net has three places (idle, dialtone and calling), two transitions($t1$ and $t2$), five arcs and no inhibitor arc. Transitions $t1$ and $t2$ have labels offhook(x) and dial(x,y), respectively. Each arc is attached a label such as <x> or <x,y>. Input places of $t2$ are idle and dialtone, the output place of $t2$ is calling. Surrounding arcs of $t2$ are (idle,$t2$), (dialtone,$t2$) and ($t2$,calling). Note that all labels on arcs around a place are tuples with the same arity (e.g., two arcs around place idle have labels of 1-tuple, the arc around place calling has a label of 2-tuple).

Each place p may contain dynamically varying number of constant tuples (denoted by $\langle a_1, \ldots, a_k\rangle$ where k is arity of p), which are called *tokens*. An arbitrary distribution of tokens on the places is called a *marking* and it is also expressed by a vector. Consider again Figure 3. Place idle have four tokens <A>,<B>,<C> and <D>, and other two places have no token. Thus, this marking(say, M_0) is expressed as:

$$\begin{array}{lccc} & \mathit{idle} & \mathit{dialtone} & \mathit{calling} \\ M_0= & [\ \{\langle A\rangle, \langle B\rangle, \langle C\rangle, \langle D\rangle\} & \phi & \phi\ ] \end{array}$$

Consider a transition t and a marking M. Let θ denote a substitution of all variables appearing on labels of surrounding arcs of t with constants. Then, we say transition t is *enabled* for θ under M iff each input place of t contains at least the number of tokens prescribed by the substituted labels of the corresponding input arcs (If the input arc is an inhibitor arc, the input place never contains the corresponding token).

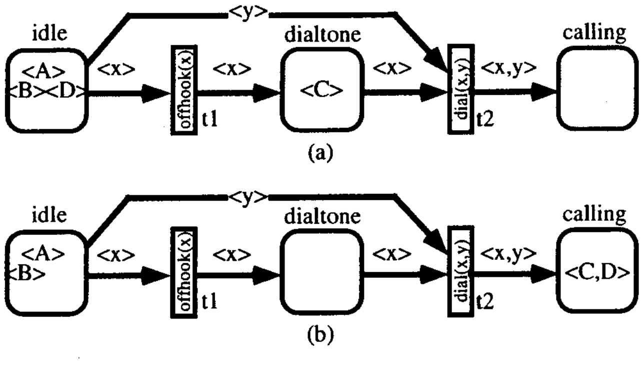

Figure 4. Example of firing

Consider a marking M_1 in Figure 4(a) which is also specified by

$$\begin{array}{lccc} & \mathit{idle} & \mathit{dialtone} & \mathit{calling} \\ M_1 = & [\ \{\langle A\rangle, \langle B\rangle, \langle D\rangle\} & \{\langle C\rangle\} & \phi\ ] \end{array}$$

For example, take transition $t2$. Since there are two variables x and y on the labels of surrounding arcs of $t2$, consider a substitution θ=<x=C,y=D>. Then, the label on arc (dialtone,$t2$) is substituted with <C>, and arc (idle,$t2$) is substituted with <D>. For this substitution, since both input places of $t2$, dialtone and idle, respectively contain tokens <C> and <D>, $t2$ is enabled for θ under M_1.

If a transition t is enabled for θ under M, then t can *fire*. Firing of t changes the current marking M into the *next marking* M'. The effect of a firing is that tokens are removed from input places and added to output places. The removed/added tokens are specified by the labels on input/output arcs of t, substituted by θ. A marking M is *reachable* from M_0 iff there exists at least one sequence of markings $M_0, M_1, \ldots, M_j = M$, where M_i $(1 \le i \le j)$ is a next marking of M_{i-1}.

Consider again the marking M_1 in Figure 4(a). Suppose that $t2$ fires for θ=<x=C,y=D> under M_1. Then tokens <C> and <D> are respectively removed from input places dialtone and idle. Moreover, a new token <C,D> is added to output place calling because θ substitutes the label on arc ($t2$,calling) with <C,D>. Thus, we obtain the next marking M_2 shown in Figure 4(b), which is also specified by:

$$\begin{array}{rcccl} & & idle & dialtone & calling \\ M_2 = & [& \{\langle A\rangle, \langle B\rangle\} & \phi & \{\langle C, D\rangle\} \;] \end{array}$$

The behavior of the labeled Pr/T net reflects an important aspect of the service specification. Suppose that we interpret a marking on the net as a state of the service specification as follows: "when an instance of predicate $p(a_1, \ldots, a_k)$ holds at state s, place p of the net contains a token $\langle a_1, \ldots, a_k\rangle$ under marking M." Then, the firing of the net is just the same as the rule application of the service specification. Moreover, on a firing of a transition t for a substitution θ, we can correspond the label of t evaluated by θ to a triggered event.

For example, the markings M_0, M_1 and M_2 mentioned above are interpreted as the following three states s_0, $s_1$and s_2, respectively.

s_0 = idle(A), idle(B), idle(C),idle(D)
s_1 = idle(A), idle(B), dialtone(C), idle(D)
s_2 = idle(A), idle(B), calling(C,D)

The firing sequence from M_0 through M_2 is just a sequence of rule applications from s_0 to s_2, which is explained in Example 1.

Of course, in order to completely trace the behavior of the given service specification, the labeled Pr/T net has to possess a specific net structure for the given service specification. We call this particular labeled Pr/T net *service specification net.*

A service specification net $N(S)$ for a given service specification S is a labeled Pr/T net which satisfies the following seven conditions:

(1) The set of places in $N(S)$ is a set of all predicate symbols in S.

(2) Each transition t_i in $N(S)$ corresponds to exactly one rule r_i in S.

(3) Each transition t_i in $N(S)$ has an event of rule r_i as a label.

(4) For each predicate $p_{ij}(x_{i1}, \ldots, x_{ik})$ in pre-condition of rule r_i, $N(S)$ has exactly one arc (p_{ij}, t_i) with a label $\langle x_{i1}, \ldots, x_{ik}\rangle$.

(5) For each negation of predicate $\neg p_{ij}(x_{i1}, \ldots, x_{ik})$ in pre-condition of rule r_i, $N(S)$ has exactly one inhibitor arc (p_{ij}, t_i) with a label $\langle x_{i1}, \ldots, x_{ik}\rangle$,

(6) For each predicate $p_{ij}(x_{i1}, \ldots, x_{ik})$ in post-condition of rule r_i, $N(S)$ has exactly one arc (t_i, p_{ij}) with a label $\langle x_{i1}, \ldots, x_{ik}\rangle$,

(7) If the initial state s_0 of S includes an instance of predicate $p(a_1, \ldots, a_k)$, then $N(S)$ has a token $\langle a_1, \ldots, a_k\rangle$ on place p.

Remark 1: A similar Petri-Net model for the rule-based service specification is presented in [8]. However, our service specification net can describe the service specification more precisely in the sense that (1) inhibitor arcs are introduced to deal with the negations in preconditions of rules, (2)the label is attached to each transition to explicitly represent an event of the corresponding rule.

For example, consider again the labeled Pr/T net in Figure 3. Then, we can easily understand that this net is a service specification net *N*(*S*) for the service specification $S = \langle R, s_0 \rangle$. where

R = { r1: idle(x) [offhook(x)] dialtone(x).
r2: dialtone(x), idle(y) [dial(x,y)] calling(x,y). }
s_0 = idle(A), idle(B), idle(C), idle(D)

We can completely simulate the behavior of *S* on *N*(*S*). The following lemma holds for *N*(*S*) which implies that *N*(*S*) is logically equivalent to *S* with respect to the reachable states.

Lemma 1: *For a given service specification S and a service specification net N(S) for S, there exists one-to-one correspondence between a set of reachable markings of N(S) and a set of reachable states of S.*

3.2 P-invariant Method

A P-invariant is known as a powerful tool for the analysis of Petri-Net and it provides an important condition with respect to the reachable marking. The P-invariant of a service specification net with *m* places is expressed by an *m*-dimensional vector *Y* whose elements are linear functions and can be calculated from only net structure(i.e., independent of the marking(state)). This implies that the P-invariant is obtained independent on the number of users. Since the P-invariant of our service specification net is the same as that of the colored Petri-net[7], we omit the detail of it in this paper. For the formal definition and calculation method of the P-invariant, readers can refer to [1][7]. The following theorem is well-known theorem with respect to the reachability of markings.

Theorem 1: *Let Y be a P-invariant of the service specification net N(S). If marking M is reachable from the initial marking M_0, then $Y*M^t=Y*M_0^t$, where * is a formal product operation of matrix*[1][7].

Remark 2: The equation $Y*M^t=Y*M_0^t$ is only 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.

Intuitively, the P-invariant generates an equation which always holds for any reachable marking. In other words, for an arbitrary marking *M* which violates this equation, we can conclude that *M* is not reachable without any state enumeration. Since a state *s* uniquely corresponds to a marking *M* according to Lemma 1, thus we can check the reachability of any state *s* by the P-invariant.

Example 3: Consider again the service specification net *N*(*S*) in Figure 3, and markings M_0, M_1 and M_2 discussed in Section 3.1. Then, the P-invariant of *N*(*S*) is

$$\begin{array}{ccccc} & & idle & dialtone & calling \\ Y = & [& id & id & p1 + p2 \;] \end{array}$$

where, *id*, *p*1 and *p*2 are linear functions such that (1) *id* is identity function: *id*<x>=<x>, (2) *p*1 and *p*2 are projection functions: *p*1<x,y>=<x> and *p*2<x,y>=<y>. Let us apply *Y* to markings M_0, M_1 and M_2. In the following, + denotes a union operation defined on multiset, and we suppose (*p*1+*p*2)<x,y>=*p*1<x,y>+*p*2<x,y>. Intuitively, operation *Y**M* is performed like inner product of two vector: (1) evaluate the tokens on each place in *M* by the corresponding linear function of P-invariant *Y*, then (2) make a union of multiset of obtained results.

$Y*M_0$ = *id*{<A>,<B>,<C>,<D>}+*id*{ }+(*p*1+*p*2){ }={*id*<A>, *id*<B>, *id*<C>, *id*<D>}
= {<A>,<B>,<C>,<D>}

$$Y{*}M_1 = id\{<A>,<B>,<D>\}+id\{<C>\}+(p1+p2)\{\ \}=\{<A>,<B>,<C>,<D>\} = Y{*}M_0$$

$$Y{*}M_2 = id\{<A>,<B>\}+id\{\ \}+(p1+p2)\{<C,D>\}=\{id<A>,id<B>\}+\{p1<C,D>\}+\{p2<C,D>\}$$

$$= \{<A>,<B>\}+\{<C>\}+\{<D>\} = \{<A>,<B>,<C>,<D>\}=Y{*}M_0$$

Next consider the following marking M and apply P-invariant Y to M

$$\begin{array}{llll} & \textit{idle} & \textit{dialtone} & \textit{calling} \\ M = [& \{\langle A\rangle, \langle B\rangle\} & \{\langle A\rangle\} & \{\langle C, D\rangle\}\ ] \end{array}$$

$$Y{*}M = id\{<A>,<B>\}+id\{<A>\}+(p1+p2)\{<C,D>\}=\{<A>,<B>\}+\{<A>\}+\{<C>\}+\{<D>\}$$

$$= \{2<A>,<B>,<C>,<D>\} \neq Y{*}M_0$$

Thus, we can conclude M is not reachable from M_0. Note that this fact is derived without any marking (state) enumeration.

4. New Detection Algorithm Ω

4.1 Definition of Algorithm Ω

Now we present a new detection algorithm, denoted by **Algorithm Ω**, using the P-invariant. Figure 5 depicts an outline of Algorithm Ω. At first, Phase 1 obtains a set S_{P2} of states which satisfy Condition P2 based on the rules of the given service specification S. Any state in S_{P2} can be easily calculated by joining pre-conditions of such two rules that have the same event symbol. Then, in Phase 2, we delete the states using the P-invariant which are not reachable from the initial state.

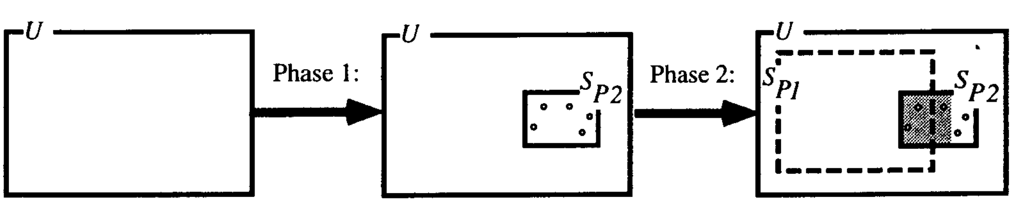

Figure 5. Outline of Algorithm Ω

The following shows a brief description of the new detection algorithm Ω. In the next subsection, we will give a detailed explanation of each step using examples.

Detection Algorithm Ω:

Phase 0(Preliminary): Construct a service specification net $N(S)$ for a given service specification S=<R,s_0>. Then calculate the P-invariant Y of $N(S)$. Initialize a set S_{P2} to be empty.

Phase 1(Decision of States in S_{P2}):

Step 1: Select two rules r_i and r_j from R whose event symbols are identical.

Step 2: Apply a pair of substitution θ_i and θ_j to rules r_i and r_j, respectively, such that two events of $r_i\theta_i$ and $r_i\theta_j$ are identical.

Step 3: For two pre-conditions c_i and c_j of $r_i\theta_i$ and $r_i\theta_j$, respectively, obtain a condition $C_{ij} = c_i, c_j$. If C_{ij} forms a null condition, we conclude that r_i and r_j never conflict(i.e., mutual exclusive) with each other, and go to Step 5. Otherwise, go to Step 4.

Step 4: Based on predicates in C_{ij}, put tokens to the corresponding place of $N(S)$. Then, for each place p, put the *wild-card of token* $\Sigma\langle x_{p1}, \ldots, x_{pk}\rangle$ where x_{pi} is a variable to which any constant value can be assigned, and k is arity of p. Let a marking M_{ij} to be the resultant marking. Finally, put M_{ij} into S_{P2}.

Step 5: If some pairs of rules to be selected still remain, then go to Step1.

Phase 2(Check of Reachability using P-invariant): For each marking M_{ij} in S_{P2}, evaluate an equation $Y{*}M_{ij}{}^{t} = Y{*}M_0{}^{t}$. If there exists no assignment of constant value to wildcards of M_{ij} such that $Y{*}M_{ij}{}^{t} = Y{*}M_0{}^{t}$ holds, then we conclude M_{ij} is not reachable (that is, rules r_i and r_j never conflict with each other), and delete M_{ij} from S_{P2}.

As for Algorithm Ω, the following theorem holds[11].

Theorem 2: *For the given service specification $S = \langle R, s_0 \rangle$, if rules $r_i, r_j \in R$ conflict, then the resultant set S_{P2} obtained by Algorithm Ω surely contains the marking Mij constructed at Step4.*

Using this theorem, we can convince that, for a given service specification S, rules r_i and r_j never conflict if the resultant S_{P2} does not include M_{ij}. Moreover, if the resultant S_{P2} becomes an empty set, then S is free from the non-deterministic interaction.

Since Algorithm Ω requires no state enumeration, the time and space needed to the analysis are thus drastically reduced compared with Algorithm EXH. Instead, theoretically speaking, the resultant S_{P2} may contain such states that do not belong to the intersection of S_{P1} and S_{P2}(see Figure 5), because the converse of Theorem 2 does not necessarily hold(the reason is that the P-invariant is used as necessary condition of reachability). That is, Ω may detect the redundant pairs of rules which does not actually conflict. So, in order to show the effectiveness of the algorithm for the practical use, we must evaluate the *quality of detection* (i.e., how small the number of such redundant pairs is) by applying Ω to practical service specifications.

Remark 3: In order to exactly determine the intersection of S_{P1} and S_{P2}, we have to apply the reachability analysis to the resultant S_{P2}. Thus, in our earlier paper[11], additional Phase 3 is introduced for the reachability analysis. However, the cost needed to the reachability analysis may be, at the worst case, as expensive as that of Algorithm EXH. So, in this paper, we discuss only Phases 0, 1 and 2, and evaluate the result obtained by the necessary condition.

4.2 Aplication to POTS Specification

We explain Algorithm Ω through an application to POTS specification in Example 1.

Phase 0(Preliminary):

First, the mapping of the specification S onto net model $N(S)$ can be easily done based on the definition of $N(S)$. Then, we calculate the P-invariant Y of $N(S)$ by applying any available method[1][7], and make the set S_{P2} to be empty. For example, from POTS specification S shown in Example 1, we can obtain the following P-invariant Y and initial marking M_0

$$\begin{array}{lccccc} & idle & dialtone & calling & busytone & path \\ Y = [& id & id & p1 + p2 & id & p1\ ] \end{array}$$

$$\begin{array}{lccccc} & idle & dialtone & calling & busytone & path \\ M_0 = [& \{\langle A\rangle, \langle B\rangle, \langle C\rangle, \langle D\rangle\} & \phi & \phi & \phi & \phi\ ] \end{array}$$

Phase 1(Decision of States in S_{P2}):

As an example, let us consider rules pots1 and pots6 from POTS specification. By applying substitutions <x=A> and <x=B, y=A> to pots1 and pots6, respectively, we get

pots1<x=A>: idle(A) [offhook(A)] dialtone(A).
pots6<x=B,y=A>: calling(B,A) [offhook(A)] path(A,B), path(B,A).

Then by combining two pre-conditions of these rules, we get a condition C_{16} as follows:

C_{16} = idle(A), calling(B,A)

Note that the combined condition C_{ij} sometimes forms null condition because of the negation of predicate. At this time, we conclude that rules r_i and r_j never conflict(e.g., pots3 and pots4 never conflict).

Next, consider the following states s' and s'' both of which include C_{16}:

s' = idle(A), calling(B,A), path(C,D), path(D,C)

s'' = idle(A), calling(B,A), idle(C), busytone(D)

Then, it is clear that both s' and s'' satisfy Condition P2, because rules pots1 and pots6 are enabled for <x=A> and <x=B,x=A> under s' (also s''), respectively, and the events of both rules are identical to "offhook(A)". That is, any state including C_{16} must satisfy Condition P2. Generally speaking, there are a lot of different states including predicates of C_{16} such as s' and s''. So, in order to deal with all possible states satisfying C_{16} in a convenient way, we suppose that all predicates, except those of C_{16}, should be "don't care" and we assign *wildcards* to them. The wildcard is expressed as a multi-set of tokens with variables to which any constant value is allowed to be assigned[11].

For example, we obtain the following marking M from C_{16} and extend M to M_{16} by putting a wild card on each place. Considering the arity of each place, we put five wildcards as follows:

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & path \\ M = [& \{\langle A\rangle\} & \phi & \{\langle B, A\rangle\} & \phi & \phi\ ] \end{array}$$

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & path \\ M_{16} = [& \{\langle A\rangle, \Sigma\langle x_1\rangle\} & \Sigma\langle x_2\rangle & \{\langle B, A\rangle, \Sigma\langle x_3, x_4\rangle\} & \Sigma\langle x_5\rangle & \Sigma\langle x_6, x_7\rangle\ ] \end{array}$$

From the interpretation of marking discussed in Section 3, we understand that M corresponds to C_{16}. Here, we show that M_{16} represents any arbitrary state including C_{16}. Consider the following two assignments $a1$ and $a2$ of tokens to wildcards.

$a1$ = [Σ<x1>=Σ<x2>=Σ<x3,x4>=Σ<x5>=φ , Σ<x6,x7>={<C,D>,<D,C>}]

$a2$ = [Σ<x1>={<C>}, Σ<x2>=Σ<x3,x4>=Σ<x6,x7>=φ, <x5>={<D>}]

If we respectively apply $a1$ and $a2$ to M_{16}, we can get the following markings M' and M'':

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & path \\ M' = [& \{\langle A\rangle\} & \phi & \{\langle B, A\rangle\} & \phi & \{\langle C, D\rangle, \langle D, C\rangle\}\ ] \end{array}$$

$$\begin{array}{rccccc} & idle & dialtone & calling & busytone & path \\ M'' = [& \{\langle A\rangle, \langle C\rangle\} & \phi & \{\langle B, A\rangle\} & \{\langle D\rangle\} & \phi\ ] \end{array}$$

Then, M' and M'' respectively correspond to state s' and s'', and thus we can see that M_{16} surely represents any state including C_{16}. Finally, the marking M_{16} is put into S_{P2} as a candidate of a state at which rules pots1 and pots6 conflict.

Phase 2(Check of Reachability by P-invariant):

According to Theorem 1, if the equation $Y{*}M_{ij}{}^t = Y{*}M_0{}^t$ does not hold, then we conclude that M_{ij} is not reachable, i.e., rules r_i and r_j never conflict. On the evaluation of $Y{*}M_{ij}{}^t = Y{*}M_0{}^t$, if there is no assignment of tokens to wild-cards such that $Y{*}M_{ij}{}^t = Y{*}M_0{}^t$ holds, then the equation never holds and thus we delete M_{ij} from S_{P2}. Otherwise, we cannot derive any decision on the reachability of M_{ij}, thus we leave M_{ij} in S_{P2}(see Remark 2).

As an example, let us evaluate $Y{*}M_{16}{}^t = Y{*}M_0{}^t$.

$Y{*}M_0$ = *id*{<A>,<B>,<C>,<D>}+*id*{ }+(*p*1+*p*2){ }+*id*{ }+*p*1{ } = {<A>,<B>,<C>,<D>}

$$\begin{aligned} Y{*}M_{16} &= id\{<\mathrm{A}>,\Sigma\langle x_1\rangle\}+id\{\Sigma\langle x_2\rangle\}+(p1+p2)\{<\mathrm{B},\mathrm{A}>,\Sigma\langle x_3, x_4\rangle\}+id\{\Sigma\langle x_5\rangle\}+p1\,\{\,\Sigma\langle x_6, x_7\rangle\,\} \\ &= \{2<\mathrm{A}>,<\mathrm{B}>,\ \Sigma\langle x_1\rangle,\ \Sigma\langle x_2\rangle,\Sigma\langle x_3\rangle,\Sigma\langle x_4\rangle,\Sigma\langle x_5\rangle,\Sigma\langle x_6\rangle\} \end{aligned}$$

For this, no matter how nicely we choose the assignment to the wild-cards, $Y{*}M_{16}{}^t = Y{*}M_0{}^t$ never holds, that is, M_{16} does not satisfy Condition P1. So, we can conclude that rules pots1 and pots6 never conflict. Similarly, to other markings in S_{P2}, we apply the P-invariant. Finally, resultant S_{P2} of POTS specification has become empty. Thus, we can conclude that POTS specification is free from conflict rules, and that non-determinism never occurs.

5. Experimental Evaluation

To measure the applicability of the proposed method to the practical feature interaction detection, this section presents two experiments.

5.1 Preliminaries

For the experiments, we have developed a software written in C language about 5000 lines of code, which can execute both Algorithm EXH[5] and Algorithm Ω. The experiments were performed on a UNIX Workstation (Sun Sparc UA-1) with 448MB main memory. UNIX's "time" command was utilized for measuring the execution time.

Next, we have prepared the rule-based service specifications for the following practical telecommunication services: CW(Call Waiting), CFV(Call Forwarding Variable), DC(Direct Connect), DO(Dined Origination) and DT(Denied Termination). These service specifications were described based on ITU-T Recommendations[14] and BellCore's LSSGR[15], which are also referred in [6][4][12]. Finally, we have combined each pair of service specifications in order to analyze the non-deterministic interaction between any two services. Thus, we obtained total ten(= $_5C_2$) combined specifications: (1) CW&CFV, (2) CW&DC, (3) CW&DO, (4) CW&DT, (5) CFV&DC, (6) CFV&DO, (7) CFV&DT, (8) DC&DO, (9) DC&DT and (10) DO&DT.

Here, we present two kinds of experiments: Experiment 1 and Experiment 2. The objectives of these experiments are summarized as follows.

Experiment 1: Investigate the quality of detections obtained by the proposed algorithm Ω. At the same time, measure the performance of Algorithm Ω .

Experiment 2: Examine the scalability of Algorithm Ω . That is, we study the applicability to the complex services including many users by comparing the state space needed.

5.2 Experiment 1(Detection Quality and Performance)

In this experiment, we applied both Algorithm Ω and Algorithm EXH to ten practical service specifications prepared in Section 5.1, and compared the detected pairs of conflict rules.

As mentioned before, for any service specification, the pairs of rules detected by Algorithm EXH surely cause the non-deterministic interactions. On the other hand, Algorithm Ω may detect the redundant pairs of rules which do not actually conflict. So, by comparing the detected pairs, we can measure the detection quality of Algorithm Ω. That is, the fewer such redundant pairs are, the better the quality of Algorithm Ω is. As for the execution time, it is expected that Algorithm Ω can attain the drastically better performance than that of Algorithm EXH because Algorithm Ω never faces with so-called state explosion problem. Since the execution time may depend on the implementation of the tool, we especially devised the programming so that Algorithm EXH can give the best performance by using several techniques such as the compression of data and hash searching.

The number of users assumed in each service specification is only *three*. Because of memory overflow of the workstation caused by state explosion, Algorithm EXH couldn't deal with more than three users on our environment (Of course, Algorithm Ω can be applied to the cases with more users. This will be discussed in Experiment 2).

Table 1 shows the result of Experiment 1. The contents of columns are: name of service specification, the number of rules in the specification, the numbers of conflict pairs of rules detected by Algorithm Ω and Algorithm EXH, the execution times of Algorithm Ω and Algorithm EXH, and performance of Algorithm Ω (i.e., a ratio of the execution time of Algorithm EXH to that of Algorithm Ω).

Table 1. Result of Experiment 1

Specifications		# of detected pairs		Execution time (sec.)		
Services	# of rules	Ω	EXH	Ω	EXH	Ratio
CW&CFV	43	5	5	3.4	58061.4	17076
CW&DC	27	0	0	1.4	270.4	193
CW&DO	26	0	0	1.3	89.3	69
CW&DT	26	1	1	1.3	187.2	144
CFV&DC	33	0	0	0.8	22046.9	27559
CFV&DO	32	0	0	0.7	7700.3	11000
CFV&DT	32	8	8	0.8	14984	18730
DC&DO	16	2	2	0.2	57.9	290
DC&DT	16	0	0	0.2	69.6	348
DO&DT	15	0	0	0.1	15.7	157

From this table, we can see Algorithm Ω never detected the redundant pairs of conflict rules. That is, we can conclude that, for these ten practical service specifications, Algorithm Ω achieved very high quality. All of the detected interactions are well-known non-deterministic interactions, as are often studied in many papers[2][4][12][13]. In addition to the quality, the performance of Algorithm Ω is about 28000 times of Algorithm EXH on the maximum.

From this experiment, we can say our method provides both high detection quality and much lower cost quite enough for the practical applications.

5.3 Experiment 2(Scalability)

Experiment 2 examines a *scalability* of Algorithm Ω (that is, applicability to more complex services such as teleconference). As mentioned in Section 5.2, Experiment 1 was performed under the very strict assumption that each service contains only three users. When we allowed four users, the execution time of Algorithm EXH exponentially increased and our tool ran out of memory(Actually, our work station began to use the virtual memory. However, due to its frequent swapping, analysis of CW&CFV specification did not complete even in four days). So, due to the lack of the scalability, it may be impossible to apply Algorithm EXH to complex services including many users.

In Experiment 2, we try to investigate the scalability of Algorithm Ω and compare it with Algorithm EXH. We had selected POTS specification in Example 1 as the service specification. Concretely, we have varied the number of users in the specification. Then, we measured the number of states needed to each algorithm and observe the growth of state space with the increase of the user. Actually, for Algorithm EXH the number of enumerated states is measured, on the other hand, for Algorithm Ω the number of states determined in Phase 1 (i.e., the size of S_{P2}) is measured.

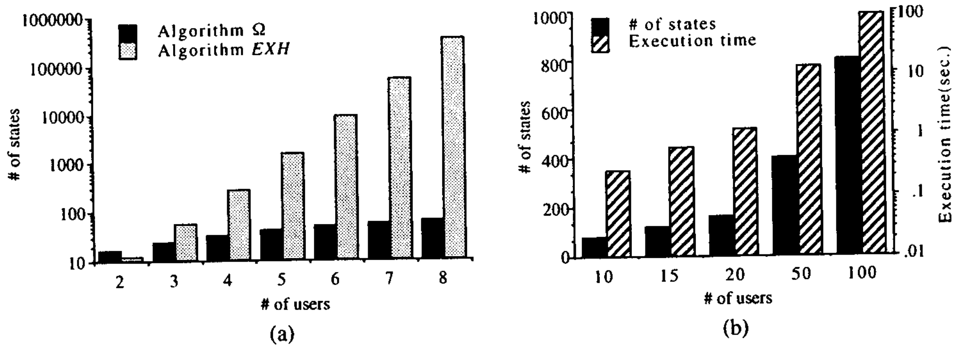

Figure 6. Result of Expriment 2: (a) Comparison of state space, (b) Cases with many users

Figure 6 shows the result of Experiment 2. In Figure 6(a), x and y axis represent the number of users and the number of states, respectively. From this graph, we can see that the state space of our Algorithm Ω grows much slower than that of Algorithm EXH. In the experiment, our tool began to run out of memory from the case of 8 users of Algorithm EXH, and the execution speed had slowed down. Finally, we could not apply Algorithm EXH to the case of 9 users.

On the other hand, Algorithm Ω could work well for cases of much more users. Figure 6(b) shows the number of states and execution time needed to Algorithm Ω. From this graph, we can see that Algorithm Ω can execute even for the case of 100 users in a very short time (86.44 sec.) and smaller space (800 states). Thus, we can say Algorithm Ω has much more scalability than Algorithm EXH as for POTS specification. Additionally, Algorithm Ω could also handle CW&CFV specification with 40 users and its execution was finished in about an hour.

Although this experimentation is performed only for the POTS specification, the above fact implies that Algorithm Ω has a good scalability concerning the number of users. Therefore, it is expected to be applied to the analysis of complex services including many users.

5.4 Discussion

Theoretically speaking, Algorithm EXH detects the conflict rules based on necessary and sufficient condition, while Algorithm Ω determines them by applying only necessary condition (i.e., Algorithm Ω may sometimes detect the redundant pairs that do not actually conflict). Thus, we cannot directly compare these two algorithms. However, for a very complex service specification containing many users, Algorithm EXH cannot avoid the state explosion and thus may not be applied to the analysis in a short time. Then, it prevents the rapid creation of new services and quick adaptation to the service requirement of customers. Instead, Algorithm Ω gives us a useful information of *suspected* conflicts (in the sense that some redundant conflicts may be included) in a short time. Additionally, if we can resolve all the suspected conflicts obtained by Algorithm Ω, then the specification is sufficiently free from non-deterministic interaction. This approach is quite reasonable and realistic for the practical service development. This kind of resolution of non-deterministic interaction is discussed in many papers, and is also one of our future research works.

The key idea of Algorithm Ω is to utilize the P-invariant of the Petri-Net for the checking of the reachability of states. For other types of interaction discussed in [8][12], this reachability of state is a fundamental and essential property needed to be detected (e.g., transition to abnormal state[12]). Therefore, the P-invariant method on our Petri-net model is expected to be applicable for the detection of other feature interactions as well as non-determinism.

6. Conclusion

In this paper, we have presented a new detection algorithm Ω for non-deterministic feature interaction based on the P-invariant of the Petri-net model. Then we also evaluated and examined its effectiveness for the practical feature interaction problem through experiments. The evaluation results showed that our method attain high quality of detection, drastic performance improvement, and good scalability concerning the number of users. As a result, we can expect that Algorithm Ω is applicable to the interaction detections of more complex services with many users.

The followings summarize our future research works:

(a) Application of Algorithm Ω to other practical service specifications with many users.
(b) Improvement of Algorithm Ω in order to apply it to other types of feature interactions.
(c) Examination of an efficient resolution scheme for the detected interactions.

Acknowledgement

The authors would like to thank Mr. Nobuhiko Ido and Mr. Takuro Ikeda (of Kikuno Lab.,Osaka Univ.) for their cooperation with the experiments and drawing figures. This research was partly supported by Kokusai Denshin Denwa Co., Ltd. (KDD).

References

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

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

[3] 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, 1994.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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, 1995.

[9] 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.

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

[11] Nakamura, M., Kakuda Y, and Kikuno T., "Analyzing non-determinism in telecommunication services using P-invariant of Petri-Net model," Proc. of IEEE INFOCOM'97, Apr. 1997, to appear.

[12] 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.

[13] 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.

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

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