Test Suite Generation Methods for Concurrent Systems based on Coloured Petri Nets

Harumi Watanabe

Department of Computer Science
Tokyo Institute of Technology
Meguro, Tokyo 152 JAPAN

Tomohiro Kudoh

Department of Information Technology
Tokyo Engineering University
Hachioji, Tokyo 192 JAPAN

Abstract

Automatic generation of test suites for concurrent systems is a newly exploited area of conformance tests. A few methods based on Finite State Machine (FSM) have been proposed recently. However, these methods require a large amount of computation costs. By using Coloured Petri net (CPN), the required amount of computation costs can be reduced, and the length of the test suites can be reduced by using the equivalent marking technique on CPN. In addition, these methods allow us to test interaction parameters of concurrent systems.

In this paper, we propose two CPN based test suite generation methods for conformance tests: the Coloured Petri net Tree (CPT) method and the Coloured Petri net Graph (CPG) method. An experimental Test Suite Generator(TSG) based on CPT method is presented. To show the advantages of the CPT and CPG methods, the effects of test suite length reduction by equivalent markings are evaluated.

1 Introduction

In the field of communication protocols, automatic generation of test suites for conformance tests is widely used, and various kinds of FDT(Formal Description Technique) based methods [2] [4] [7] have been proposed and practiced. However, for systems whose components communicate each other, conformance tests are done by test suites generated by hand, random inputs, abstraction using rough models, or benchmarks[19]. One of the considerable reasons of the fact is that it is hard to generate test suites for concurrent systems. Typical examples of such systems are operating systems and real time systems. In the field of communication protocols, abstractions that ignore interaction parameters are widely used to avoid this problem.

Some works of conformance test suites generation for concurrent systems have been reported[1][18][2]. These methods are based on Finite State Machine(FSM). To test interaction parameters, in these methods, concurrent systems are specified in a CNFSM (Communicating Nondeterministic Finite-State Machine) which consists of multiple NFSMs(Nondeterministic Finite-State Machine), each of which corresponds to a component of the concurrent system. Since a test suite can not be directory generated from a CNFSM, there is need to construct an integrated NFSM from the CNFSM. The integrated NFSM is a Cartesian product of the NFSMs in the original CNFSM. The main problem of these methods is that the generated integrated NFSM may have many unreachable states, and a huge amount of computation costs is required to eliminate these states.

On the other hand, essentially no unreachable states exist in a reachability tree generated from a Petri net, since a specification in Petri net involves constraints among concurrent components. While modeling a large system using a conventional Petri net is difficult, recently proposed Coloured Petri Nets have the possibility to solve this problem[15].

In this report, we propose two test suites generation methods based on the Coloured Petri Nets: CPT(Coloured Petri net Tree) method and CPG(Coloured Petri net Graph) method. These methods have the following advantages: the required amount of computation costs is smaller than existing methods based on FSM, and the length of test suites can be reduced by using the equivalent marking technique. In addition, since the Coloured Petri net Graph(CP-graph) can be regarded as an FSM, existing methods based on FSM can be applied, and thus the CPG method has the same coverage as existing methods based on FSM.

In section 2, existing test suites generation meth-

0-8186-7171-8/95 $04.00 © 1995 IEEE

ods for concurrent systems are surveyed. In section 3, Coloured Petri Nets and equivalent marking are shown. In section 4, CPT method and CPG method are proposed. In section 5, an experimental Test Suite Generator(TSG) is presented. In section 6, the effects of test suite length reduction by equivalent markings are evaluated. Some discussions on the proposed methods are shown in section 7. Section 8 summarizes this paper.

2 Existing test suites generation methods for concurrent systems

In this section, existing test suites generation methods for concurrent systems are described[1][2][3][18]. In general, test suites for concurrent systems are generated as shown in figure 1. The NFSM(Nondeterministic Finite-State Machine) is a kind of FSMs that has transitions allowing to receive multiple inputs at the same time. The CNFSM(Communicating Nondeterministic Finite-State Machine) consists of multiple NFSMs, channels that combine NFSMs, and message queues.

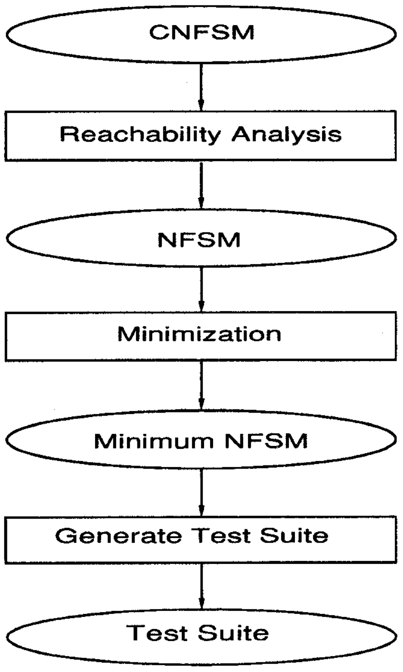


Figure 1: An Example of existing test suites generation method for concurrent systems.

Here, We will explain the above method using the "five dining philosophers". The philosophers, $PH = \{ph_1, ph_2, \cdots, ph_5\}$, are sitting a round a round table with only five forks, $F = \{f_1, f_2, \cdots, f_5\}$, positioned between them as shown in figure 2. Each philosopher alternates EATING and THINKING. To eat spaghetti, a philosopher ph_i needs to take both of the two forks surrounding to him.

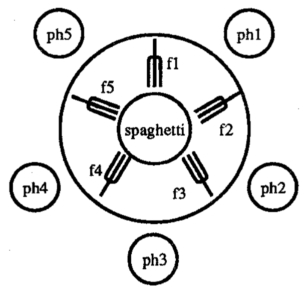


Figure 2: Five Dining Philosophers

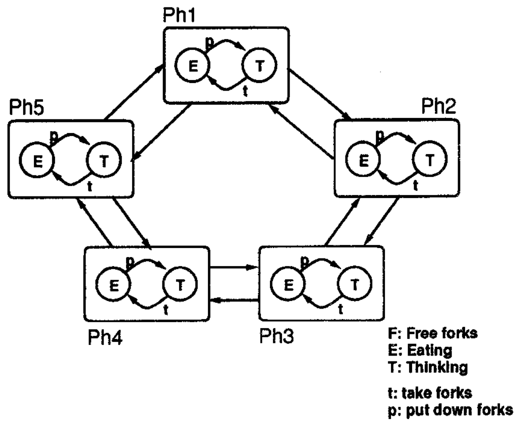


Figure 3: CNFSM for the philosopher system.

The CNFSM of "five dining philosophers" is shown in figure 3. Each rectangle in this figure is an NFSM that represents the states of a philosopher. Each arc combining these rectangles represents channels and message queues that are elements of the CNFSM.

Since the CNFSM cannot generate test suites that enable test of interaction parameters of philosophers, it is needed to construct an integrated NFSM from the CNFSM. Here, we construct the integrated NFSM from ph_1, ph_2, ph_3. since each philosopher has equivalent states. The NFSM is shown in figure 4.

As this figure shows, the NFSM has some unreachable states, since when we construct integrated NFSM,

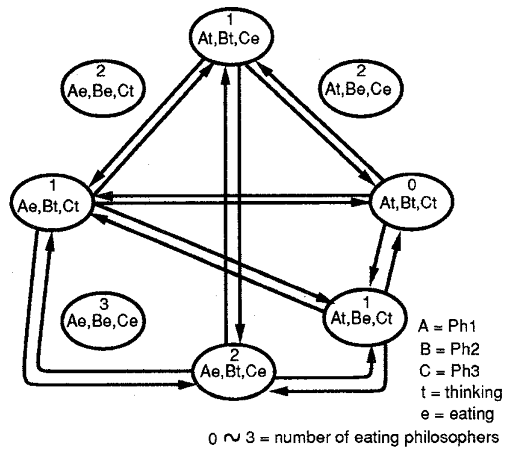


Figure 4: NFSM for the philosopher system.

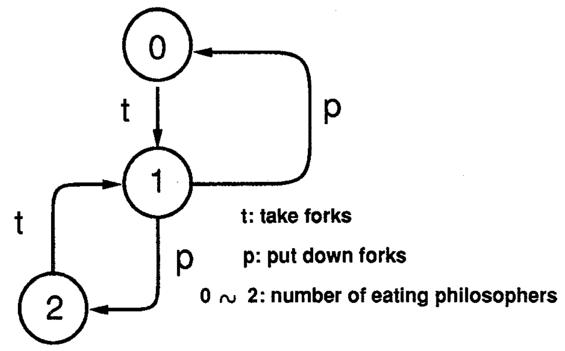


Figure 5: Minimum NFSM for the philosopher system.

it is needed to calculate the Cartesian product of the NFSMs in the CNFSM. In addition, the NFSM has some equivalent states, since all the philosophers are equivalent. All state with the same number of eating philosophers are equivalent states. These unreachable states and equivalent states can be deleted. This minimum NFSM is shown in figure 5. We can get test suites by applying existing methods based on FSM to the minimum NFSM. The typical existing method, the Wp method is surveyed in 4.2.2 [1] [6].

Existing methods have the following problems. It is needed to construct an integrated NFSM from the CNFSM by calculating the Cartesian product of the NFSMs in the CNFSM, since test suites can not be directly generated from a CNFSM. The main problem of existing methods is that a huge amount of computation costs is required for the reachability analysis. Even if the number of states of an NFSM (which is a member of a CNFSM) is very small, the integrated NFSM would have many unreachable states. For example, if one CNFSM consists of 10 NFSMs with 2 states, the integrated NFSM has 2^{10} states, but most of them are unreachable. In general, if there are n NFSMs with k states, the integrated NFSM have n^k states. Reduction operations such as reachability analysis should be performed on the n^k states, and thus a large amount of computation costs is required. figure 3, figure 4 and figure 5 show CNFSM , its NFSM, and its minimum NFSM of the "five dining philosophers" respectively.

3 Coloured Petri Nets

The CPT method and the CPG method are based on Coloured Petri Nets, and reachability trees reduced by the equivalent marking technique are used to achieve the practical test suite length. This tree is called CP-tree[14]. In this section, the Coloured Petri nets which we use in this paper was defined first, and then the CP-tree is described.

Definition 1:

Coloured Petri Nets is a 6-tuple

$$CPN = (P, T, C, A, I, M_0)$$

where,

1. P is a finite set of places.
2. T is a set finite of transitions.
3. P ∩ T = ϕ and P ∪ T ≠ ϕ (ϕ stands for empty set)
4. C is the colour-function defined from P ∪ T into non-empty sets. It attaches to each place a set of possible token-colours and to each transition a set of possible occurrence-colours.
5. A ⊆ (P × T) ∪ (T × P) is a finite set of arcs.
6. I is a finite set of inhibitor arcs.
7. The initial marking M_0 is a function defined on P, such that $M_0(p) \in C(p)_{MS}$ for all $p \in P$, where $_{MS}$ denotes multi-set which can contain multiple occurrences of the same element.

3.1 CP-tree

The CP-tree is a reachability tree of a Coloured Petri Nets. It was proposed by K.Jensen[15]. The detailed illustrations are shown in [14]. One of the distinguished advantages of the CP-tree is that the reachability trees can be reduced by equivalent markings. K.Jensen illustrates the CP-tree using "five dining philosophers" as follows.

The philosopher system can be described by the following Coloured Petri Nets.

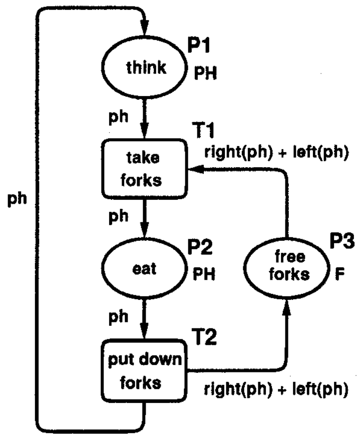


Figure 6: Coloured Petri Nets for the philosopher system.

Here, following markings are analyzed.

$M_1 = (ph_2 + ph_3 + ph_4 + ph_5, ph_1, f_3 + f_4 + f_5)$

$M_2 = (ph_1 + ph_3 + ph_4 + ph_5, ph_2, f_1 + f_4 + f_5)$

$M_3 = (ph_2 + ph_4 + ph_5, ph_1 + ph_3, f_5)$

We want M_1 and M_2 to be equivalent. The point is that we do not need to identify eating philosophers, because all philosophers "behave in the same way". In [14], the **equivalence** and the CP-tree are defined in Definition 2 and Definition 3, respectively.

Definition 2:

Two ω-markings(which is defined by [17]) M_1 and M_2, of a Coloured Petri Net are equivalent, which is written as $M_1 \approx M_2$, iff there exists a symmetry(which is defined by [14]) $\varphi \in \Phi$ such that $M_1 = \varphi(M_2)$. Φ is a set of symmetries. The ω-marking of node x is denoted by M_x . It is easy to verify that $\approx$ is an equivalence relation.

Definition 3:

A reachability tree, CP-tree, for a Coloured Petri Net with an equivalence relation $\approx$ is the full reachability tree reduced with respect to covering markings and equivalent markings.

1. If a node y strictly covers one of its predecessors z then we assign $M_y(p)(c) := \omega$ for all $p \in P$ and all $c \in C(p)$ satisfying $M_y(p)(c) > M_z(p)(c)$.
2. Only one node in each (reachable) equivalence class of $\approx$ is developed further. Only one node in a set of equivalent brothers is included in the tree. The other nodes are removed, but the arc to the included brother node contains information about whether they exist or not in the tree.
3. Associated to each node is an ω-marking and a node-label. The node-label is a (possibly empty) sequence of status information, which may indicate that the marking is equivalent to the marking of an earlier processed node, covering the marking of a predecessor node, or dead.
4. Associated to each arc from node n_1 to n_2 is an arc-label which is a list of occurrence information. Each element is a pair (t, c) where $t \in T$ and $c \in C(t)$. Each pair in the list is enabled in the marking which are equivalent to the marking of n_2.

The philosopher system (figure 6) can be described by the CP-tree shown in figure 7. It is reduced by covering markings (none in this tree) and by equivalent markings.

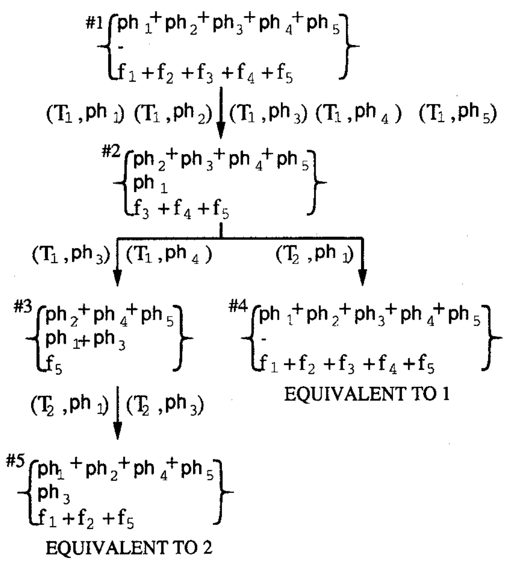


Figure 7: CP-tree for the philosopher system. It is reduced by covering markings (none in this tree) and by equivalent markings.

The each node of figure 7 represents a marking of CPN, i.e. The each columns of the node consists of thinking philosopher's states, eating philosopher's states, and unused fork's states. The label on the right side of each node, such as "#1", is a node-label.

The arc of the tree represents a fire transition. The arc-label, "(T_1, ph_1)", represents a fire transition T_1 with token ph_1. The "$(T_1, ph_3)(T_1, ph_4)\cdots$" represent equivalent fire transitions. Only one node of each class of equivalent marking is developed further. Only one node in a set of equivalent brothers is included in the tree. The other nodes are removed, but the arc to the included brother node contains information of their existence. The number used in the notation "EQUIVALENT TO 2" represents the node-label of the equivalent node.

4 Test Suite Generation Methods based on Coloured Petri Nets

The problem of the amount of computation costs can be solved by using Petri net. The reason is as follows: Since a Petri net is an FDT which models concurrent systems, constraints of concurrent systems are involved in the specification. Thus, through the generation of a reachability tree using tokens, no unreachable states are generated.

However, when modeling a large system in a traditional Petri net, many equivalent subnets are often involved, and thus a large effort is required to specify the system. To save these efforts, CPN has been proposed [15]. Since a CPN allows tokens to have attributes, hierarchical design is possible.

When CPN is used, reduction by the equivalent markings is used in the test suites generation so as to obtain a test suite with reasonable length. The essential nature of the equivalent markings fits the basic idea of the test case generation method "equivalence partitioning" [20] well. The idea is that if one test case of an equivalence class detect errors, another test case of the class will detect the same errors.

In this section, we propose the CPT(Coloured Petri net Tree) method and the CPG(Coloured Petri net Graph) method.

4.1 Test suites generation method based on CP-tree (CP-Tree method)

The test suites generation process of CP-Tree Method (CPT method) is simpler than existing methods. This method is not powerful enough to detect transfer errors, and uses many reset messages. However, the detection of these errors are not always required. For example, if a state message is associated to every state, the detection of transfer errors is not needed. Because of its simplicity, this method is suitable for large systems to which the CPG method can not be applied. Test suites generated by the CPG method is longer than that of the CPT method and the required amount of the computation is larger.

The test suites generation procedure of the CPT method is as follows. First, a specification of the target system described by a Coloured Petri Net is provided. Then, a CP-tree is constructed from this specification, and a test suite is generated from the CP-tree. A test suite is a set of input sequences and correct output sequences. The input sequences and the output sequences are generated from traces of arcs, and traces of nodes from the root to leaves of the CP-tree respectively. In the CPT method, a test suite is generated as follows.

Generation of a test suite:

1. N is a finite set of nodes of a CP-tree.

2. B is a finite set of arcs of the CP-tree.

3. When there are $k+1$ leaf nodes $l_0 \cdots l_k$ on a CP-tree, there are traces from the root node to every leaf node. If there are m nodes on a path to leaf l_i ($l_i = n_{mi}$), the finite sequence of nodes and arcs on a trace can be defined as:

$$X_i = n_r.b_{0i}.n_{0i}.b_{1i}.n_{1i}.\cdots.b_{mi}.n_{mi}$$

where n_r is the root node, $n_{ji} \in N$, $b_{ji} \in B$, and "." is a concatenation.

Then the test suite $\prod$ is:

$$\prod = X_0.R.X_1.R\cdots.X_k$$

4.2 Test suites generation method based on CP-graph (CPG method)

While the CPG method is more complicated and generates longer test suite than CPT method, the CPG is much simpler than existing methods. The main advantage of this method is that existing methods based on FSM(Finite State Machine) can be applied to the graph. Therefore transfer errors can be detected and reset messages can be reduced. CPG method generate test suites in the following procedure.

1. The specification of the target system is provided as a Coloured Petri Net.

2. A CP-graph is constructed from the specification.

3. Regarding this graph as an FSM, test suites are generated by applying an existing method based on FSM.

4.2.1 CP-graph

CP-graph is generated by the following procedure.

1. Construct a CP-tree.

2. Delete all of equivalent marking node, and connect the arc which has originated from the deleted node to the equivalent node.

A CP-graph of the "five dining philosophers" (figure 6) is shown in figure 8. The CP-graph is defined as follows.

Definition 4:

CP-graph is a 2-tuple CP-graph = (N, B), where

1. N is a finite set of nodes of a CP-graph, and n_j corresponds to a marking of CPN,

   $N = \{n_0, n_1, n_2, \cdots, n_k\}$.

2. B is a finite set of arcs of the CP-graph, and b_j corresponds to a transition of CPN,

   $B = \{b_0, b_1, b_2, \cdots, b_l\}$.

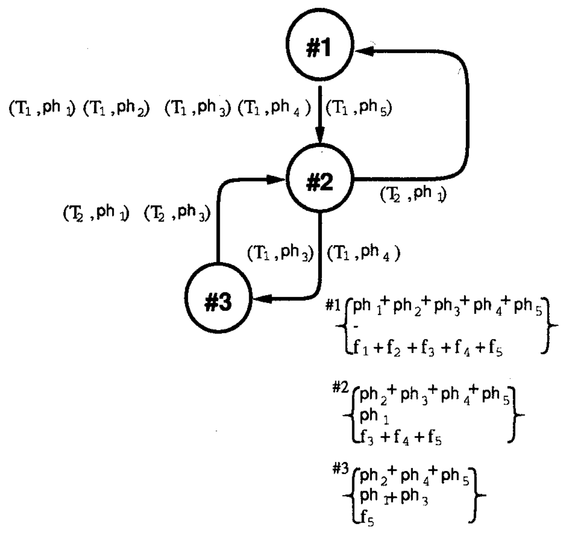

#1
(T1 ,ph 1) (T1 ,ph2) (T1 ,ph3) (T1 ,ph4) (T1 ,ph5)
#2
(T2 ,ph 1)
(T2 ,ph 1) (T2 ,ph3)
(T1 ,ph3) (T1 ,ph4)
#3
#1 ph1+ph2+ph3+ph4+ph5 - f1+f2+f3+f4+f5
#2 ph2+ph3+ph4+ph5 ph1 f3+f4+f5
#3 ph2+ph4+ph5 ph1+ph3 f5

Figure 8: CP-graph for the philosopher system.

4.2.2 Test suite generation based on CPG method

States of a Petri net are defined by its markings[17]. Hence, a node of a CP-graph represents a state of the CPN, and CP-graph can be regarded as an NFSM. Therefore, we can get test suites by applying existing methods based on FSM (provided that the CP-graph satisfies prerequisite conditions of the existing method). Here, the procedure of applying the Wp-method [6][1] is shown. First, it is shown that the CP-graph can be regarded as an NFSM so as to apply the Wp-method.

1. $n_j \in N$ corresponds to a state of a state machine St, which represents the system.

   Thus, $St = N$.

   $$St = \{S_0, S_1, S_2, \cdots, S_n\}.$$

   n is the number of states or nodes of St or N.

2. Initial State: S_0 or n_0 is the initial marking.

3. A set of the pairs of arcs and their pointing nodes is a behavior function h. h can be defined as

   $$h : S_t \times L_i \rightarrow powerset(S_t \times L_o)\ \{\phi\}$$

   where L_i is a set of inputs, L_o is a set of outputs, and ϕ is the empty set.

4. The finite set of correct outputs L_o is a set of states s_j, and a set of states is a set of nodes. Thus we have,

   $$L_o = N.$$

5. Therefore, above equations get following result: CP-graph can be represented a 5-tuple

   $$(St, L_i, L_o, h, S_0). \qquad (1)$$

   This equation can be regarded as an NFSM. Therefore, we can apply an existing method based on FSM to the NFSM.

Now, the Wp-method is applied to the CP-graph according to the descriptions in [6][1] as follows.

Input

A specification CP-graph (1) in the form of a minimal NFSM with n states, and the upper bound $m (n \leq m)$ on the number of states in the prime machine of the given NFSM implementation.

Notations

L	$L_i \times L_o$, a set of input/output pairs; u denotes such a pair.
ε	ε is the empty sequence.
L^*	set of sequences over L; x denotes such a sequence. Note that $\varepsilon \in L^*$.
$P = \varepsilon \Rightarrow Q$	$P = Q$.
$P = x \Rightarrow Q$	$\exists P_1, \cdots, P_{k-1} \in St$ $(P = P_0 - u_1 \rightarrow P_1 \cdots - u_k \rightarrow P_k = Q)$ where $u_1, \cdots, u_k \in L$, and $x = u_1 \cdots u_k$.
$P = x \Rightarrow$	$\exists Q \in St(P = x \Rightarrow Q)$.

tr(P)	$tr(P) = \{x \mid P = x \Rightarrow\}$ (note that $tr(P) = \{x \mid P = x \Rightarrow Q$ for some $Q \in St\}$).
x^{in}	For $x \in L^*, x^{in}$ is an input sequence obtained by deleting all outputs in x (note that $x^{in} \in L_i^*$).
V^{in}	For V $\subseteq L^*, V^{in} = \{x^{in} \mid x \in V\}$.

Output

a test suite $\prod$.

step 1

Construct a characterization set W, and tuple of state identification sets

$$\{W_0, W_1, \cdots, W_{n-1}\}.$$

step 2

Construct a (preferably minimal) set $Q \subseteq L_i^*$ such that:

$$\forall S_i \in St \quad \exists x \in L^* \ \ (x^{in} \in Q \ \& \ S_0 = x \quad \Rightarrow S_i).$$

step 3

Construct two sets P and R such that:

$$P = Q.(\{\varepsilon\} \cup L_i)$$

and

$$R = P \setminus Q.$$

step 4

First, define operator $\oplus$ as follows: for $V \subseteq L_i^*$,

$V \oplus \{W_0, W_1, \cdots, W_{n-1}\} = \bigcup_{So=x \Rightarrow Si \& x^{in} \in V} \{x^{in}\}.Wi$

Then, construct a test suite $\prod$ in the following manner:

$$\prod = \prod 1 \cup \prod 2$$

where

$$\prod 1 = Q.(\{\varepsilon\} \cup Li \cup Li^2 \cup \cdots \cup Li^{m-n}).W,$$

$$\prod 2 = R.Li^{m-n} \oplus \{W_0, W_1, \ldots, W_{n-1}\}.$$

5 Test suites Generator (TSG)

We designed and implemented an experimental Test Suites Generator(TSG) that is able to generate test suites based on CPT method. The TSG consists of an input module, a reachability analysis module, and an output module. The input module has two modes of inputs, a GUI(Graphic User Interface) mode and a text mode. Examples of these modes are shown in figure 9 and figure 10 respectively. Before passing inputs data to the reachability analysis module, the GUI mode outputs data of the same format as the text mode. In the GUI mode, users can input CPN items such as places, transitions, arcs, and tokens, using "place_win" window, "transition_win" window, "arc_win" window, and "token_win" window.

In the GUI mode, the relation between CPN items can be described in the "TSG" window. In the text mode, the relation between places, transitions, arcs are specified by the reserved words "place", "transition", and "arc", respectively. For example, an arc definition is as follows.

```
arc[21] = {
          label = Arc_X;
          place =3;
          transition =8;
          arctype =1;
}
```

The reachability analysis module generates CP-tree based on the information from the input module. The tokens that have the same "Color" are regarded as equivalent.

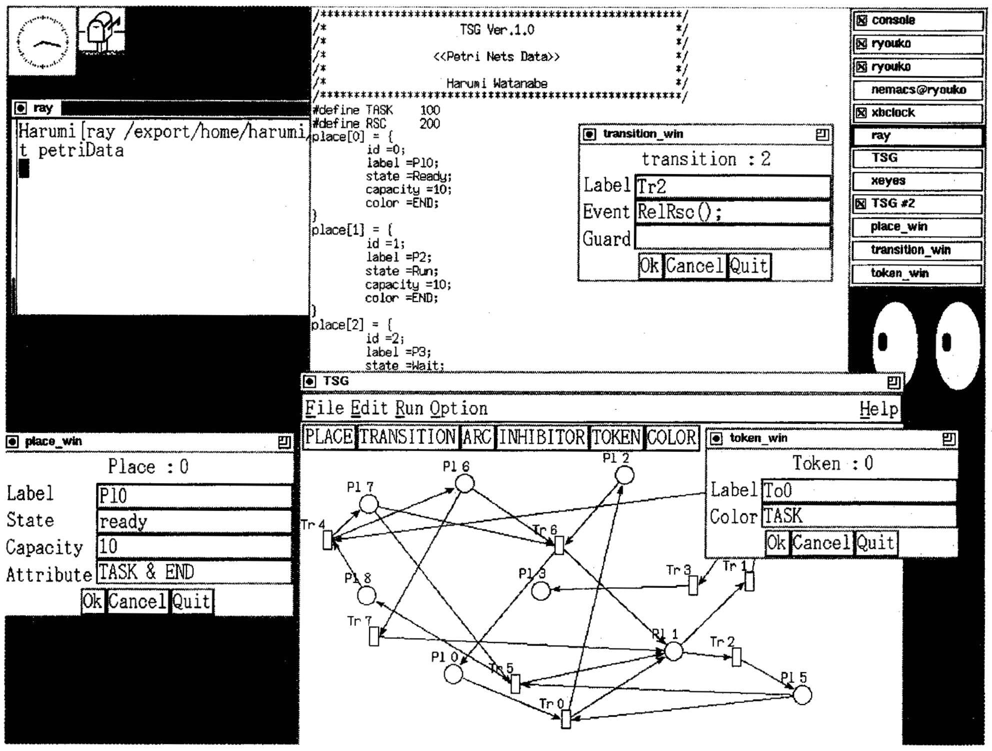
An example of outputs of the output module is shown in figure 11. The outputs represents test suites. In addition, the output module outputs additional data: the number of CP-tree nodes, the number of CP-tree leaves, and correct IUT(Implementation Under Test) outputs. The correct outputs are generated by "State" in "place_win" window.

```
task0()             task1()
{                   {
    GetRsc();           RelRsc();
    GetRsc();       }
    GetRsc();
}
```

Figure 11: An output of TSG.

6 Evaluations of the test suite length

In this section, the effects of test suite length reduction by equivalent markings are evaluated. The reduction by the equivalent markings is the main feature of the CPT and the CPG methods.


```
Harumi [ray /export/home/harumi/
t petriData
```

```
/*****************************************************/
/*                    TSG Ver.1.0                    */
/*                                                   */
/*                <<Petri Nets Data>>                */
/*                                                   */
/*                  Harumi Watanabe                  */
/*****************************************************/
#define TASK    100
#define RSC     200
place[0] = {
        id =0;
        label =P10;
        state =Ready;
        capacity =10;
        color =END;
}
place[1] = {
        id =1;
        label =P2;
        state =Run;
        capacity =10;
        color =END;
}
place[2] = {
        id =2;
        label =P3;
        state =Wait;
```

transition : 2

Label	Tr2
Event	RelRsc();
Guard	

Place : 0

Label	P10
State	ready
Capacity	10
Attribute	TASK & END

Token : 0

Label	To0
Color	TASK

Figure 9: TSG inputs in the GUI mode.

The CPN used for the evaluation has the following properties.

1. The number of places ······ 9
2. The number of transitions ······ 8
3. The number of arcs ······ 26

The CPN is shown in figure 12. This figure shows output from TSG. The length of the test suites generated by the CPT method is shown in figure 13. This result shows we can get a significant test suite length reduction when the number of equivalent concurrent state is large.

7 Discussions

1. The amount of computation costs

   Existing methods require larger amount of computation costs than the CPT method and the CPG method. Since an integrated NFSM is generated from the Cartesian product of NFSMs belongs to the original CNFSM, the integrated NFSM may have many unreachable states. Therefore, a large amount of computation costs for reachability analysis is needed.

   The CPT method and the CPG method generate test suites from the CP-tree, and the CP-graph, respectively. Since the CP-tree and the CP-graph are generated through reachability analysis of CPN, there are no unreachable states. Therefore these methods requires smaller amount of computation costs than existing methods.

   For example, in the "five dining philosophers" problem, if an integrated NFSM is generated from the CNFSM by existing methods, the integrated NFSM has 2^5 states. While the elimination of the many unreachable and equivalent states included in the NFSM results in the smallest NFSM with only 3 states, the minimization NFSM process requires a large amount of computation costs. On the other hand, the CPG method directly generates a CP-graph with 3 states. The reachability analysis and the states reduction by the equivalent markings are performed at the same time.

2. The error detection coverage

```
/**************************/
/*      TSG Ver.1.0       */
/*  <<Petri Nets Data>>   */
/**************************/
#define TASK        10
#define RSC         100
place[0] = {
        id =0;
        label =P10;
        state =run;
        capacity =1;
        attribute =END;
}

transition[1] = {
        id =1;
        label =Tr2;
        event ="GetRsc();";
        label =;
}
token[3] = {
        id =0;
        label =task0;
        color =TASK;
        init =0;
}
```

Figure 10: TSG inputs in the text mode

CPT method cannot fully detect transfer errors. Since this method is essentially equivalent to the Transition Tour method(TT method), it has the same coverage as the TT method[10].

Since existing methods based on the FSM can be applied for the detection of transfer errors, the coverage of the CPG method follows that of the existing FSM based method.

3. The length of test suites

   A test suite is constructed from set, reset and status messages. While a test suite generated by the CPT method includes smaller number of set messages, reset messages and status ones are included. On the other hand, the length of a test suite generated by the CPG method depends on the applied existing FSM based methods. Some existing methods effectively reduce the number of the reset messages and the status ones.

8 Conclusion

In this report, we proposed the CPT method and the CPG method for conformance test suite generation, and presented TSG that can generate test suites based on the CPT method. These methods have following advantages: the required amount of computation is smaller than existing methods based on the FSM, and the length of the test suites can be reduced by equivalent markings. In addition, the CPG method has the same coverage as existing methods based on the FSM. The advantages of the CPT and the CPG methods are shown through the evaluation of the test suite length reduction by equivalent markings. The reduction is very effective when the number of equivalent concurrent states is large. These methods make it possible to test interaction parameters of communication protocols sufficiently.

While the significant advantage of proposed two methods is their ability to deal with concurrency so far, we are currently working on the use of Time Petri nets[21] and Stochastic Petri nets[22] to handle time dependent or stochastic systems.

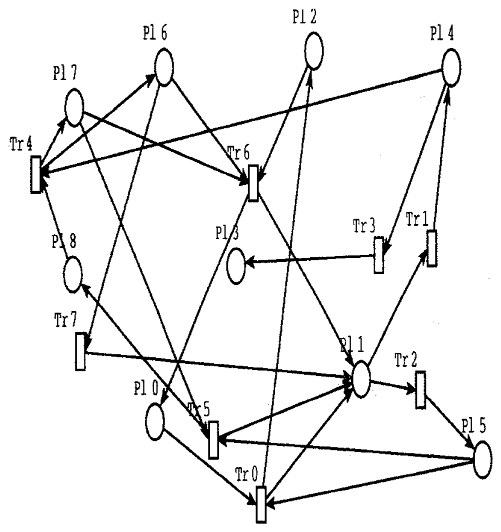


Figure 12: An example of CPN.

Acknowledgement

We would like to thank Prof. Motoshi Saeki at Department of Computer Science, Tokyo Institute of Technology for his valuable advices.

References

[1] G.Luo, G,v,Bochman, A.Petrenko, "Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized Wp-Method", IEEE Transactions on Software Engineering VOL.20,No.2,February 1994

[2] G.Luo, A.Das, G,v,Bochman "Software Testing Based on SDL Specifications with Save", IEEE Transactions on Software Engineering VOL.20, No.1, pp.72-87, January 1994

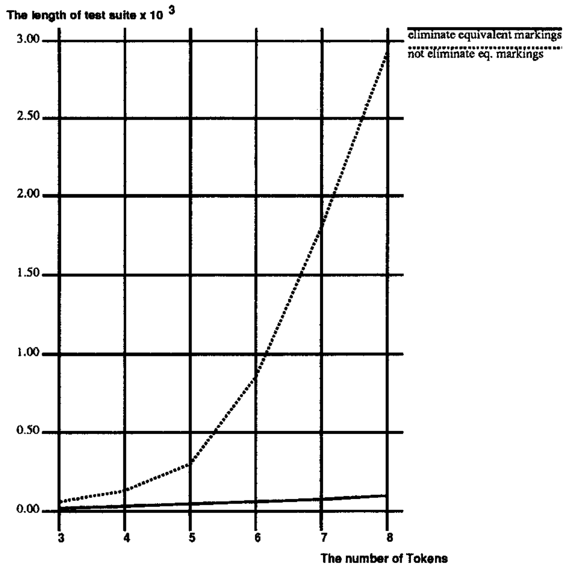


Figure 13: The length of test suite

[3] G.Luo, A.Petrenko, G,v,Bochman "Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines", IFIP IWPTS 7th Int. Protocol Test Systems, pp91-106, November 1994

[4] T.Higashino, G.v.Bochmann, "Automatic Analysis and Test Case Derivation for a Restricted Class of LOTOS Expressions with Data Parameters" IEEE Transactions on Software Engineering VOL.20, No.1, pp.29-42, January 1994

[5] K.Katsuyama, F.Sato, T.Nakakawaji,T.Mizuno, "Strategic Testing Environment with Formal Description Techniquess", IEEE Transaction on Computerss, Vol.40, No.4, pp.514-525, April 1991

[6] S.Fujiwara, G.v.Bochmann, F.Khendek, M.Amalou, A.Ghedamsi, "Test Selection Based on Finite State Models", IEEE Transactions on Software Engineering VOL.17, No.6, pp.591-603, June 1991

[7] D.H.Pitt, D.Freestone, "The Derivation of Conformance Tests from LOTOS Specification", IEEE Transactions on Software Engineering VOL.16, No.6, pp.1337-1343, December 1990

[8] G.v.Bochmann, R.Dssouli, J.R.Zhao "Trace Analysis for Conformance and Arbitration Testing", IEEE Transactions on Software Engineering VOL.15, No.11, pp.1347-1356, November 1989

[9] D.P.Sidhu, T.Leung "Formal Methods for Protocol Testing: A Detailed Study", IEEE Transactions on Software Engineering VOL.15, No.4, pp.413-426, April 1989

[10] S.Naito, M.Tsunoyama, "Fault Detection for Sequential Machines by Transition-Tour", Proceedings of IEEE Computing Conference,pp.238-243, 1981

[11] T.S.Chow, "Testing Software Design Modeled by Finite-State Machines", IEEE Transactions on Software Engineering VOL.4, No.3, pp.178-187, March 1978

[12] G.Gonenc, "A method for the design of fault-detection experiment", IEEE Transactions on COMPUTERS VOL.C-19, pp.551-558, June 1970

[13] A.Gill,"Introduction to the Theory of Finite-State Machines", NewYoork:McGraw-Hill,1962

[14] K.Jensen, "COLOURED PETRI NETS", Lecture Notes in Computer Science, No254, Springer-Verlag, pp.207-247, 1987

[15] K.Jensen, "Coloured Petri nets and the invariant-method", Theoretical Computer Science 14, pp.317-336, 1981

[16] M.Notomi, T.Murata, "Hierarchical Reachability Graph of Bounded Petri Nets for Concurrent-Software Analysis", IEEE Transactions on Software Engineering VOL20, No. 5, pp.325-336, May 1994

[17] James L. Peterson, "Petri Net Theory and the Modeling of Systems", Englewood Cliffs, New Jersey, Prentice Hall Inc., 1981,

[18] G.J.Hoizmann, "Design and Validation of Computer Protocols", Englewood Cliffs, NJ: Prentice-Hall, 1991

[19] K.D.Shere, R.A.Carlson "A Methodology for Design, Test and Evaluation of Real-Time Systems" IEEE COMPUTER, pp.35-pp.48, Feburary 1994

[20] G.J.Myers, "The Art of Software Testing", John Wiiley & Sons, Inc., 1979

[21] P.M.Merlin, "A Methodology for the Design and Implementation of Communication Protocols", IEEE Transactions on COMMUNICATIONS, VOL.COM-24, No.6, pp.614-621, June, 1976

[22] M.A.Marsan, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", ACM Transactions Computer Systems, VOL.2, No.2, pp.93-122,1984