"계층 상태공간 축약방법"에 기반한 효율적인 상호운용성 시험 방법론

최 영 한[†] · 진 병 문[†] · 이 동 익[††] · 진 성 일[†††]

요 약

상호운용성은 정보기술 및 통신분야에서 가장 중요한 요소의 하나이다. 본 논문에서는 상호운용성에 관한 시험방법 및 시험 스위트의 생산에 대하여 논의하고 있다. 적합성시험을 비롯하여 정형적인 검증에서 가장 먼저 고려되는 치명적인 문제점은 상태공간의 폭발에 관한 사항이며 이는 상호운용성의 시험방법 및 시험 스위트의 개발에 있어서도 가장 먼저 해결되어야 할 문제이다. 본 논문에서는 페트리네트를 이용한 상호운용성 시험을 지원하기 위해 새로운 상태 공간 축약 방법을 제시하며 이를 이용한 상호운용성의 시험방법 및 시험 스위트 생성 방안을 IOSM, Quasi stable state 등을 이용하여 HOSS에 기반한 상태축약 결과를 보임으로써 상호운용성시험을 효율적으로 지원하는 방안을 제시한다.

An Efficient Interoperability Test methodology Based on Hierarchically Organized State Space

Young-Han Choe[†] · Byoung-Moon Chin[†] · Dong-Ik Lee[††] · Seong-Il Jin[†††]

ABSTRACT

Interoperability(IOP) is one of the major goal of Information Technology and Telecommunication fields. In this paper, we discuss developing an interoperability testing(IOPT) method. As is easily guessed from conformance testing and formal verification, state space explosion problem is the most serious problem we encounter in deriving interoperability test method and its test suite. A new state space reduction method to support interoperability testing is suggested based on Petri nets. The proposed test method can be applied to 1 to many communication protocols as well as 1 to 1 communication protocols efficiently

1. Introduction

As software systems are getting larger and more complicated, testing has become a very important and expensive part of the software development cycle. Lots of work have been done to reduce the testing costs, especially in the area of communicating software. Testing of communication systems consists of three parts [1]: (1) 'conformance testing' for checking whether each protocol entity conforms to its functio-nal specification, (2) interoperability testing for checking system behavior without concern for each protocol entity, and (3) 'acceptance testing' which checks performance and durability. Considerable resource has been spent on confor

† 정 회 원 : 한국전자통신연구원 표준연구센터
†† 정 회 원 : 광주과학기술원 정보통신공학과
††† 정 회 원 : 충남대학교 컴퓨터과학과
논문접수 : 1997년 9월 25일, 심사완료 : 1998년 5월 26일

mance test or CT for communication systems, and many researchers have spent much effort to produce conformance abstract test suites. For interoperability testing, in short IOPT, we must check whether more than one implementation together behave as expected. Even though a set of implementations passed CT, among them interoperability is not guaranteed. Thus to ensure IOP, IOPT has to be performed after CT.

1.1 related work

There has been only a few work in the area of abstract IOP test case generation [8][1][7][3][4]. All these works are based on reachability analysis. [8][7] introduced both upper and lower testers. On the contrary others adopted only upper tester. [1][4] introduced the notion of stable states to avoid state space explosion problem. In [1], stable states are defined as a system state (1) that is reachable from the initial state adhering to the single simulating principle, that is only one stimulation from environment must be given at each stable state, and (2) from which no change can occur without an other stimulation. However, as will be discussed, this definition of stable states is insufficient in the IOPT. In [4], IOP is defined as [4] is worthy noting that they proposed various optimization method of IOP test derivation in the context of symmetric protocols and hence it works well with symmetric protocols.

Previous work listed above, however, are devoted into testing for only one to one communication protocols. Assume that there are n implementations which should interwork. Then we have to perform IOPT n(n-1)/2 times to complete IOPT, if we use above IOPT methodologies. Furthermore if our objective protocols are designed for 1 to many communication such as teleconferencing systems, there is no way to perform IOPT.

On the other hand, as shortly discussed above, when formal verification and testing of concurrent systems are considered, we always suffer from state space explosion problem. Furthermore, since many protocol entities are associated in IOPT, appropriate state space reduction methods have to be developed. In the area of formal verification, many state space reduction methods have been proposed, and some of them are proved useful in the protocol design [2][5][6][9][10][11]. However, most of them are only useful to check liveness and/or safeness properties, since they optimize state space according to some specific properties by pruning unnecessary states and transitions based on trace theory. On the contrary, HOSS (Hierarchically Organized State Space) proposed in [5] was successfully applied to static analysis of Ada programs. The basic idea of HOSS is that a given Petri net, they say Ada net, is regarded as a collection of subnets interacting with each other only through transitions corresponding to events. HOSS starts with generating state space of each subnet, then compress the state space under a certain equivalent relation and combine them to obtain higher level state space. This idea can be applied to IOPT, since IOPT is a functional testing a system which can be regarded as collection of some subsystems. However, it is not directly applied since only interactions via transitions are defined in Ada net, while there is buffer between interacting communicating entities.

In this paper, we suggest a framework for an efficient IOPT to support 1 to many communication as well as 1 to 1 communication. Our approach can be regarded as generalization of [4] in the sense of architecture, which derives test cases from stable state to stable state in IOSM. Further by merging notions of modified HOSS and quasi-stable states, IOPT can be done more efficiently.

1.2. overview of our approach

Figure 1 shows the proposed test architecture with 3 implementations under test, denoted as IUT. The test architecture is considered as natural generalization of proposed in [4].

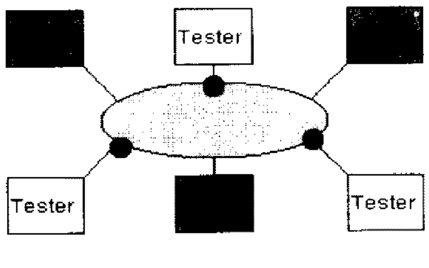
In our approach, we assume that: (1) each implementation to be tested is modeled by an FSM and (2) each implementation has passed CT.

In general, testing is done based on stimulating/observing principle, namely testing consists of observing states resulting from a stimulation of its environment, where environment refers to outside the system. From the assumption (2), we only need to observe external events of each entity or implementation, where an external event refers to stimulation from the environment of an implementation and/or a system currently considered.

To have reduced state space during preserving useful information for IOPT, i.e., information about external states or events, we generate state space for two entities based on Petri nets. Then, in the generated state space, a set of equivalent states in the sense of external events is mapped to a state. This combine and compress operations are repeated till global state space of total system is obtained.

Based on this global state space, test suite is derived based on methods which have been used in CT.

1.3 organization of the paper



(Fig.1) Test architecture for IOPT

In Section 2, some terminology and concepts which are useful to follow the paper is explained. In Section 3, main topic of the paper, state space reduction for efficient IOPT will be introduced. A simple example is shown in Section 4. The final Section 5 is the conclusions and future work.

2. Notations and definitions

In this paper, we assume that specifications and implementations of entities are modeled by FSM called IOSM. As an intermediate model for IOP test suite we adopt Petri nets to model communication or interoperation between FSMs, then reachability analysis will be performed based on it.

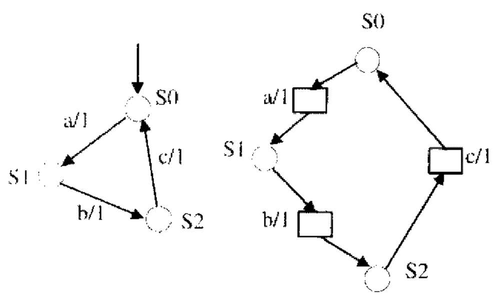
Definition 1. (IOSM) [4]
An *IOSM* is a 5-tuple (S, I, O, δ, s_0). Where
S is the set of states,
I is the input alphabet,
O is the output alphabet,
$\delta \subseteq \{s\text{-}v/U\text{->}s' \mid s, s' \in S \wedge v \in I \wedge U \in O^*\}$ is a state transition function, and
s_0 is the initial state.

The set of events which results in state transitions, in an entity or an IOSM, are classified as shown in [AS 91]:

Trigger event : signal reception from the environment of the system, such as from user.
Receive event : signal reception from another entity.
Send event : signal transmission to another entity.
Internal event : event which does not directly affect to nor is not directly affected from outside of the entity.

Events which are not internal events, i.e.

trigger, receive and send events, are called *external events*.



(a) FSM (b) Corresponding PN

(Fig. 2) FSM and Corresponding PN

Definition 2. (Petri Net)

A *Petri Net* PN is a bipartite directed graph with two types of nodes connected by directed arcs and formally specified by a 4-tuple $(P,T;F,M_0)$.

P is the set of places,

T is the set of transitions,

$F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs, and

M_0 is the initial marking.

A *marking* is a function $M_0 : P \rightarrow Z+$ which assigns a number of tokens to each place, where Z+ denotes the set of non-negative integers. A marking implies that a state of the system modeled by a Petri net. M(p) denotes the number of tokens in the place p. In a graph representation a place is represented by a circle, a transition by a box and tokens by black dots.

Note here that an FSM is considered as a structurally restricted class of Petri nets where each transition has exactly one input and one output place, by regarding a state and a state transition in an FSM as a place and a transition in PN (see Fig. 2). The initial state is represented by putting a token on the place corresponding to the initial state.

Definition 3. (firing rule and firing sequence)

A transition t is *enabled* under a marking M if each input place of t has at least one token, i.e., $\forall p \in \bullet t : M(p)>0$, where $\bullet t(t\bullet)$ denotes the set of input(output) place, respectively. By the firing of a transition t we obtain a new marking M′, denoted by M[t>M′, such that

$$\begin{aligned} \forall p \in P : M'(p) &= M(p) \ -1, \text{ if } p \in \bullet t \wedge p \notin t\bullet \\ &= M(p) \ +1, \text{ if } p \in t\bullet \wedge p \notin \bullet t \\ &= M(p) \ \text{ otherwise.} \end{aligned}$$

A sequence of transitions is a *firing sequence* from M_0, if there exists a sequence of markings $M_1M_2...M_{n+1}$ such that $M_i[t_{i+1}>M_{i+1}$, i = 1, ..., n. M is said *reachable* from M_0, denoted by $M0[\sigma>M$, if there exists a firing sequence $=t_1t_2...t_{n+1}$ from M_0 to M. Let T′ be a subset of T. $\sigma|T$ denotes the *restriction* of over $T' \subseteq T$, that is a firing sequence obtained by deleting all the transitions in T−T′ from σ.

Definition 4. (labeled Petri nets)

A *labeled Petri net* N is 5-tuple $(P,T;F,M_0,L)$. Where $(P,T;F,M_0)$ is PN and $L : T \rightarrow I/O^*$ is a labeling function. An IOSM can be considered as a labeled Petri net by simple transformation explained above. In the rest of the paper the term IOSM means an IOSM or its Petri net representation.

Definition 5. (communicating IOSMs) [4]

A *communicating system* Σ of n IOSM's is 4-tuple $(\{(P_i,Q_i)|1 \le i \le n\}, I_\Sigma, O_\Sigma, s_{\Sigma 0})$. P_i, $1 \le i \le n$, is an IOSM, Q_i, $1 \le i \le n$, is an input queue, I_Σ is an external input alphabet, O_Σ is an external output alphabet and $s_{\Sigma 0} = ((s_{10},Q_{10}), ..., (s_{n0},Q_{n0}))$, where s_{i0} and Q_{i0} are the initial state of Mi and an empty queue, respectively.

In [4], a stable state is defined as a system state in which input queues are all empty.

Also they assumed that a sequence of inputs from the environment one by one. Namely the next input from the environment is read only when the system is in a stable states. The limitation of this notion of the stable states can be explained as followings: Consider two IOSM's $IOSM_1$ and $IOSM_2$. $IOSM_1$ is in a stable state s and $IOSM_2$ is in s′ which is not a stable state. Thus an arbitrary input symbol u′ executable at s′ is an internal event and the input symbol u executable at s is an external one, that is an input from its environment. Then in the global state, u and u′ are executable concurrently, that is input sequences uu′ as well as u′u are allowed in the real world even if we assume interleaving semantics. However, only u′u is allowable input sequence in the above definition of stable states. This means that we cant investigate states evolved from s[u〉. Before defining a new concept of stable states, called quasi-stable state, we consider a composition of IOSM's based on Petri nets.

Definition 6. (composition of IOSM's)

Let $N_1=(P_1,T_1;F_1,M_{01})$ and $N_2=(P_2,T_2;F_2,M_{02})$ be Petri net representations of two IOSM's. The *composition of N_1 and N_2*, denoted by $N_1 \times N_2$, forms a Petri net $N=(P,T;F,M_0)$, where

$P = P_1 \cup P_2 \cup Q$ and $P_1 \cap P_2 = \varnothing$, where Q is the set of input queues,

$T=T_1 \cup T_2$ and $T_1 \cap T_2 = \varnothing$,

$F \subseteq (P \times T) \cup (T \times P)$ and

$M_0(p)=1$, if $M_{01}(p)=1 \wedge p \subset P_1$ or $M_{02}(p)=1 \wedge p \in P2$

$=0$, otherwise.

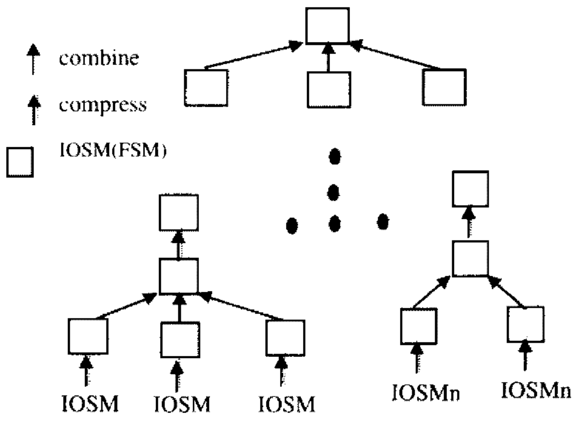
If $t \in T_1$ corresponds to a receiving event, then there exists an event in $t' \in T_2$ corresponding to the receiving event. Add a place $p \in Q$ and arcs from t′ to p and from p to t. This place corresponds to an input queue of N_1. The definition of IOSM composition is extended to n IOSM's composition in a natural way and composition of n IOSM's is denoted by $IOSM_1 \times IOSM_2 \times \ldots IOSM_n$.

Here we define a quasi-stable state as:

Definition 7. (quasi-stable state, qss in short)

A Marking M or a state s in a composed IOSM is a *quasi-stable state*, denoted *qss*, if a transition corresponding to a trigger event is executable under M or s.

By the notion of qss, we can investigate all the possible traces in a whole system.



(Fig. 3) Hierarchy of state space

3. Reduced state space generation for IOPT based on HOSS

The tree structure shown in Fig. 3 represents the state space constructed by composition of IOSM's, where the lowest level nodes are IOSM's. The state space is compressed then combined to produce higher level state space. This compress-and-combine operation is repeated until no more applicable.

To define compress operation, we need to define an equivalent relation over the set of states.

Definition 8. (in-set, out-set, IO-similar) [5]

Let M be an IOSM. Note that the transition function δ can be regarded as the set of labeled arcs if we consider a graph representation of an IOSM. In this definition we use δ as the set of arcs. Let (I/O*) be a subset of (I/O*) and δ' be the maximal subset of δ such that the edge labels of them are in (I/O*).

S_{in} = {s∈S | s is s_0 or an edge in δ' is terminating at s}

S_{out} = {s′∈S | an edge in δ' is outgoing from s′}

For each s in S, in-set and out-set are defined as followings:

in-set of s = {s′∈S_{in} | there is a transition sequence, which does not contain any transition of (I/O*)′, from s′ to s}

out-set of s = {s′∈S_{out} | there is a transition sequence, which does not contain any transition of (I/O*)′, from s to s′}

Two states s_1 and s_2 are said to be *IO-similar* with respect to (I/O*)′ if they have the same in-set and the same out-set. It is trivial that IO-similarity is an equivalent relation. Using this notion of the IO-similar equivalence, states of an IOSM can be divided into equivalence classes. For the set of states in a partition, a state which does not have any successor state is called the corresponding state to the partition.

Compression of an IOSM is performed by mapping the set of states in the same class to one state. By the definition of IO-similarity, if T_{trg} which is the set of trigger events is taken as (I/O*)′, any event between states in the same class is not a trigger event.

Definition 9. (combine)

Let N be a composition of two IOSM′s or two compressed IOSM′s, $IOSM_1$ and $IOSM_2$. And let G be a reachability graph of N. Then G is the *combined* IOSM of $IOSM_1$ and $IOSM_2$.

In IOPT, it is sufficient to consider only external events, from the assumption (2) in Sec. 1.2. The following is an algorithm to obtain a reduced state space.

Algorithm 1: generate reduced state space

```
Input : a set of IOSM's, $IOSM_1$, $IOSM_2$, ..., $IOSM_n$.
Output : $IOSM_{fin}$ which is an IOSM representing global state space.
$IOSM_{init}$ = the initiator IOSM
SIOSM : = the given set of IOSM - ISOMinit.
$T_{ext}$ : = the set of external transitions in the given set of IOSM's.
$T_{trg}$ : = the set of transitions corresponding to trigger events.
Step 1. Take two arbitrary IOSM's, $IOSM_i$ and $IOSM_j$ from SIOSM.
Step 2. $T_{ij}$ : = the set of receiving and sending transitions in $IOSM_i$ or $IOSM_j$.
Step 3. Compress $IOSM_i$ and $IOSM_j$ over $T_{ext}$.
Step 4. $T_{ext}$ : = $T_{ext}$ - $T_{ij}$.
Step 5. Compose $IOSM_i$ and $IOSM_j$, let $N_{ij}$ be the Petri net obtained by composition.
Step 6. Generate the reachability graph $IOSM_{ij}$ of $N_{ij}$, which is also an IOSM.
Step 7. SIOSM : = {SIOSM∪$IOSM_{ij}$} - {$IOSM_i$, $IOSM_j$}.
Step 8. Repeat Step 1 - Step 7 till |SIOSM| =1. Let $IOSM_r$ be the remained IOSM.
Step 9. Compress and combine $IOSM_{init}$ and $IOSM_r$ over $T_{ext}$.
Step 10. Compress the final IOSM over $T_{trg}$.
```

Theorem 1.

Let Ω be $IOSM_1 \times IOSM_2 ... IOSM_n$, $R(\Omega)$ be the set of reachable states of Ω. For any quasi-stable state qss in $R(\Omega)$, there exists a quasi-stable state in $IOSM_{fin}$ corresponding to qss.

[Proof] Though the proposition is almost trivial, we prove it by mathematical induction over the number of given IOSM′s.

(Basis) If only one IOSM is given, $IOSM_{fin}$ is the compression of the IOSM. Since T_{ext} is the set of trigger events, qss has a corresponding state in $IOSM_{fin}$

(Hypothesis) Assume that the proposition is true, if n=k.

(Induction) It is sufficient to consider combining and compression of two IOSM's by the hypothesis. The reachability graph of a combined IOSM preserves corresponding qss. From the Basis, compression of IOSM also preserves the corresponding qss.

Theorem 2

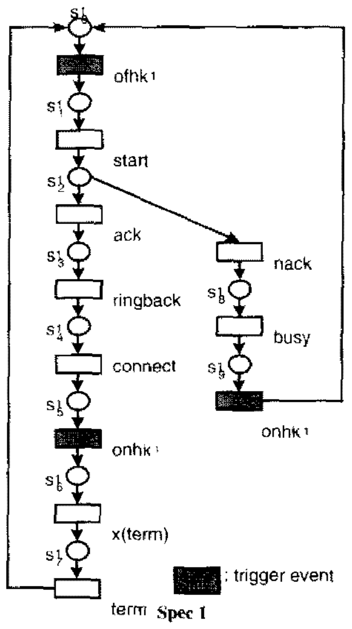
Let Ω be $IOSM_1 \times IOSM_2 \ldots IOSM_n$, and TR($\Omega$) the set of firing sequences for Ω. For $\forall \sigma \in$ TR(Ω), $\exists \sigma' \in$ TR($IOSM_{fin}$) such that $\sigma | T_{trg} = \sigma' | T_{trg}$, where T_{trg} denotes the set of trigger events.

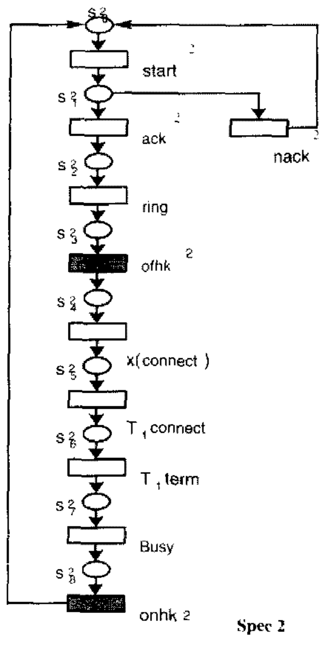
[Proof] Only $\sigma | T_{trg} \subseteq \sigma' | T_{trg}$ part will be proved here. Consider an arbitrary firing sequence σ of Ω, namely σ is a firing sequence in the reachability tree of Ω. From Theorem 1, any quasi-stable state in Ω has its corresponding state in $IOSM_{fin}$. This means that no trigger transitions are deleted by compression.

If n IOSM's are given, at first an arbitrary IOSM is selected as the initiator. Then apply Algorithm one and derive a test suite based on traditional test suite generation methods. Then another IOSM is selected as the initiator and perform above operations. If every IOSM has been selected as an initiator, the derived test cased are a IOP test suite.

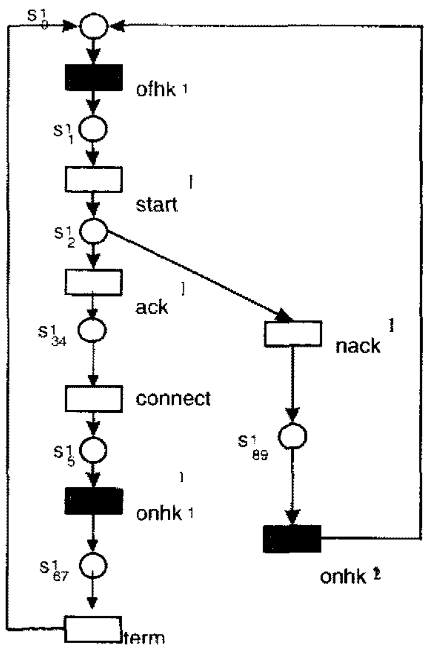
4. A simple example

As an example consider the protocol specification shown in Fig. 4. Black transitions represent trigger events. T^1_{ext} = {ofhk^1, start^1, ack^1, nack^1, connect, onhk$^1_1$, onhk$^1_2$, term}. T^2_{ext} = {start^2, ack^2, nack^2, ofhk^2, T_1connect, T_1term, onhk^2}. The compressed IOSM of Spec1 is obtained as Fig. 5.

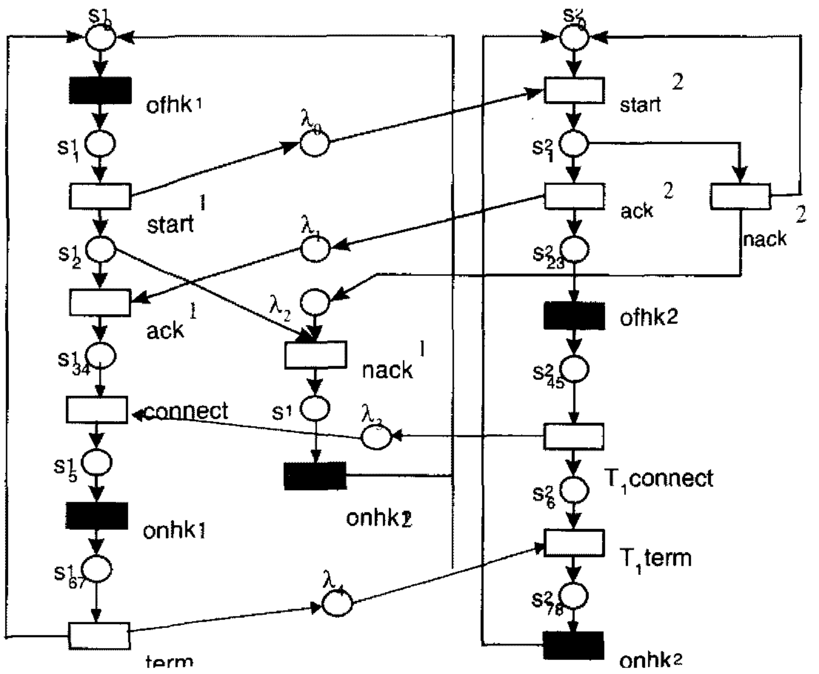




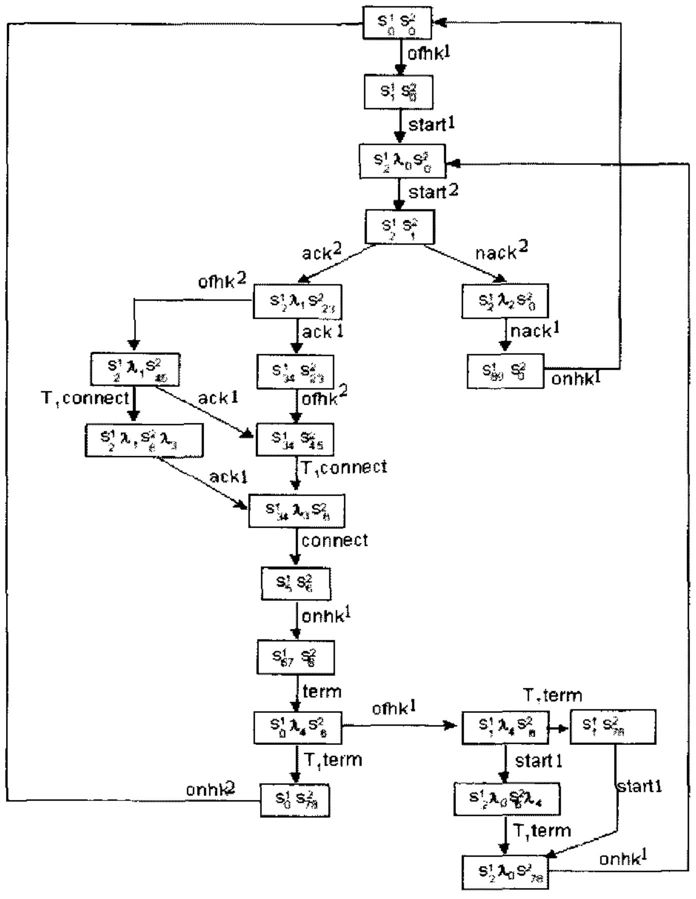
(Fig.4) An example of protocol specs



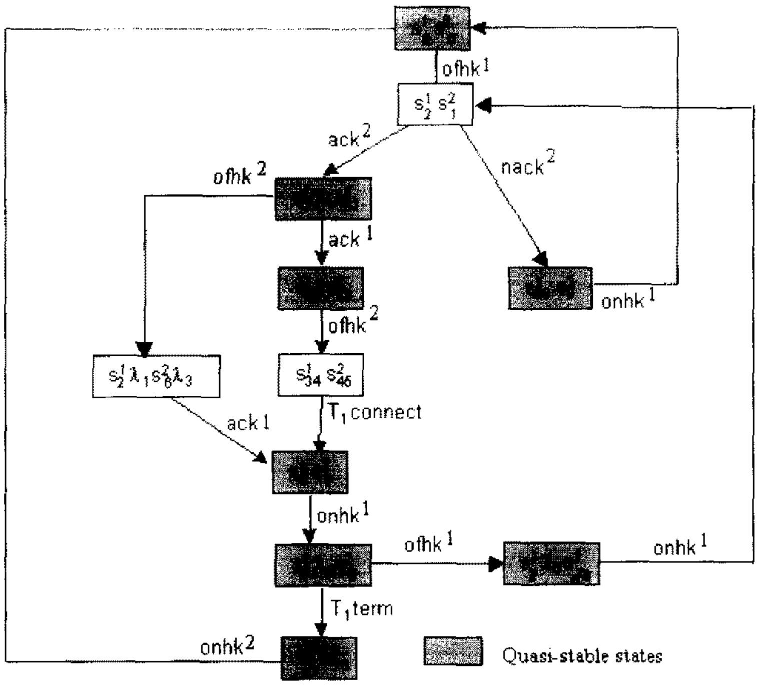
(Fig. 5) Compression of Spec. 1



(Fig. 6) Composition of compressed IOSM's



(Fig. 7) the Compression IOSM



(Fig. 8) Compression of $IOSM_{fin}$

Figures 6 and 7 show composition of two compressed IOSM′ Spec1 and Spec 2. Then we can derive a test case from $IOSM_{fin}$ shown in Fig. 8 with traditional methods, for example see [4]. Total state space of Spec 1 and Spec 2 has 34 states, while obtained one has only 11 states. Further if we are given more IOSM′s with stopping condition, then we can save more space and enjoy the advantage of the proposed method. Finally it is worthy noting that, in $IOSM_{fin}$ shown in Fig. 8, by the definition of stable states given in [1] the path $ofhk_1 \rightarrow onhk_1$ is not appeared and we may miss at least one test case.

5. Conclusions and further work

In this paper, an efficient methodology for IOP test and test suite derivation based on modified HOSS and quasi-stable states. By the proposed method, we can derive a IOP test suite very efficiently in cases of one to many communication protocol has been developed. Further limit of the definition of stable states has been clarified and a new concept of stable state, Quasi-stable states(qss), is defined. By the notion of qss we can derive more coverage can be accomplished. Further the method can be paralleled for speed-up.

Developing more efficient test case generation algorithm is included in further study. Also, computer supported tool for automatic IOP test and test suite generation has to be developed and simultaneously more case studies are needed.

References

[1] Arakawa, N. and Soneoka, T., "A Test Case Generation Method for Cocurrent Programs", Protocol Test Systems, IV, J. Kroon, R. J. Jeijink and E. Brinksma (Eds.) Elsevier Science Publishers B. V., 1992.

[2] Godefroid, P., "Partial-Order Methods for the Verification of Concurrent Systems, An

approach to the state-explosion problem". Lecture notes in computer science 1032. Springer, 1996.

[3] Kang, S. and Kim, M., "Test Sequence Generation for Adaptive Interoperability Testing". IFIP TC6/WG6.1 The 8th International Workshop on Protocol Test Systems, 1995.

[4] Kang, S. and Kim, M., "Interoperabili Test Suite Derivation for Symmetric Communication Protocols", to be appeared in the Proc. of FORTE 97, 1997.

[5] Notomi, M. and Murata, T., "Hierarchical Ada (Petri) Net State Space for Reachability and Deadlock Detection", Proc. of 8th Workshop on Discrete Event Systems, 1991.

[6] Nakagawa, M., Kumagai, S., Miyamoto, T., and Lee, D.-I. "Equivalent Net Reduction for Firing Sequence". IEICE Trans. Fundamentals, Vol. E78-A, No. 11, 1995.

[7] Park, M., Osakzaki, N., Ohta, M., Takahashi K., Shiratori, N. and Noguchi, S., "Test Sequence generation Method for Interoperability Testing Exploiting the Independency of Processes", Trans. of IEICE B-1, Vol. J76-B-1, No. 3, 1993. (in Japanese)

[8] Rafiq, O. and Castanet, R., Fro Conformance Testing to Interoperability Testing. The 3rd International workshop on Protocol Test Systems, 1990.

[9] Valmari, A.., "A Stubborn Attack on State Explosion, In Proc. of 2nd Workshop on Computer Aided Verification, Lecture notes in computer science 531, Springer, 1990.

[10] Valmari, A., "On-the-fly Verification with Stubborn Sets", In Proc. of 5th Conference of Computer Aided Verification, Lecture notes in computer science 697, Springer, 1993.

[11] Valmari, A., "Compositionality in State Space Verification Methods", In Proc. of 17th International Conference on Applications and Theory of Petri Nets, Lecture notes in computer science 1091, Springer, 1996.

최 영 한

1981년 경북대학교 전자공학과(공학사)
1992년 충남대학교 전산학과(이학석사)
1997년~현재 충남대 컴퓨터과학과 박사과정
1982년~현재 한국전자통신연구원 표준연구센터, 표준시험연구팀, 책임연구원.
관심분야 : 프로토콜시험, 이동통신망.

진 병 문

1976년 서울대학교 공과대학 전기공학과(학사)
1983년 서울대학교 공과대학 컴퓨터공학과(석사)
1996년 한국과학기술원 전산학과(박사)
1980년~현재 한국전자통신연구원 책임연구원, 표준연구센터장
관심분야 : 프로토콜시험, 프로토콜공학, 컴퓨터 네트워크

이 동 익

1985년 영남대학교 전기공학과(공학사)
1989년 Osaka대학 전자공학과(공학석사)
1993년 Osaka대학 전자공학과(박사)
1990년~1995년 Osaka대학 전자공학과 Research Associate
1993년~1994년 Univ. of Illinois at Urbana Champaign, CSRL 객원교수
1995년~현재 광주과학기술원 정보통신공학과 조교수
관심분야 : 병행시스템 이론 및 설계, Petri-Nets, 정형기법, 비동기 회로 설계 및 CAD, 에이전트 지향 시스템 등.

진 성 일

1978년 서울대학교 계산통계학과 (학사)

1980년 한국과학기술원 전산학과 (석사)

1994년 한국과학기술원 전산학과 (박사)

1987년~1989년 Northwestern대학 전산학과 객원교수

1990년~1992년 충남대학교 전자계산소장

1989년~현재 정보과학회 데이터베이스 연구회 운영, 편집, 협력위원

1994년~현재 충남대학교 컴퓨터과학과 교수

1995년~현재 정보처리학회 멀티미디어연구회 운영위원

관심분야 : 데이터베이스, 성능분석, 정보모델링, 멀티미디어