Proceedings of
The 9th International Conference on Information Networking (ICOIN-9)

Senri Life Science Hall, Senri, Osaka, Japan
Dec 12-14, 1994

Sponsored by:
Information Processing Society of Japan (IPSJ), Japan
Co-Sponsored by:
IEEE Computer Society
Korea Information Science Society (KISS), Korea
Industrial Technology Research Institute (ITRI), Taiwan

Protocol Synthesis from Acyclic Formed Service Specifications

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
Phone: +81 6 844 1151 (ext. 4841)   Fax: +81 6 850 3051
E-mail: {masa-n, kakuda, kikuno}@ics.es.osaka-u.ac.jp

Abstract

In the conventional protocol synthesis, it is generally assumed that primitives in service specifications cannot be executed simultaneously at different Service Access Points (SAPs). Thus if some primitives are executed concurrently, then protocol errors of unspecified receptions occur.

In this paper, we try to extend a class of service specifications from which protocol specifications are synthesized by the previous methods, to a new class of an acyclic formed service specifications in which parallel execution of primitives is permitted. We introduce priorities into primitives such that one primitive of the highest priority is selected from a set of primitives executable simultaneously and executed. Then, based on this execution ordering, we propose a new protocol synthesis method which can avoid protocol errors due to message collisions, communication competitions and so on.

Key Words

protocol engineering, protocol synthesis, parallel execution of primitives

1 Introduction

Rapid progress of computer-communication systems enables advancement and diversification of communication services. In order to realize such services, it is required to establish efficient and reliable design method for large-scale and complicated communication protocols.

Protocol synthesis is a method to derive a protocol specification from a service specification and it is recognized widely as one of the most promising methods to meet the above requirement. In the protocol synthesis, service specification prescribes relationships between service primitives from either user in a higher layer or process in a lower layer. On the other hand, a protocol specification which is derived from the given service specification defines relations on messages between processes in the lower layer. Interfaces between the higher layer and the lower layer are called Service Access Points, shortly SAPs.

The behavior of protocols can be modeled by Finite State Machines, shortly FSM. Thus both service specification and protocol specification are represented by FSM. Saleh et al.[6] and Liu et al.[1, 2] have already proposed synthesis methods of protocol specifications modeled by FSM. In these synthesis methods, it is not allowed that parallel execution of service primitives occurs at different SAPs. However, it is generally observed that the parallel execution of multiple primitives can occur in real communication systems.

Figure 1 shows a sequence chart which illustrates parallel execution of two service primitives. User 1 sends a primitive *Connection Request*(C_req1) in order to make connection with User 2, and User 3 also sends a primitive C_req3 to connect with User 2 at the same time. Parallel execution of two primitives C_req1 from User 1 and C_req3 from User 3 brings on a competition for an access with User 2. Protocol errors of unspecified receptions are generally caused by a parallel execution of service primitives, since the parallel execution induces message collision and access competition in the lower layer.

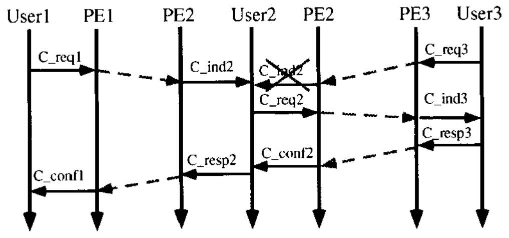

Figure 1: Sequence chart when parallel execution occurs.

A protocol synthesis method to derive a protocol specification including message collisions from a service specification is proposed in [4]. However, in this method the number of processes in the service specification is restricted to two. In this paper, we extend the number of processes from two to $n(\geq 3)$ and propose a new synthesis method which derives a protocol specification from an acyclic formed service specification with multiple primitives executable in parallel.

The rest of this paper is organized as follows. In

Section 2, definitions of service and protocol specifications are given. In Section 3, the protocol synthesis problem is formulated and the class of service specification is discussed. Section 4 describes the details of the synthesis method and Section 5 concludes this paper.

2 Preliminary

2.1 Service Specification

A service specification defines sequences of primitives to be realized as communication services, which are exchanged between users and processes through SAP. Each service access point is denoted by SAPi, and each protocol entity is denoted by PEi.

Definition 1: A service specification, shortly S-SPEC, is modeled by a *Finite State Machine* (FSM) **S** $=< S_s, \Sigma_s, T_s, \sigma >$ where

(1) S_s is a non-empty finite set of service states (or simply states).

(2) Σ_s is a finite set of service primitives (or simply primitives). Each primitive $p \in \Sigma_s$ has, as an attribute, an index of service access point through which p passes. If primitive p passes through SAPi, then we define a function $sap(p) = i$, and also p_i denotes it.

(3) T_s is a partial transition function: $S_s \times \Sigma_s \rightarrow S_s$. For simplicity, we use T_s also as a set of triples (u, p, v) such that $v = T_s(u,p)$ $(u, v \in S_s, p \in \Sigma_s)$.

(4) $\sigma \in S_s$ is an initial service state.

An example of the S-SPEC is shown in Figure 2. In this figure, an oval denotes a service state, an arrow denotes a transition between states. The state drawn by bold line is an initial state.

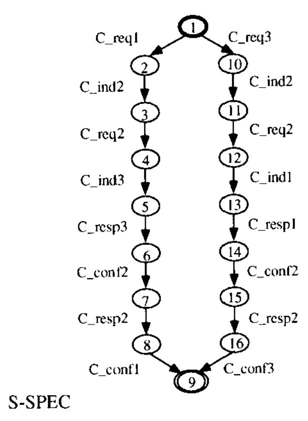

Figure 2: Example of a service specification S-SPEC.

A sequence of transitions (u_1, p_1, u_2) (u_2, p_2, u_3) ... (u_k, p_k, u_{k+1}) from u_1 to u_{k+1} in a S-SPEC implies or defines an execution ordering of primitives p_1, p_2, ..., p_n.

Definition 2: Consider an S-SPEC **S** $= < S_s$, Σ_s, T_s, $\sigma >$. For any node which represents a service state $s \in S_s$ in **S**, let $OUT(s)$ denotes a set of indices of primitives which leaves from s, and let $d^+(s)$ denotes the number of edges outgoing from s. We classify service states into four kinds of states.

(1) If $d^+(s) = 0$, then s is called *final state.*

(2) If $d^+(s) = 1$, then s is called *normal state.*

(3) If $d^+(s) \geq 2$ and $\mid OUT(s) \mid = 1$, then s is called *choice state.*

(4) If $d^+(s) \geq 2$, $\mid OUT(s) \mid > 1$ and $\mid OUT(s) \mid = d^+(s)$, then s is called *parallel state.*

We also call the state $s \in S_s$ *join state* if more than one edges enter s.

Consider again S-SPEC shown in Figure 2. In this S-SPEC, state 9 is a final state because $d^+(9) = 0$ and state 9 is also a join state since two edges enter state 9. Next, state 1 is a parallel state, since $d^+(1) = 2, |OUT(1)| = |\{1,3\}| = 2 > 1$ and thus $d^+(1) = |OUT(1)|$. It implies that at state 1, primitives C_req1 and C_req3 can be concurrently executed at SAP1 and SAP3 respectively. Other states 2, 3,.., 8, 10 ,.., 15 and 16 are normal states. It implies that at each normal state a primitive is executable at a certain SAP.

2.2 Protocol Specification

Transmission and reception of messages are defined as follows.

Definition 3: If message e is transmitted to PEj , then it is denoted by an event message !e(j). If message e is sent to PEj_1, PEj_2,.., PEj_k, then it is denoted by an event message !e(j_1, .., j_k). On the other hand, if message **e** is received from PEj, then denoted by an event message ?e(j) .

The protocol specification consists of n-tuples of specifications for protocol entities. PEs communicate with each other through underlying communication medium. The protocol specification is also modeled by FSM.

Definition 4: A protocol entity specification PE-SPEC i is defined as a FSM P_i=<S_{ip},Σ_{ip},T_{ip}, σ_{ip}> where

(1) S_{ip} is a non-empty finite set of protocol states (or simply states).

(2) Σ_{ip} is a non-empty finite set of protocol events. $\Sigma_{ip} = \Sigma_{is} \cup MEX_i \cup \{\varepsilon\}$, where Σ_{is} is a set of primitives in Definition 2, and MEX_i is a set of events messages which are sent from PEi or received by PEi.

(3) T_{ip} is a partial transition function: $S_{ip} \times \Sigma_{ip} \rightarrow S_{ip}$. For simplicity we use T_s also as a set of triples (u, p, v) such that $v = T_{ip}(u, p)$.

(4) $\sigma_{ip} \in S_{ip}$ is an initial protocol state.

A PE-SPEC is especially called fundamental protocol entity specification, shortly f-PE-SPEC, if the graph of the PE-SPEC and a graph of S-SPEC are isomorphic. The formal definition of the f-PE-SPEC is omitted due to the limited pages and can be referred to [5].

Protocol specification P-SPEC **P** consists of n-tuples of PE-SPECi Pi '$s(1 \leq i \leq n)$.

An example of a P-SPEC is shown in Figure 3. In this figure, an oval denotes a protocol states, and an arrow denotes a transition.

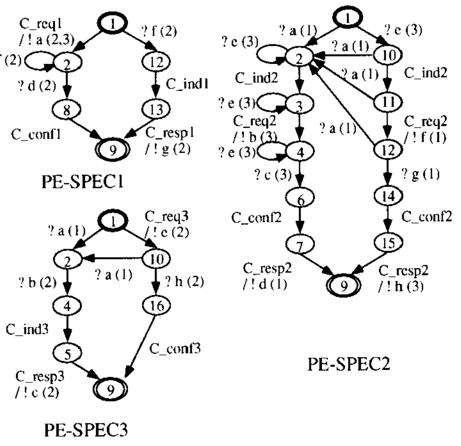

Figure 3: Example of a protocol specification P-SPEC.

We assume that communication links between any two protocol entities are modeled by two unidirectional reliable queues and that protocol messages are delivered in a FIFO order. Suppose that a current state of PEj is state u. If transition (u, E, v) or $(u, E/!e(X), v)$ is specified in T_{jp}, then we say primitive E is executable or primitive E and transmission of message e are executable, respectively. If primitive E is executed, PEj enters state v.
Consider a case that message x that is sent by PEi is on the top of FIFO channel from PEi to PEj $(j \neq i)$. If transition $(u, ?x(i), v)$ is specified in T_{jp}, then we say reception of message x from PEi is executable. If PEj receives message x, then x is deleted from top of the queue and PEj enters state v. If multiple transitions are executable, one of those is non-deterministically chosen and executed.

Definition 5: Consider a case that message x that is sent by PEi is on the top of FIFO channel from PEi to PEj $(j \neq i)$ and a current state of PEj is state u.
If there does not exist any edge $(u', ?x(i), v)$ in T_{jp}, such that there exists a path from state u to u' which only includes (a, r, b) in T_{jp} where r is a primitive, transmission of a message or executable reception of a message from other PEk $(k \neq i)$, then we say that an unspecified reception with respect to x occurs in PE-SPECj.

3 Protocol Synthesis

3.1 Protocol Synthesis Problem

This paper imposes the following three restrictions R1, R2 and R3 to assure correctness of the proposed protocol synthesis method.

Restriction R1: Any state in the S-SPEC graph is exactly one of normal state, final state, choice state and parallel state.

Restriction R2: For any parallel state s, let $Pal(s)$ denote a set of primitives attached to edges outgoing from s. Then, for each parallel state s in the S-SPEC, priorities are assigned to all primitives in $Pal(s)$.

Restriction R3: The S-SPEC is acyclic, that is, S-SPEC is either a tree or a directed acyclic graph (DAG). Furthermore, if S-SPEC is a DAG, S-SPEC satisfies the following condition S.

Condition S: In the S-SPEC, if there exists more than one path from a parallel state (let it be w) to some state, then consider any pair of two disjoint paths which start from the parallel state w and converge in a join state z. These paths are denoted by (w, p_1, x_1) (x_1, p_2, x_2) ...(x_{k-1}, p_k, x_k) and (w, q_1, y_1) (y_1, q_2, y_2)...(y_{l-1}, q_l, y_l) such that $x_k = y_l = z$ and $x_m \neq y_n$ $(1 \leq m \leq k-1, 1 \leq n \leq l-1)$. Assume that $sap(p_1) = i$ and $sap(q_1) = j$ $(i \neq j)$. Then these two paths must satisfy the following (1) and (2).

(1) There exists a transition (y_{n-1}, q_n, y_n) such that $sap(q_n) = i$ $(1 \leq n \leq l-1)$. Additionally, let Q be a set of $sap(q)$ such that q is a primitive in the path w to y_{n-1}, and let Q' be a set of $sap(q')$ such that q' is a primitive in the path y_n to y_l $(y_l = z)$. Then $Q \subseteq Q'$.

(2) There exists a transition (x_{m-1}, p_m, x_m) such that $sap(p_m) = j$ $(1 \leq m \leq k-1)$. Additionally, let P be a set of $sap(p)$ such that p is a primitive in the path w to x_{m-1}, and let P' be a set of $sap(p')$ such that p' is a primitive in the path x_m to x_k $(x_k = z)$. Then $P \subseteq P'$.

Protocol Synthesis Problem to be solved in this paper is formally defined as follows:

Input: A service specification S-SPEC with restrictions R1, R2 and R3.

Output: A protocol specification P-SPEC which satisfies Conditions P1 and P2.

Condition P1: The execution order of primitives defined by S-SPEC is kept in P-SPEC.

Condition P2: No unspecified reception caused by parallel execution of primitive occurs in P-SPEC.

The previous protocol synthesis methods [1, 2, 6] could not assure Condition P2, if a service specification which allows the parallel execution of primitives at different SAPs is given. That is, a protocol specification includes unspecified receptions.

3.2 Class of Service Specification

A service specification defines the execution order of service primitives which are exchanged between the User and the protocol entity (PE). In the service specification, the parallel execution of primitives are allowed at a parallel state. When the parallel execution of primitives occurs at a parallel state, each PE related to primitives which are concurrently executed independently triggers execution of primitives in the individual primitive sequence that starts from the parallel state. Consequently, this execution causes the message collisions and the access competitions, which induces unspecified receptions in the previous methods.

For example, consider the S-SPEC shown in Figure 3. This S-SPEC defines a service that three protocol entities make connection with each other. The sequence of transitions which include states 1,2,..,8 and 9 indicates making a connection in the order of PE1, PE2 ,PE3 (we call it Service A). On the other hand, the sequence of transitions which include states 1,10,11,..,16 and 9 also indicates making a connection in the order of PE3, PE2, PE1 (we call it Service B). This S-SPEC allows the execution of both Service A and Service B. Primitives C_req1 and C_req3 initiate the Service A and Service B, respectively. If parallel execution of primitives C_req1 and C_req3 at state 1 occurs, then two services A and B compete with each other and it may cause message collisions and/or access competitions. These phenomena induce unspecified receptions in the previous methods.

In order to solve this problem, we prioritize the primitives which are concurrently executable. When parallel execution of primitives occurs at the parallel state, the execution of primitive with the highest priority is taken precedence to the others and carried out. The execution of the others are aborted. The sequence of primitives which follows the primitive with the highest priority are executed prior to the others. This fundamental idea first appears in [5]. Based on the above approach, we focus on the class of service specifications satisfying the restrictions R1, R2 and R3 and propose the protocol synthesis method from service specification belonging to that class.

We have discussed above that when parallel execution of primitives occurs, a primitive sequence with the highest priority (here we call it highest sequence) is executed and the other primitive sequences with lower priority (we call them lower sequences) are aborted. First, consider that S-SPEC forms a tree. Execution of the lower sequences stops at some leaf node of S-SPEC even in the worst case. Next, consider that S-SPEC forms a DAG. If there exists more than one path from a parallel state to a join state and parallel execution of primitives occurs at the parallel state, then execution of lower sequences may possibly reach the join state faster than that of highest sequence. In this situation, execution precedence of the highest sequence from the parallel state to a final state via the join state to the other sequence is violated. Condition S gives a sufficient condition to synthesize a protocol so that this execution precedence is preserved.

After expanding DAG formed S-SPEC into tree formed one, we can synthesize the protocol specification from that one. In general, the number of states in the expanded S-SPEC tends to become quite large. This expansion approach is undesirable. We thus propose the protocol synthesis from acyclic service specifications.

4 Proposed Method

4.1 Outline

The proposed method to derive a protocol specification P-SPEC from a service specification S-SPEC consists of the following five steps.

Step 1: Based on given priorities of primitives, assign the priorities to all primitive execution sequences in a service specification S-SPEC.

Step 2: Obtain $n(\geq 2)$ projected service specifications PS-SPECs by applying the projection to a service specification S-SPEC.

Step 3: Construct $n(\geq 2)$ fundamental protocol entity specifications f-PE-SPECs by applying transition synthesis rules (to be shown in Table1) to PS-SPECs.

Step 4: Obtain protocol entity specifications PE-SPECs by adding some transitions for parallel execution of primitives to f-PE-SPECs refined at Step 3.

Step 5: Remove ε transitions from each PE-SPEC, and obtain a protocol specification P-SPEC.

4.2 Step 1

In this step, priorities are assigned to all primitive in a service specification S-SPEC. Priorities are used to identify which primitive to be given preference of execution, when parallel execution of primitives occurs.

From Restriction R2, priorities are pre-assigned to some primitives, that is, primitives leaving from any parallel state in S-SPEC. Based on the priorities of those primitives, the priorities to all primitives are assigned in the order of the Depth First Search.

Consider a S-SPEC shown in Figure 2. State 1 is parallel state and two primitives C_req1 and C_req3 are leaving from state 1. Suppose that we give the higher priority C_req1. Figure 4 shows a S-SPEC obtained by applying Step 1. In the figure, priorities are described by numbers at the side of transitions, and the less the number is, the higher priorities are assigned.

For the service specification S-SPEC in which priorities are assigned to all primitives, we define parallel subgraph PSG in the following.

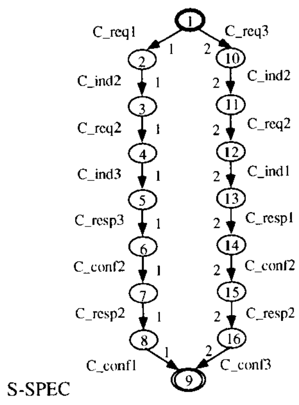

Figure 4: A service specification S-SPEC after Step 1.

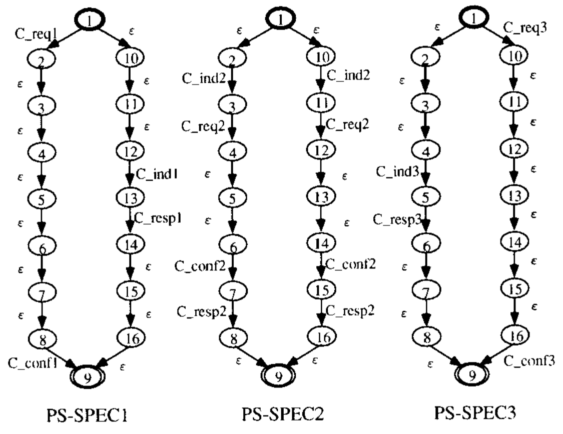

Figure 5: Projected service specifications PS-SPECs.

Definition 6: Consider any parallel state w and a primitive E attached to an outgoing edge from w in the directed graph **G** representing S-SPEC. Assume that $sap(E) = i$ and a priority x is assigned to E. Then a parallel subgraph PSG-S(w,E) is a maximal connected subgraph **G'** of **G** which satisfies the following (1) and (2).

(1) **G'** has a root w.

(2) For any edge (a,p,b) in **G'**, $sap(p) \neq i$ and a priority assigned to p is lower than x.

And consider the directed graph **H** representing f-PE-SPECi based on the S-SPEC **G**. Then PSG-PE$i(w,E)$ is a connected subgraph **H'** of **H** such that **H'** is a fundamental protocol entity specification based on PSG-S(w,E).

Consider the S-SPEC shown in Figure 4, $PSG(1, C\_req1)$ is a subgraph of the S-SPEC which consists of the states 1,10,11 and 12, and $PSG(1, C\_req3)$ is just state 1.

4.3 Step 2

In this step, projected service specifications PS-SPECi ($1 \leq i \leq n$) are obtained from a service specification S-SPEC by substituting each transition not associated with SAPi by ε. For brevity, the formal definition of PS-SPEC is not presented in this paper and can be referred to [5].

As an example, consider a service specification S-SPEC shown in Figure 2. Then Figure 5 shows resultant three PS-SPECs obtained from S-SPEC.

4.4 Step 3

In this step, $n(\geq 2)$ fundamental protocol entity specifications f-PE-SPECs (see Definition 4) are obtained from $n(\geq 2)$ projected service specifications PS-SPECs. This transformation is performed by applying transition synthesis rules shown in Table 1. In Table 1, $E_i (1 \leq i \leq n)$ denotes some primitives in the PS-SPECi. Each pair of transition synthesis rules Ak and Bk ($1 \leq k \leq 3$) is applied to n pairs of transitions (S_1, E_i, S_2) in PS-SPECi and (S_1, ε, S_2) in PS-SPECj ($j \neq i$), respectively. Message e is uniquely generated for each primitive E_i in Rules Ak and Bk (k = 2,3). The concepts for these rules are explained in [5].

By applying transition synthesis rules to the PS-SPECs shown in Figure 5, a f-PE-SPECs shown in Figure 6 are obtained.

Table 1: Transition synthesis rules.

Rule	*Input*	*Condition*	*Output*	
A1	$S_1 \xrightarrow{E_i} S_2$ PS-SPEC i	S1 is not parallel state and OUT(S2)= { i } .	$S_1 \xrightarrow{E_i} S_2$ PE-SPEC i	
B1	$S_1 \xrightarrow{\varepsilon} S_2$ PS-SPECj (≠i)		$S_1 \xrightarrow{\varepsilon} S_2$ PE-SPEC j (≠ i)	
A2	$S_1 \xrightarrow{E_i} S_2$ PS-SPEC i	S1 is not parallel state and OUT(S2)≠ { i } .	$S_1 \xrightarrow{Ei / ! e(X)} S_2$ PE-SPEC i	
B2	$S_1 \xrightarrow{\varepsilon} S_2$ PS-SPECj (≠i)	X = OUT(S2) - {i}	$S_1 \xrightarrow{? e(i)} S_2$ PE-SPEC j (j∈X)	$S_1 \xrightarrow{\varepsilon} S_2$ PE-SPEC k(k∉X)
A3	$S_1 \xrightarrow{E_i} S_2$ PS-SPEC i	S1 is parallel state. Y= { y ∣ y = sap(p), ∀p in PSG-S(S1,Ei)}	$S_1 \xrightarrow{Ei / ! e(X)} S_2$ PE-SPEC i	
B3	$S_1 \xrightarrow{\varepsilon} S_2$ PS-SPECj (≠ i)	X=(Y∪OUT(S2)) - { i }	$S_1 \xrightarrow{? e(i)} S_2$ PE-SPEC j (j∈X)	$S_1 \xrightarrow{\varepsilon} S_2$ PE-SPEC k(k∉X)

4.5 Step 4

According to the projection in Step 2 and the transition synthesis rules in Step 3, f-PE-SPECs obtained from Step 3 are based on the given S-SPEC. That is, there exists one-to-one correspondence between any state in PE-SPECi and that in S-SPEC, and between any edge in PE-SPECi and that in S-SPEC. In order to avoid the unspecified reception due to parallel execution of primitives, some transitions of message reception are added to PE-SPECs refined at Step 3 such that the sequence of primitives with the highest priority can be taken precedence.

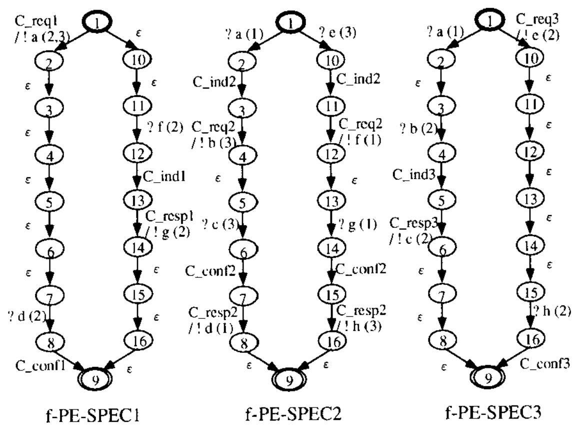

Figure 6: Fundamental protocol entity specifications f-PE-SPECs after Step 3.

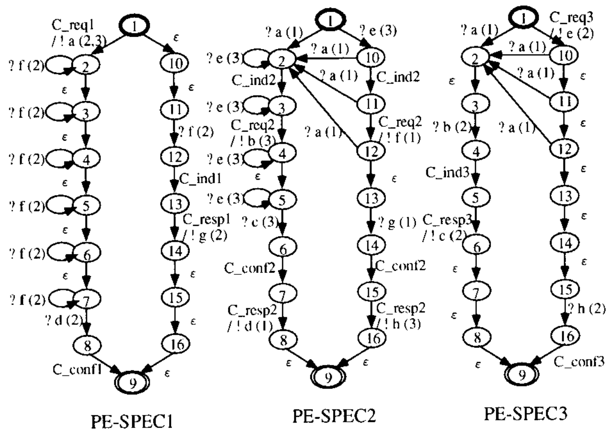

Figure 7: A protocol specification after Step 4.

A concrete method for adding transitions to PE-SPECs is as follows :

> Consider any transition $(w, Ei/!e(X), v)$ such that it leaves from a state w in f-PE-SPECi, where the state w corresponds to a parallel state in S-SPEC, and that E_i is a primitive related to SAPi. The following procedures A and B are performed.

Procedure A: Let Y be a set of indices of SAPs through which primitives in PSG-S(w, Ei) pass. For each state u except w in each f-PE-SPECj $(j \in Y)$, such that u is in PSG-PE$j(w, Ei)$, insert a transition $(u, ?e(i), v)$.

Procedure B: Consider transitions of message receptions in PSG-PE$h(w, Ei)$ for each f-PE-SPECh $(h \in Y \cup \{i\})$. For each $?a(l)$ of message receptions, insert transitions $(s, ?a(l), s)$ where states s are in paths from v to states at which another message reception from PEl is specified.

Consider f-PE-SPECs in Figure 6. Then, Figure 7 shows resultant protocol entity specifications PE-SPECs in which some transitions are added to f-PE-SPECs for parallel execution of primitives.

4.6 Step 5

In this step, ε transitions are removed from the PE-SPECs. The ε transitions are removed by applying the ε removal algorithm in [3]. Using this algorithm, PE-SPECs can be reduced to equivalent finite state machines in the sense that connectivity between any pair of two states is preserved.

Figure 3 shows an example of a final protocol specification P-SPEC after Step 5.

5 Conclusion

In this paper, we have proposed a new synthesis method of a protocol specification from a given service specification which has an arbitrary number of processes and which allows concurrent execution of multiple primitives at different SAPs. Therefore, more reliable protocol specifications can be efficiently synthesized than the previous methods.

References

[1] Chu, P. M. and Liu, M. T., "Protocol synthesis in a state transition model," *Proc. COMPSAC'88*, pp.505-512, Oct. 1988.

[2] Chu, P. M. and Liu, M. T., "Synthesizing protocol specifications from service specifications in the FSM model," *Proc. Computer Networking Symp.*, pp.173-182, April 1988.

[3] Hopcroft, J. E. and Ullman, J. D., "Introduction to Automata Theory, Language, and Computation," Chapter 3, Addision-Wesley, 1979.

[4] Kakuda, Y., Igarashi, H. and Kikuno, T., "Automated synthesis of protocol specifications with message collisions and verification of timliness," *Proc. of Second Int'l. Conference on Network Protocols(ICNP'94)*, Oct. 1994.

[5] Kakuda, Y., Nakamura, M. and Kikuno, T., "Automated synthesis of protocol specifications from service specifications with parallelly executable multiple primitives," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, Oct. 1994.

[6] Saleh, K.,"Automatic synthesis of protocol specifications from service specifications," *Proc. Int'l. Phoenix Conference on Computers and Communications*, pp.615-621, March 1991.