

# Automated Synthesis of Protocol Specifications from Service Specifications with Parallely Executable Multiple Primitives

Yoshiaki KAKUDA†, Masahide NAKAMURA† and Tohru KIKUNO†, *Members*

**SUMMARY** 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. We first introduce priorities into primitives in protocol specification so that it always selects exactly one primitive of the highest priority from a set of primitives that can be executed simultaneously, and executes it. 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. By applying the proposed synthesis method, we can automatically synthesize a protocol specifications from a given service specification which includes an arbitrary number of processes and allows parallel execution of primitives.

**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 [5], [8]. The principle of protocol synthesis is depicted in Fig. 1. As shown in Fig. 1, 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 between messages from 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], [7] 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 2 shows a sequence chart which illustrates parallel execution of two service primitives. User 1 sends a primitive *Connection Request* (CN\_req1), and then sends a primitive *Abort Request* (AB\_req1) successively before receiving the acknowledgement primitive *Connection Confirmation* (CN\_conf1) from User 2. Parallel execution of two primitives *Abort Request*

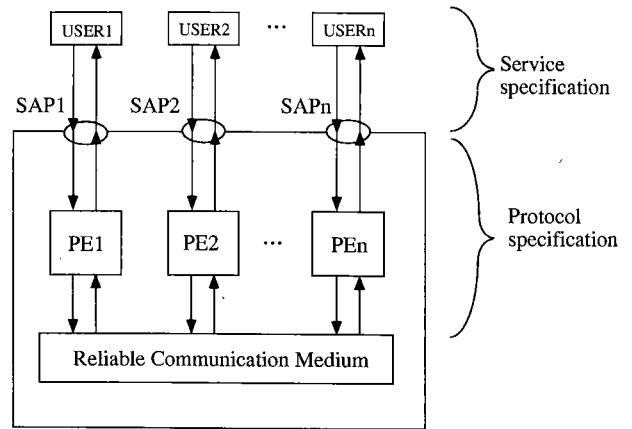


Fig. 1 Communication architecture model.

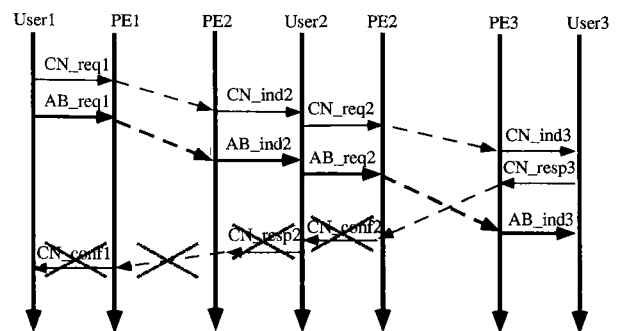


Fig. 2 Sequence chart when parallel execution occurs.

Manuscript received May 12, 1994.

† The authors are with the Faculty of Engineering Science, Osaka University, Toyonaka-shi, 560 Japan.

(*AB\_req2*) from User 2 and *Connection Response* (*CN\_resp3*) from User 3 brings on a crossing of two lines between processes PE2 and PE3. This situation is called a message collision. 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.

Igarashi et al. [4] first proposed a protocol synthesis method to derive a protocol specification including message collisions from a service specification. 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 a service specification with multiple primitives executable in parallel.

The rest of this paper is organized as follows. In Sect. 2, definitions of service and protocol specifications are given and the protocol synthesis problem is formulated. Section 3 explains an outline of a new protocol synthesis method and Sect. 4 describes the details of the synthesis method. Section 5 gives the correctness proof of the proposed method. In Sect. 6, conclusion and future researches are summarized.

## 2. Definitions

### 2.1 Communication Model

As shown in Fig. 1, a communication service is specified by service primitives exchanged between users in the higher layer and processes in the lower layer through service access points (SAPs), and the internal architecture of communication system can be viewed as a black box from user's view point. The processes are also called protocol entities which are denoted by PEs in the following.

Services to User  $i$  are provided by PE  $i$  through SAP  $i$ . Each of PEs is linked by an underlying communication medium. In this paper, we assume that the communication medium is reliable and that messages are delivered in a FIFO order. Each channel between any two PEs modeled by two unidirectional queues.

### 2.2 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 SAP  $i$ , and each protocol entity is denoted by PE  $i$ .

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

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

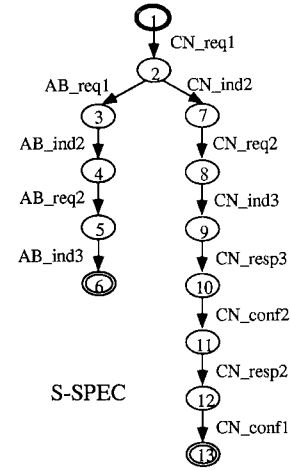


Fig. 3 Example of a service specification S-SPEC.

simply states).

- (2)  $\Sigma_s$  is 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 SAP  $i$ , 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.

A service specification  $S$  is often represented by a labeled directed graph. We call this graph a S-SPEC graph. However, we often refer the S-SPEC graph by S-SPEC.

In the S-SPEC graph, a node is defined for each state in  $S_s$ . For each transition  $(u, p, v) \in T_s$  ( $u, v \in S_s, p \in \Sigma_s$ ), we define an edge from node  $u$  to node  $v$ , and attach a label  $p$  to the edge. In the following, we refer this edge by  $(u, p, v)$ .

An example of the S-SPEC graph is shown in Fig. 3. 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.

Next, projection is used for dividing a service specification into a set of projected service specifications (PS-SPEC). Each service specification PS-SPEC  $i$  corresponds to each PE  $i$ .

**Definition 2:** Let  $S = \langle S_s, \Sigma_s, T_s, \sigma \rangle$  be a given service specification. Then a service specification  $S' = \langle S', \Sigma_{is}, T_{is}, \sigma' \rangle$  defined by the following (1) through (4) is called a projected service specification, shortly PS-SPEC  $i$ , of  $S$ .

- (1)  $S' = S_s$ .
- (2)  $\Sigma_{is} = \{p | p \in \Sigma_s \text{ and } sap(p) = i\} \cup \{\varepsilon\}$ .
- (3)  $T_{is} = \{(u, p, v) | (u, p, v) \in T_s \text{ and } sap(p) = i\} \cup \{(u, \varepsilon, v) | (u, p, v) \in T_s \text{ and } sap(p) \neq i\}$ .
- (4)  $\sigma' = \sigma$ .

In the projected service specification PS-SPEC  $i$ ,

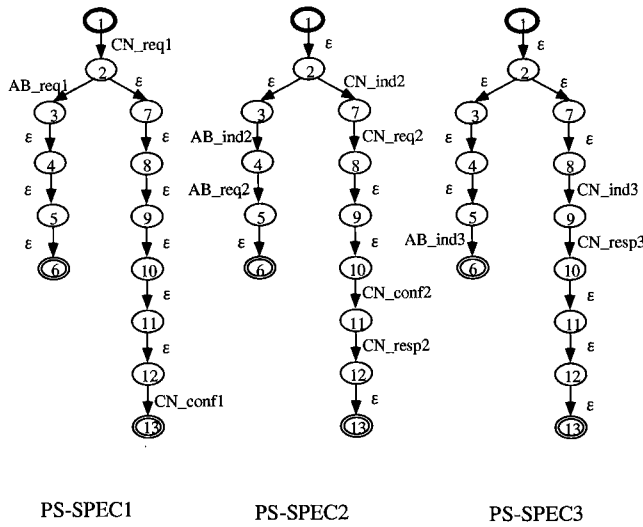


Fig. 4 Example of projected service specifications PS-SPECs.

all service primitives that do not contribute to  $PE_i$  are substituted by a primitive  $\varepsilon$ . The primitive  $\varepsilon$  is null primitive that causes no message sending. As in the case of S-SPEC, we use a labeled directed graph to represent PS-SPEC $i$ . We call this graph a PS-SPEC $i$  graph. However, we refer the PS-SPEC $i$  graph by PS-SPEC $i$ .

**Definition 3:** Consider an S-SPEC graph which represent a service specification  $S = \langle S_s, \Sigma_s, T_s, \sigma \rangle$ . For any node which represents a service state  $s \in S_s$  in  $S$ , define  $OUT(s) = \{i | p \text{ is a label attached to an edge outgoing from } s \text{ and } sap(p) = i\}$

S-SPEC, shown in Fig. 3, is projected to PS-SPEC1, PS-SPEC2 and PS-SPEC3 shown in Fig. 4. From Definition 3, each PS-SPEC $i$  has the same number of states as S-SPEC and the correspondence between states in PS-SPEC $i$  and S-SPEC is clear. Next, we give some examples of function  $OUT$  on S-SPEC in Fig. 3. For instance  $OUT(1) = \{1\}$ ,  $OUT(2) = \{1, 2\}$ ,  $OUT(3) = \{2\}$ , and  $OUT(6) = \phi$ .

**Definition 4:** For a service state  $s = S_s$ , let  $d^+(s)$  denotes the number of edges outgoing from  $s$  in the S-SPEC graph. 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  $|OUT(s)| = 1$ , then  $s$  is called *choice state*.
- (4) If  $d^+(s) \geq 2$ ,  $|OUT(s)| > 1$  and  $|OUT(s)| = d^+(s)$ , then  $s$  is called *parallel state*.

**Remark 1:** From Definition 4, we do not consider a case of  $d^+(s) \geq 2$ ,  $|OUT(s)| > 1$  and  $|OUT(s)| \neq d^+(s)$ . By the following Restriction 2, we restrict a class of service specifications not to include this case.

Consider again S-SPEC shown in Fig. 3. In this S-SPEC, both states 6 and 13 are the final states because  $d^+(6) = d^+(13) = 0$ . Next, state 2 is a parallel

state, since  $d^+(2) = 2$ ,  $|OUT(2)| = |\{1, 2\}| = 2 > 1$  and thus  $d^+(2) = |OUT(2)|$ . It implies that at state 2, primitives  $AB\_req1$  and  $CN\_ind2$  can be concurrently executed at SAP1 and SAP2 respectively. Other states 1, 3, 4, 5, 7, 8, 9, 10, 11 and 12 are normal states. It implies that at each normal state a primitive is executable at a certain SAP.

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

**Restriction R1:** The S-SPEC is a tree. Additionally, 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  $Pal(s)$ .

R1 implies that in the given S-SPEC there exists at most one directed path between any two states. Based on tree structure, we introduce execution ordering for primitives: Consider two adjacent transitions  $(u, E, v)$  and  $(v, E', w)$ . Then we say primitive  $E$  must be executed before primitive  $E'$ . R1 also implies that any branch states, that is a state  $s$  with  $d^+(s) \geq 2$ , is either choice state or parallel state.

**Definition 5:** Consider a S-SPEC graph satisfying Restriction R1 (that is, the S-SPEC graph is a tree). For any path  $(u_1, p_1, u_2) (u_2, p_2, u_3) \dots (u_k, p_k, u_{k+1})$  where  $u_1$  is a root and  $u_{k+1}$  is a leaf node, then we define an execution ordering among primitives  $p_1, p_2, \dots, p_k$  such that any primitive  $p_i$  must be executed before primitives  $p_{i+1}, p_{i+2}, \dots, p_k$ .

### 2.3 Protocol Specification

Transmission and reception of messages are defined as follows.

**Definition 6:** If message  $e$  is transmitted to  $PE_j$ , then it is denoted by an event message  $!e(j)$ . If message  $e$  is sent to  $PE_{j_1}, PE_{j_2}, \dots, PE_{j_k}$ , then it is denoted by an event message  $!e(j_1, \dots, j_k)$ . On the other hand, if message  $e$  is received from  $PE_j$ , 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 7:** A protocol entity specification PE-SPEC  $i$  is defined as a FSM  $P_i = \langle S_{ip}, \Sigma_{ip}, T_{ip}, \sigma_{ip} \rangle$  where

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

- (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.
- Protocol specification P-SPEC  $\mathbf{P}$  consists of  $n$ -tuples of PE-SPEC $i$   $P_i^s$  ( $1 \leq i \leq n$ ), and P-SPEC  $\mathbf{P}$  is denoted by a FSM  $\mathbf{P} = \langle S_p, \Sigma_p, T_p, \sigma_p \rangle$  where
- (1)  $S_p = S_{1p} \times S_{2p} \times \dots \times S_{np}$ .
  - (2)  $\Sigma_p = \Sigma_{1p} \cup \Sigma_{2p} \cup \dots \cup \Sigma_{np}$ .
  - (3)  $T_p = (T_{1p}, T_{2p}, \dots, T_{np})$ .
  - (4)  $\sigma_p = (\sigma_{1p}, \sigma_{2p}, \dots, \sigma_{np})$ .

**Definition 8:** Suppose that a current state of PE $j$  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, PE $j$  enters into state  $v$ .

Consider a case that message  $x$  that is sent by PE $i$  is on the top of FIFO channel from PE $i$  to PE $j$  ( $j \neq i$ ).

If transition  $(u, ?x(i), v)$  is specified in  $T_{jp}$ , then we say reception of message  $x$  from PE $i$  is executable. If PE $j$  receives message  $x$ , then  $x$  is deleted from top of the queue and PE $j$  enters state  $v$ . If multiple transitions are executable, one of those is non-deterministically chosen and executed.

**Definition 9:** Consider a case that message  $x$  that is sent by PE $i$  is on the top of FIFO channel from PE $i$  to PE $j$  ( $j \neq i$ ) and a current state of PE $j$  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 PE $k$  ( $k \neq i$ ), then we say that an unspecified reception with respect to  $x$  occurs in PE-SPEC $j$ .

As in Definition 1, we use a labeled directed graph to represent PE-SPEC $i$ . We call this graph a PE-

SPEC $i$  graph or simply a PE-SPEC $i$ . For a P-SPEC we similarly use a labeled directed graph, and call it a P-SPEC graph or simply a P-SPEC.

An example of a P-SPEC is shown in Fig. 5. In this figure, an oval denotes a protocol states, and an arrow from  $u$  to  $v$  denotes transition  $(u, p, v)$  or  $(u, p/q, v)$  where  $p \in \Sigma_s$  and  $q \in MEX_1 \cup MEX_2 \cup \dots \cup MEX_n$ . “ $p/q$ ” denotes a successive execution of transmissions of primitive  $p$  and message event  $q$ . At first, all PE-SPECs are at initial states drawn by bold line. For example, PE1 is in state 1, receives primitive  $CN\_req$  from user 1, and sends message “ $a$ ” to PE2. After PE1 enters state 2. Similarly, PE2 receives message “ $a$ ” from PE1 at state 1 and enters state 2.

**Definition 10:** Let  $\mathbf{S} = \langle S_s, \Sigma_s, T_s, \sigma \rangle$  be a given service specification. Then a protocol entity specification  $P_i = \langle S_{ip}^f, \Sigma_{ip}^f, T_{ip}^f, \sigma_{ip}^f \rangle$  satisfying the following (1) through (4) is called a fundamental protocol entity specification based on  $\mathbf{S}$ , shortly f-PE-SPEC $i$ .

- (1)  $S_{ip}^f = S_s$ .
- (2)  $\Sigma_{ip}^f = \Sigma_{is} \cup MEX_i \cup \{\varepsilon\}$ .
- (3)  $T_{ip}^f = \{(u, p, v) \text{ or } (u, p/q, v) \mid (u, p, v) \in T_s \text{ and } sap(p) = i \text{ and } q \in MEX_i\} \cup \{(u, q, v) \text{ or } (u, \varepsilon, v) \mid (u, p, v) = T_s \text{ and } sap(p) \neq i \text{ and } q \in MEX_i\}$ .
- (4)  $\sigma_{ip}^f = \sigma$ .

**Remark 2:** A fundamental protocol entity specification f-PE-SPEC $i$  has the same number of states and edges as that of the service specification S-SPEC. There exists one-to-one correspondence between any state in f-PE-SPEC $i$  and that in S-SPEC, and between any edge in f-PE-SPEC $i$  and that in S-SPEC.

Based on the S-SPEC in Fig. 3, PE-SPECs in Fig. 6 are f-PE-SPECs.

**Definition 11:** Let  $S_{ip}^f$  be a set of states  $\alpha_i$  in f-PE-SPEC $i$ . Then an  $n$ -tuple  $(\alpha_1, \alpha_2, \dots, \alpha_n)$  is called a

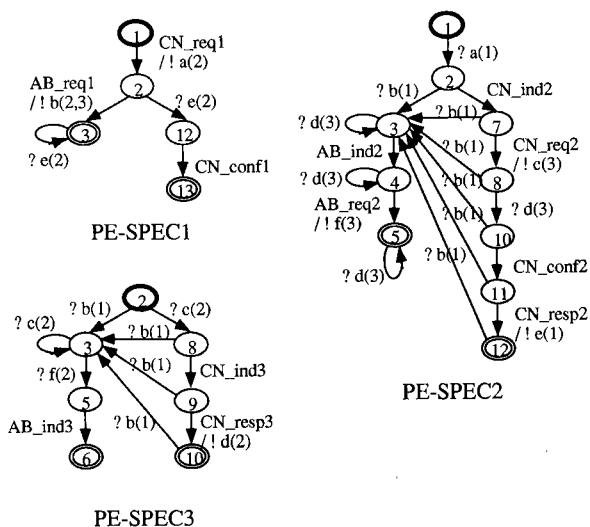


Fig. 5 Example of a protocol specification P-SPEC.

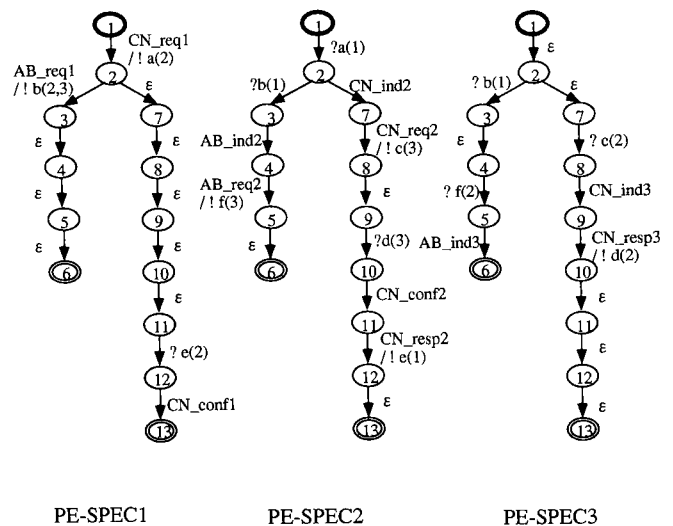


Fig. 6 Example of fundamental protocol entity specifications f-PE-SPECs.

global state. Let  $S-S$  be a set of  $n$ -tuples of  $(\beta_1, \beta_2, \dots, \beta_n)$  where  $\beta_i$  ( $1 \leq i \leq n$ ) is a state in S-SPEC ( $\beta_i \in S_s$ ). A function  $\tau: S_{1p}^f \times S_{2p}^f \times \dots \times S_{np}^f \rightarrow S-S$  is defined as follows.

For each  $\alpha_i \in S_{ip}^f$  ( $1 \leq i \leq n$ ), then  $\tau(\alpha_1, \alpha_2, \dots, \alpha_n) = (\beta_1, \beta_2, \dots, \beta_n)$ , where  $\beta_i = \alpha_i$  ( $1 \leq i \leq n$ ).

**Definition 12:** Consider a S-SPEC satisfying Restriction R1 (that is, a tree graph), and a P-SPEC consisting of f-PE-SPECs.

Let  $\alpha_i$  be a state in f-PE-SPEC $i$ . If an  $n$ -tuple of states  $\tau(\alpha_1, \alpha_2, \dots, \alpha_n) = (\beta_1, \beta_2, \dots, \beta_n)$  satisfies the following Conditions S1 and S2, then we say a global synchronization is kept at global state  $(\alpha_1, \alpha_2, \dots, \alpha_n)$  and such global state is called  $k$ -global state. Otherwise the global synchronization is lost at global state  $(\alpha_1, \alpha_2, \dots, \alpha_n)$  and such global state is called  $l$ -global state.

**Condition S1:** There exists a path  $\rho$  from a root to a leaf of the S-SPEC, such that  $\rho$  contains all of  $\beta_i$ 's.

**Condition S2:** Consider for any  $\beta_i$  and  $\beta_j$  such that  $\beta_i$  is an ancestor of  $\beta_j$  in the S-SPEC. If there exists an edge  $(u, p, v)$  such as  $sap(p) = i$  in the path from  $\beta_i$  to  $\beta_j$ , then for any edge  $(a, q, b)$  in the path from  $v$  to  $\beta_j$ ,  $sap(q) \neq j$ .

We call transition from  $l$ -global state to  $k$ -global state recovery of global synchronization. On the contrary, we call transition from  $k$ -global state to  $l$ -global state the loss of global synchronization.

Consider PE-SPECs shown in Fig. 6 and S-SPEC shown in Fig. 3, then these PE-SPECs are f-PE-SPECs according to Definition 10. Let  $\alpha_i$  be a current state of PE $i$ . Some examples of global synchronization are in the following.

- (1) Consider the case that  $\alpha_1=11$ ,  $\alpha_2=8$  and  $\alpha_3=9$ . In order to map the current states of PEs on S-SPEC, function  $\tau$  is applied. Since  $\tau(11, 8, 9) = (11, 8, 9) = (\beta_1, \beta_2, \beta_3)$ , Conditions S1 and S2 in Definition 12 are satisfied. Therefore in this case a global synchronization is kept.
- (2) Consider the case that  $\alpha_1=3$ ,  $\alpha_2=8$  and  $\alpha_3=9$ . Similarly function  $\tau$  is applied and  $\tau(3, 8, 9) = (3, 8, 9) = (\beta_1, \beta_2, \beta_3)$ . Since Condition S1 is not satisfied, in this case a global synchronization is lost.
- (3) Consider the case that  $\alpha_1=13$ ,  $\alpha_2=8$  and  $\alpha_3=9$ , then  $\tau(13, 8, 9) = (13, 8, 9) = (\beta_1, \beta_2, \beta_3)$ . Since Condition S2 is not satisfied, in this case a global synchronization is lost. Actually, this case cannot happen in these PE-SPECs (shown in Fig. 6). Because the current state of PE1 is state 13 in PE-SPEC1, PE1 must receive message  $e$  from PE2 and primitive  $CN\_conf1$  must be executed. On the other hand, PE2 is at state 8 and message  $e$  is not still sent to PE1.

## 2.4 Protocol Synthesis Problem

Protocol Synthesis Problem to be solved in this paper

is formally defined as follows:

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

**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], [6], [7] 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. Outline of Protocol Synthesis Method

### 3.1 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  $n$  ( $\geq 2$ ) protocol entity specifications PE-SPECs by applying transition synthesis rules (to be shown in Table 1) to PS-SPECs.

**Step 4:** Add some transitions for parallel execution of primitives to PE-SPECs refined at Step 3.

**Step 5:** Remove  $\epsilon$  transitions from each PE-SPEC, and obtain a protocol specification P-SPEC.


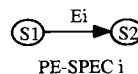
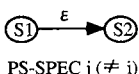
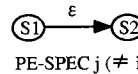
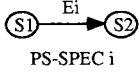
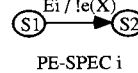
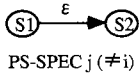

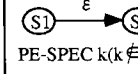
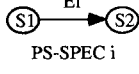
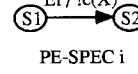
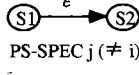
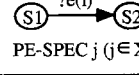
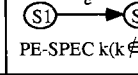
PS-SPECs in Definition 2 are realized in Step 2. And one of f-PE-SPECs in Definition 11 are obtained from S-SPEC in Step 2 and Step 3. Some transitions are added into the f-PE-SPECs in Step 4 to avoid unspecified receptions.

### 3.2 Key Idea

Protocol entities PEs are exchanging messages with each other and provide the communication service according to execution order of primitives described in the service specification S-SPEC. However, since PEs only have local states, they must keep the global synchronization by exchanging messages between PEs.

If no parallel execution of primitives occur, the global synchronization is kept and no unspecified reception occurs as in the previous methods. On the contrary, if a parallel execution of primitives occurs at parallel states, the global synchronization is lost. If the previous synthesis methods are applied to this case, then unspecified receptions occur.

**Table 1** Transition synthesis rules.

Rule	Input	Condition	Output
A1	 PS-SPEC $i$	S1 is not parallel state and $OUT(S2) = \{i\}$ .	 PE-SPEC $i$
B1	 PS-SPEC $j (\neq i)$		 PE-SPEC $j (\neq i)$
A2	 PS-SPEC $i$	S1 is not parallel state and $OUT(S2) \neq \{i\}$ .	 PE-SPEC $i$
B2	 PS-SPEC $j (\neq i)$	$X = OUT(S2) - \{i\}$	  PE-SPEC $j (j \in X)$ PE-SPEC $k (k \notin X)$
A3	 PS-SPEC $i$	S1 is parallel state. $Y = \{y \mid y = sap(p), \forall p \text{ in } PST-S(S1, Ei)\}$	 PE-SPEC $i$
B3	 PS-SPEC $j (\neq i)$	$X = (Y \cup OUT(S2)) - \{i\}$	  PE-SPEC $j (j \in X)$ PE-SPEC $k (k \notin X)$

In order to solve this problem, in the proposed method, we add some transitions for reception of messages to PE-SPECs so that the global synchronization can be recovered. The key idea of the proposed method is as follows. Consider sequences of  $k$ -global states starting from a parallel state, in which if one of primitives leaving from the parallel state is executed, then global synchronization is kept and the execution order of primitives follows that on S-SPEC. Suppose that parallel execution of primitives occurs at the parallel state and that priorities are assigned to primitives. Then, the protocol enters  $l$ -global state. Recovery from  $l$ -global state to some  $k$ -global state in a sequence starting by execution of the primitive with the highest priority is performed by transitions of message reception which are added to PE-SPECs. In the recovery, execution order of primitives is preserved and unspecified receptions are avoided.

Consider the S-SPEC in Fig. 3 and PE-SPECs in Fig. 6. Protocol specification which is synthesized from the S-SPEC (Fig. 3) in the previous methods [6], [7] is essentially equivalent to PE-SPECs shown in Fig. 6.

Let  $\alpha_i$  be a current state of PE $i$  and consider a case  $(\alpha_1, \alpha_2, \alpha_3) = (2, 2, 2)$ . Then global synchronization is kept. Suppose that  $AB\_req1$  is assigned higher priority than  $CN\_ind2$ . If two primitives  $AB\_req1$  and  $CN\_ind2$  are concurrently executed at SAP1 and SAP2, respectively, global state  $(2, 2, 2)$  moves to  $(3, 7, 2)$  and global synchronization is lost. Then unspecified reception with respect to message  $b$  to be sent from PE1 to PE2 occurs in PE-SPEC2. If a transition  $(s, ?b(1), 3)$  ( $s=7, 8, 9, 10, 11, 12, 13$ ) is added in PE-SPEC2, PE2 can enter state 3 in the sequence starting by

execution of  $AB\_req1$  with higher priority and the global synchronization can be recovered to  $(3, 3, 2)$ . Thus, in the proposed method, transitions are automatically added to recover the lost global synchronization and to avoid the unspecified reception.

#### 4. Detail of Protocol Synthesis Method

##### 4.1 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 the parallel execution of primitives occurs. The real protocol specifications are usually designed to deal with primitives executable in parallel. For example, primitives related to RESET are given the top priority, and primitives concerned with ABORT are also assigned high priority.

In Fig. 2, a sequence  $CN\_req, CN\_ind, CN\_resp$  and  $CN\_conf$  represents "connection of communication path" and a sequence  $AB\_req$  and  $AB\_ind$  represents "abort of service." Since higher priority is given to ABORT sequence than CONNECTION sequence, when parallel execution of these primitives sequences occur,  $AB\_req1$  is followed by  $AB\_ind2, AB\_req2$  and  $AB\_ind3$ . On the other hand, CONNECTION sequence is aborted at PE2.

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.

An example of assignment is shown in Fig. 7.

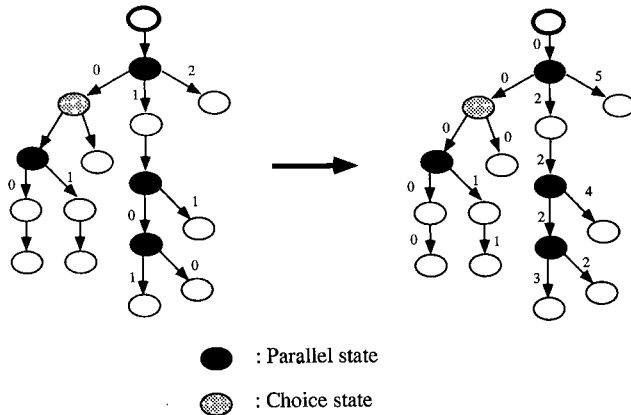


Fig. 7 Assignment of priority.

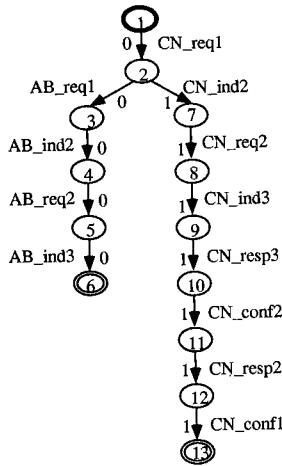


Fig. 8 Example of a service specification S-SPEC after Step 1.

Figure 8 shows a S-SPEC obtained from a S-SPEC shown in Fig. 3 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 sub-tree PST in the following.

**Definition 13:** 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 sub-tree  $PST-S(w, E)$  is a connected subgraph  $G'$  of  $G$  which satisfies the following (1) through (3).

- (1)  $G'$  is a tree with root  $w$ .
- (2) For any edge  $(a, p, b)$  in  $G'$ ,  $sap(p) \neq i$  and a priority assigned to  $p$  is lower than  $x$ .
- (3) There does not exist any graph (let it be  $G''$ ) satisfying (1) and (2) such that  $G' \neq G''$  and  $G'$  is some subgraph of  $G''$ .

And consider the directed graph  $H$  representing f-PE-SPEC $i$  based on the S-SPEC  $G$ . Then  $PST-PEi(w,$

$E)$  is a connected sub-graph  $H'$  of  $H$  such that  $H'$  is a fundamental protocol entity specification based on  $PST-S(w, E)$ .

### 4.2 Step 2

In this step, projected service specifications PS-SPEC $i$  ( $1 \leq i \leq n$ ) are obtained from a service specification S-SPEC by substituting each transition not associated with SAP $i$  by  $\epsilon$ . The formal definition of these substitutions is given in Definition 2.

As an example, consider a service specification S-SPEC shown in Fig. 3. Then Fig. 4 shows resultant three PS-SPECs obtained from S-SPEC.

### 4.3 Step 3

In this step,  $n (\geq 2)$  protocol entity specifications PE-SPECs 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-SPEC $i$ . Each pair of transition synthesis rules  $A_k$  and  $B_k (1 \leq k \leq 3)$  is applied to  $n$  pairs of transitions  $(S_1, E_i, S_2)$  in PS-SPEC $i$  and  $(S_1, \epsilon, S_2)$  in PS-SPEC $j (j \neq i)$ , respectively. Message  $e$  is uniquely generated for each primitive  $E_i$  in Rules  $A_k$  and  $B_k (k = 2, 3)$ .

The intuitive concepts for these rules are explained as follows.

**Rules A1, B1:** These rules imply that any messages need not be transmitted and received for global synchronization because two primitives are successively executed at the same SAP(SAP $i$ ).

**Rules A2, B2:** The primitive  $E_i$  is executed at SAP $i$ . Following the occurrence of  $E_i$ , other primitives can be executed at other SAPs, therefore a message for global synchronization is transmitted to other corresponding protocol entities.

**Rules A3, B3:**  $E_i$  occurs at parallel state. In this case, a message for global synchronization is transmitted not only to PEs in which other primitives are executed next to  $E_i$ , but also to PEs concerned with primitives in parallel sub-tree  $PST-S(S_1, E_i)$  (see Definition 13). These transmissions are done for the parallel execution of primitives. The message generated by these rules is used to recover the loss of global synchronization due to the parallel execution of primitives. However, transitions for the parallel execution of primitives are not generated from transition synthesis rules. The transitions for parallel executions are added in Step 4.

Consider PS-SPECs in Fig. 4. Figure 9 shows all parallel sub-trees in S-SPEC shown in Fig. 8. Consider PS-SPECs in Fig. 4. Then, by applying transition synthesis rules to these PS-SPECs, a protocol specification shown in Fig. 6 is obtained. Examples of

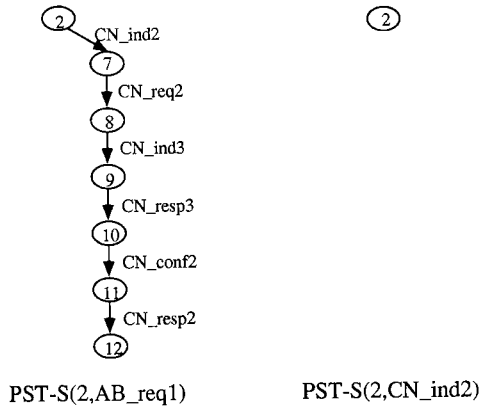


Fig. 9 Example of parallel sub-tree PST.

application of typical three rules are explained in the following.

(1) Transitions  $(1, CN\_req1, 2)$ ,  $(1, \epsilon, 2)$  and  $(1, \epsilon, 2)$  in PS-SPEC1, PS-SPEC2, and PS-SPEC3 are translated into  $(1, CN\_req1/!a(2), 2)$ ,  $(1, ?a(1), 2)$  and  $(1, \epsilon, 2)$  in PE-SPEC1, PE-SPEC2 and PE-SPEC3 by applying rules A2, B2 and B2 respectively. The reason why rules A2 and B2 are selected is that, for  $i=1$ , state 1 is not parallel state and  $OUT(2) \neq \{i\}$ . These rules generate message  $a$  from PE1 to PE2.

(2) Transitions  $(8, \epsilon, 9)$ ,  $(8, \epsilon, 9)$  and  $(8, CN\_ind3, 9)$  in PS-SPEC1, PS-SPEC2, and PS-SPEC3 are translated into  $(8, \epsilon, 9)$ ,  $(8, \epsilon, 9)$  and  $(8, CN\_ind3, 9)$  in PE-SPEC1, PE-SPEC2 and PE-SPEC3 by applying rules B1, B1 and A1 respectively. These rules are selected because for  $i=3$ , state 8 is not parallel state and  $OUT(9) = \{i\}$ . No messages among PEs are generated.

(3) Transitions  $(2, AB\_req1, 3)$ ,  $(2, \epsilon, 3)$  and  $(2, \epsilon, 3)$  in PS-SPEC1, PS-SPEC2 and PS-SPEC3 are translated into  $(2, AB\_req1/!b(2, 3), 3)$ ,  $(2, ?b(1), 3)$  and  $(2, ?b(1), 3)$  in PE-SPEC1, PE-SPEC2 and PE-SPEC3 by applying rules A3, B3 and B3. This is because state 2 is a parallel state. Since the set of indices of PEs related to primitives in  $PST-S(2, AB\_req1)$  is  $\{2, 3\} (= Y)$  and  $OUT(3) = \{2\}$ , message  $b$  is transmitted from PE1 to PE2 and PE3.

4.4 Step 4

According to the projection in Step 2, and the transition synthesis rules in Step 3, PE-SPECs obtained from Step 3 are f-PE-SPECs based on the given S-SPEC. There exists one-to-one correspondence between any state in PE-SPEC $i$  and that in S-SPEC, and between any edge in PE-SPEC $i$  and that in S-SPEC. In order to recover the loss of global synchronization due to parallel execution of primitives, some transitions of message reception are added to PE-SPECs refined at Step 3 such that global synchronization is recovered and the global state converges in the primitive with the

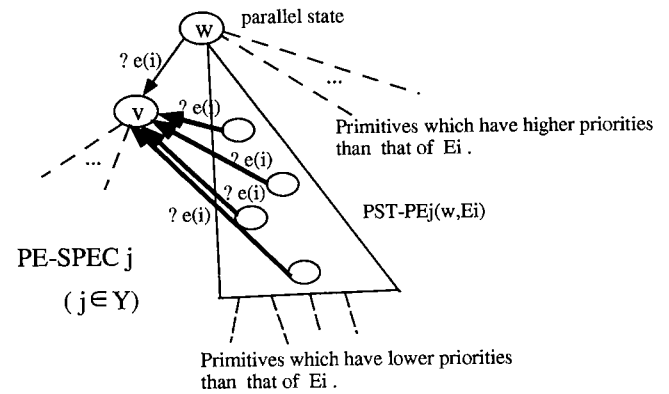
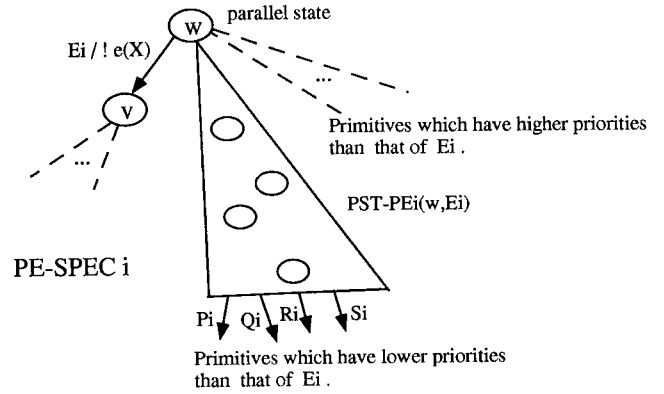


Fig. 10 Explanation for Procedure A in Step 4.

highest priority.

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

Consider any transition  $(w, E_i/!e(X), v)$  such that it leaves from a state  $w$  in PE-SPEC $i$ , 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  $PST-S(w, E_i)$  pass. For each state  $u$  except  $w$  in each PE-SPEC $j$  ( $j \in Y$ ), such that  $u$  is in  $PST-PE_j(w, E_i)$ , insert a transition  $(u, ?e(i), v)$ .

Figure 10 illustrates the procedure A.

**Procedure B:** Consider transitions of message receptions in  $PST-PE_h(w, E_i)$  for each PE-SPEC $h$  ( $h \in Y \cup \{i\}$ ). 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 PE $l$  is specified.

Figure 11 illustrates the procedure B.

**Remark 3:** Let  $G$  be a graph representing PE-SPEC $i$  before Step 4, and let  $G'$  be a graph representing PE-SPEC $i$  after Step 4. As mentioned above,  $G$  represents a fundamental protocol specification f-PE-SPEC $i$ . Then  $G$  is a subgraph of  $G'$  and any state in  $G'$  corresponds to a state in  $G$ .





Step 4 (before Step 5). It is obvious that the  $\epsilon$  removal algorithm [3] in Step 5 preserves the correctness of the protocol specification synthesized by the proposed method.

From Remark 2 and Remark 3, each PE-SPEC obtained at Step 4 has the same number of states as the given S-SPEC. Since each state in PE-SPEC has one-to-one-correspondence to that in S-SPEC, we similarly define the global state and global synchronization for PE-SPECs obtained at Step 4, as for f-PE-SPECs obtained at Step 3.

We prove the correctness of the proposed method for the following three cases respectively.

**Case 1:** No parallel state exists in the given S-SPEC.

**Case 2:** Only one of primitive which leaves from any parallel state in S-SPEC must be executed (that is, parallel execution of primitives does not occur).

**Case 3:** Parallel execution of primitives occurs.

### 5.1 Case 1

In this case, no transition is added at Step 4 since no parallel state exists. PE-SPECs obtained at Step 3 is nothing more than the protocol specification obtained by using Saleh's method [6], [7]. Proof for the correctness of his method is already given in [6], [7].

### 5.2 Case 2

In this case, each parallel state is performed as a choice state. Since parallel execution of primitives is not allowed in the PE-SPECs, Condition S1 is satisfied. By the transition synthesis rules in Step 3, Condition S2 is satisfied [6], [7]. Therefore, global synchronization is kept for PE-SPECs.

**Lemma 1:** While global synchronization is kept for PE-SPECs, Conditions P1 and P2 are satisfied.

**Proof:** By Condition S1 of Definition 12, the current state in all PE-SPECs corresponds to some state in a path of S-SPEC. Let the path be denoted by  $\rho = (u_1, p_1, u_2), \dots, (u_k, p_k, u_{k+1})$ . By Condition S2 of Definition 12, primitive  $p_l$  such that  $sap(p_l) = i$ , must have been executed before  $p_m$  ( $l < m$ ) such that  $sap(p_m) = j$ , ( $j \neq i$ ) is executed. Therefore, Condition P1 is satisfied. Consider two successive transitions  $(u_l, p_l, u_{l+1})$  and  $(u_{l+1}, p_{l+1}, u_{l+2})$  in the path  $\rho$  such that  $sap(p_l) = i$  and  $sap(p_{l+1}) = j$  ( $j \neq i$ ). By the transition synthesis rules at Step 3, a message  $x$  caused by occurrence of primitive  $p_l$  in PE-SPEC $i$  must have received before primitive  $p_{l+1}$  occurs in PE-SPEC $j$ . And only receptions of messages are added at Step 4. Therefore, Condition P2 is satisfied.

By Lemma 1, Conditions P1 and P2 are satisfied in this case.

### 5.3 Case 3

In this case, parallel execution of primitives occurs. As mentioned before, parallel execution causes the loss of global synchronization between PEs.

**Lemma 2:** The number of  $l$ -global states is finite when parallel execution of primitives is performed.

**Proof:** Let a sequence of  $k$ -global states be denoted by  $G_1, G_2, \dots, G_k, G_{k+1}, \dots, G_m$  in which following conditions are satisfied:

- (1) Parallel execution of primitives can occur at global state  $G_k$ .
- (2) For each PE, either (a) a primitive and transmission of a message, (b) reception of a message, or (c) an  $\epsilon$  is executed in the transition from  $G_j$  to  $G_{j+1}$  ( $1 \leq j \leq m-1$ ).

Let denote  $p_j$  for the primitive to be executed in the transition from  $G_j$  to  $G_{j+1}$ . Suppose that by a parallel execution of primitives  $p_k, q^1, \dots, q^r$  at  $G_k$  in this sequence,  $G_k$  moves to  $H_1$  where  $H_1$  is  $l$ -global state. Also suppose that primitive  $p_k$  has the highest priority among  $p_k, q^1, \dots, q^r$ .

By the protocol construction in Step 3 and Step 4, a global state transition diagram is obtained as shown in Fig. 13.

The length of the sequence of  $l$ -global states starting from  $H_1$  is finite due to the following reason. Consider the  $h$ -th component in above  $l$ -global states, that is, a current state of PE $h$  in the PE-SPEC $h$ . The sequence of such states are included in PST-PE $h(w, p_k)$  where  $G_k = (w, \dots, w)$ . By the definition of PST-

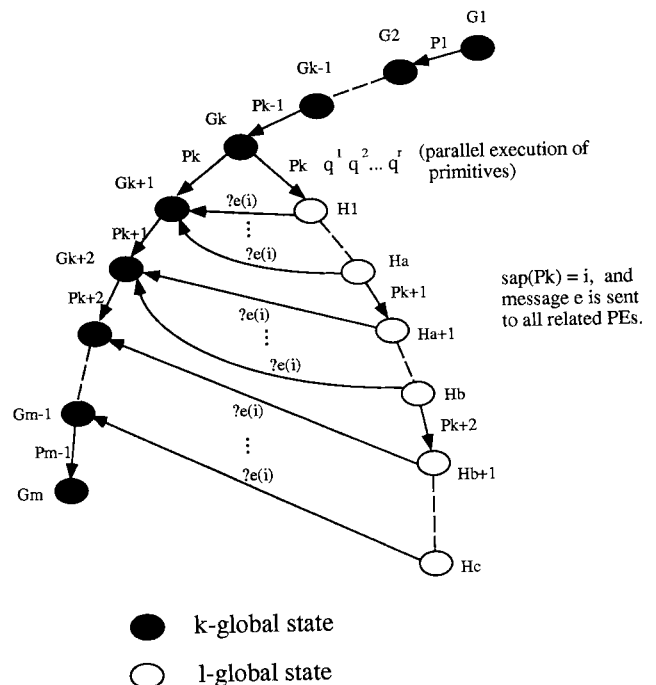


Fig. 13 Global state transition diagram.

$PEh(w, p_k)$  the following two cases are considered.

**Case A:** There is no transition leaving from a leaf of  $PST-PEh(w, p_k)$  in  $PE-SPECh$  because  $S-SPEC$  is a tree.

**Case B:** All sequence of transitions starting from a leaf of  $PST-PEh(w, p_k)$  are  $(s_1, r_1, s_2), (s_2, r_2, s_3), \dots, (s_l, r_l, s_{l+1})$  ( $l \geq 1$ ) where  $r_1 = \varepsilon, \dots, r_{l-1} = \varepsilon$  and  $r_l$  is reception of a message. By the protocol construction in Step 3, reception of the message  $r_l$  is not executable.

In either case, execution of the sequence of  $l$ -global state starting from  $H_1$  stops at some  $l$ -global state. Therefore, Lemma 2 holds.

**Lemma 3:** When parallel execution of primitives is performed, Condition P1 is satisfied.

**Proof:** As shown in the global state transition diagram of Fig. 13, each sequence of transitions starting from  $H_1$  reverts to some  $k$ -global state by reception of all messages which are sent to the related PEs with execution of primitive  $p_i$ . The last reception of such messages are depicted by arrows labeled  $?e(i)$  in Fig. 13. Note that not last reception of messages  $?e(i)$  are included in the sequence of  $l$ -global states. Reception of message  $?e(i)$  in each sequence of transitions which starts from  $H_1$  and finally reverts to some  $k$ -global state corresponds to reception of message  $?e(i)$  in each  $PE-SPECj$  in Fig. 10.

Execution of primitives  $p_1, \dots, p_{m-1}$  in this order are required in Condition P1. Although redundant primitives except  $p_1, \dots, p_{m-1}$  may be executed in the transitions denoted by dotted lines in Fig. 13, by the protocol construction,  $p_j$  must have been executed before  $p_{j+1}$  is executed for all sequences of transitions which starts from  $H_1$  and finally reverts to some  $k$ -global state.

**Lemma 4:** When parallel execution of primitives is performed, Condition P2 is satisfied.

**Proof:** Consider the global state transition diagram in Fig. 13. Messages which are caused by a primitive except for primitives  $p_1, \dots, p_{m-1}$  may be transmitted in the sequence of transitions from  $H_1$  to some  $k$ -global state, which is denoted by  $a$ . Such messages are surely received in the above sequence or in the sequence of  $k$ -global states  $G_{i+1}, \dots, G_m$ . Reception of such messages  $?a(l)$  corresponds to  $?a(l)$  in Fig. 11. Therefore, Condition P2 is satisfied.

**Theorem 1:** Conditions P1 and P2 are satisfied for  $P-SPECs$  obtained in the proposed method.

**Proof:** When global synchronization is kept, by Lemma 1 Conditions P1 and P2 are satisfied. And if parallel execution of primitives occurs and global synchronization is lost, the lost global synchronization is recovered and Conditions P1 and P2 are satisfied by Lemma 3 and Lemma 4.

## 6. 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. The characteristics of this method are:

- (1) When parallel execution of primitives occur, the execution of primitives to which higher priority is assigned takes precedence over that to which lower priority is assigned.
- (2) There are no protocol errors of unspecified receptions caused by parallel execution of primitives. Therefore, more reliable protocol specifications can be efficiently synthesized than the previous methods. However, further research remains, for instance:
  - The relaxation of the restriction that a directed graph representing the service specification  $S-SPEC$  is a tree.
  - Synthesis of protocol specifications on unreliable communication medium.

## 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, Apr. 1988.
- [3] Hopcroft, J. E. and Ullman, J. D., *Introduction to Automata Theory, Language, and Computation*, Chapter 3, Addison-Wesley, 1979.
- [4] Igarashi, H., Kakuda, Y. and Kikuno, T., "Synthesis of protocol specifications for design of responsive protocols," *IEICE Trans. Inf. & Syst.*, vol. E76-D, no. 11, pp. 1375-1385, Nov. 1993.
- [5] Liu, M. T., "Protocol engineering," *Advances in Computers*, vol. 29, pp. 79-195, Academic, 1989.
- [6] Saleh, K., "Automatic synthesis of protocol specifications from service specifications," *Proc. Int'l. Phoenix Conference on Computers and Communications*, pp. 615-621, Mar. 1991.
- [7] Saleh, K., "A service-based method for the synthesis of communication protocols," *Special issue on distributed computing and systems, International Journal of Mini and Microcomputers*, vol. 12, no. 3, pp. 97-103, 1990.
- [8] Saleh, K. and Probert, R. L., "Synthesis of communication protocols: Survey and assessment," *IEEE Trans. Comput.*, vol. 40, no. 4, pp. 468-475, Apr. 1991.

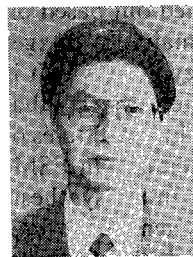


**Yoshiaki Kakuda** was born in Hiroshima, Japan, on June 29, 1955. He received the B.S. degree in electronic engineering from Hiroshima University, Hiroshima, Japan, in 1978. He also received the M.S. degree and Ph.D. degree in system engineering from the same university in 1980 and 1983, respectively. From 1983 to 1991, he was with Research and Development Laboratories, Kokusai Denshin Denwa Co., Ltd. (KDD), where he last

held the position of Senior Research Engineer. From 1991 he has been an Associate Professor with the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. His current research interests include protocol engineering and responsive systems. He was a program co-chairman of the Second International Workshop on Responsive Computer Systems in 1992. He is a member of the IEEE Computer Society and the Information Processing Society of Japan. He received the Telecom. System Technology Award from Telecommunications Advancement Foundation in 1992.



**Masahide Nakamura** was born in Hyogo, Japan, on January 14, 1972. He received the B.E. degree in information and computer sciences from Osaka University, Toyonaka, Osaka, Japan in 1994. He is currently studying towards the M.E. degree in the Graduate School of Engineering Science in the same university. He has been engaged in research on protocol synthesis.



**Tohru Kikuno** was born in Ehime, Japan, on September 11, 1947. He received the B.E., M.Sc., and Ph.D. degrees in electrical engineering from Osaka University, Toyonaka, Osaka, Japan, in 1970, 1972, and 1975, respectively. He joined Hiroshima University from 1975 to 1987. He is currently a Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University since 1990.

His research interests include analysis and design of fault-tolerant systems, quantitative evaluation of software development process and design of testing procedure of computer protocols. He was a general co-chairman of the Second International Workshop on Responsive Computer Systems in 1992. He is a member of the IEEE(U.S.A), ACM(U.S.A) and Inf. Proc. Soc. (Japan). He received the Paper Award from the Institute of Electronics, Information, and Communication Engineers of Japan in 1993.