Synthesis Algorithm of Protocol Specification with Message Collision for Two Processes

4 V − 1　Hirotaka IGARASHI, Yoshiaki KAKUDA and Tohru KIKUNO
Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University

1 Introduction

Along with the development and enrichment of communication services in ISDN and IN, it is strongly demanded to realize highly reliable communication protocols efficiently. Protocol synthesis is to produce communication protocol specifications and it is one of the most important techniques to be developed [1, 3, 4].

The principle of protocol synthesis is illustrated in Figure 1 : Given a service specification representing relations on primitives between a user in a high layer and a process in a low layer, a protocol specification representing relations on messages between processes in the low layer is derived. Interfaces between these layers are called *Service Access Points* (SAPs).

In the previous algorithms for protocol synthesis, protocol specifications with message collisions can be derived but they cause protocol errors called unspecified receptions.

This paper proposes an automated synthesis algorithm of a protocol specification from a service specification such that the synthesized protocol specifications are free from protocol errors of unspecified receptions caused by message collisions.

2 Protocol Synthesis

A *service specification* describes primitive's execution sequences between users and processes through SAPs. In this paper, the number of processes is limited to two. The service access points are thus denoted by SAP1 and SAP2.

Service specifications are modeled by Finite State Machine (FSM) as shown in Figure 2. In this figure, an oval represents a state, an arrow represents a transition between states and a label of the arrow represents a primitive. State number 1 is an initial state.

Primitives s associated with SAP1 are specified as follows: If s is delivered from a user to a process through SAP1 then s is denoted by $s_1\downarrow$, and if s is delivered from a process to a user through SAP1 then s is denoted by $s_1\uparrow$. Primitives s associated with SAP2 are done in a similar way.

A *protocol specification* consists of a number of processes cooperating by sending and receiving messages. Each process is modeled by FSM as shown in Figure 3. In the figure, ! represents sending a message and ? represents receiving a message. A *message collision* occurs between two processes by concurrently sending messages from process1 to process2 and from process2 to process1. Figure 4 shows a sequence chart which represents a message collision. A message collision due to messages Rel_req1 and C_resp2 is denoted by crossing of two dotted lines. Generally, primitives $s_1\downarrow$ and $s_2\downarrow$ have a possibility to induce a message collision if they are concurrently delivered from users to processes. These primitives are called MC (Message Collision) primitives. This paper adopts the following approach to MC primitives: If the priority of $s_1\downarrow$ is higher than that of $s_2\downarrow$, then any message induced by $s_1\downarrow$ reaches SAP2 and vice versa. If the priorities of $s_1\downarrow$ and $s_2\downarrow$ are the same, neither messages reach any SAPs.

The Protocol Synthesis problem (called PS problem) to be solved in this paper is defined as follows.

Input: A service specification and priorities between primitives.

Output: A protocol specification with message collisions.

Conditions: Precedence relation on transitions in the service specification is preserved in the protocol specification. Additionally there is no unspecified receptions in the protocol specification.

3 Proposed Synthesis Algorithm

This section proposes an algorithm to solve the PS problem. The proposed algorithm consists of the following four steps.

[Step1] Add transitions for message collisions to a given service specification.

[Step2] Project the service specification refined by Step1 to two service specifications with respect to SAP1 and SAP2.

[Step3] Apply transition synthesis rules [2] to service specifications with respect to SAP1 and SAP2, and obtain protocol specifications with respect to SAP1 and SAP2.

[Step4] Remove ε transitions from the protocol specifications.

In the algorithm, Step1 finds MC primitives and adds transitions for deleting unspecified receptions caused by message collisions.

The conditions C1 through C3 for finding MC primitives are shown below(See Figure 5). In Fig. 5, $w, w1, w2, u, v, u'$and v' are states and $y, y', s_i\downarrow, s'_j\downarrow, s''_j\downarrow$and $s'''_i\downarrow$ are primitives in the given service specification.

Here, $s''_j\downarrow$ represents the first primitive that is associated with SAP$j(\neq i)$ and delivered from a user to a process, and that appears after $s_i\downarrow$. $s'''_i\downarrow$ represents the first primitive that is associated with SAP$i(\neq j)$ and delivered from a user to a process, and that appears after $s'_j\downarrow$.

- Condition C1: There exist $s_i\downarrow$ and $s'_j\downarrow$.
- Condition C2: There exists a state (say w), from which u and u' are reachable.
- Condition C3: There exist y and y' whose origin states are both w, and $y = y_i\downarrow$ or $y_i\uparrow$ and $y' = y'_j\downarrow$ or $y'_j\uparrow$ $(i \neq j)$.

If conditions C1-C3 are satisfied, $s_i\downarrow$ and $s'_j\downarrow$ are determined as MC primitives.

Under these conditions, transitions for message collisions are added to service specification as follows. This procedure is divided into three cases.

- Case 1 (The priority of $s_i\downarrow$ is higher than that of $s'_j\downarrow$ and there does not exist any other $s_i\downarrow$ in the path from w to v'.)

  For each state (let it be x) in the path from $w2$ to $s'''_i\downarrow$, if there are no outgoing transitions $s_i\downarrow$ from x ,then insert a transition labeled Li from x to v.

  For each state (let it be y) in the path from $w1$ to $s''_j\downarrow$, insert a transition labeled Lj from y to y itself.

- Case 2 (The priority of $s_i\downarrow$ is higher than that of $s'_j\downarrow$ and there exists another s_i $\downarrow$ in the path from w to v'.)

  This case is similar to Case 1 except that messages substituted with Li and Lj are different each other for $s_i\downarrow$ and $s_j\downarrow$.

- Case 3 (The priorities of $s_i\downarrow$ and $s'_j\downarrow$ are the same.)

  For each state (let it be x) in the path from $w2$ to $s'''_i\downarrow$, if there are no outgoing transitions $s\downarrow$ from x, then insert a transition labeled Li from x to the initial state.

  For each state y in the path from $w1$ to $s''_j\downarrow$, insert a transition labeled Lj from y to the initial state.

Explanations of Step2,Step3 and Step4 are omitted due to limitation of pages. Refer to [2] for their details.

Figure 3 shows a protocol specification synthesized from the service specification in Figure 2 using the proposed algorithm.

Time complexities of this algorithm for each steps are evaluated as follows: Let n represent the number of all states and T represent the number of all transitions in the given service specification.

Step1 is searching MC primitives and adding transitions for message collisions. For searching MC primitives, it takes at most T^2 times to confirm Condition C1 and it takes at most T times to confirm Conditions C2 and C3. For adding transitions for message collisions, it takes $O(T^2n)$ times. The complexity of Step1 is thus evaluated as $O(T^3+T^2n)$ times. Step2 is projection of service specification to two service specifications with respect to SAP1 and SAP2. Projection is done for each transition. The complexity of Step2 is thus evaluated as $O(2T)$ times. Step3 is application of rules to two service specifications with respect to SAP1 and SAP2. The complexity of Step3 is evaluated as $O(2T)$ times. Step4 is removal of ϵ transitions in the protocol specification. The complexity of Step4 is evaluated as $O(T^n)$, but it could be decreased to $O(T)$ in practical cases.

4 Conclusions

This paper has proposed a synthesis algorithm of a protocol specification from a service specification. The characteristics of this algorithm include that there are no unspecified receptions caused by message collisions in the synthesized protocol specification. Therefore, more reliable protocol specifications are efficiently produced by this algorithm than those by the previous synthesis algorithms [1, 3, 4]. The formal proof of correctness is now extensively being studied.

References

[1] Peil-Ying M. Chu and Ming. T. Liu: "Protocol synthesis in a state transition model," Proc. COMPSAC'88, pp.505-512 (Oct. 1988).

[2] Hirotaka Igarashi, Yoshiaki Kakuda and Tohru Kikuno: "Synthesis of protocol specifications with recovery function from service specifications," IEICE Japan, Tech. Group Paper SSE92-10 (May 1992), in Japanese.

[3] Kassem Salch:"Automatic synthesis of protocol specifications from service specifications," Proc. Int'l. Phoenix Conference on Computers and Communications, pp.615-621 (March 1991).

[4] Kassem Saleh and Robert L. Probert :"Synthesis of communication protocols: Survey and assessment," IEEE Trans. on Computers, Vol.40, No.4, pp.468-475 (April 1991).

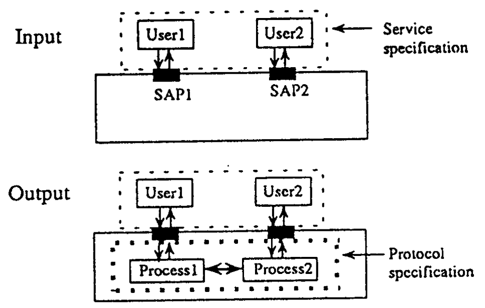

Figure 1 Protocol synthesis

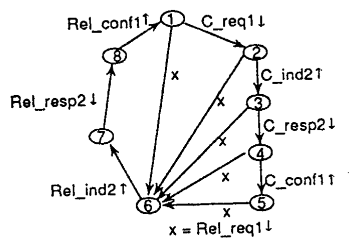

Figure 2 Example of a service specification

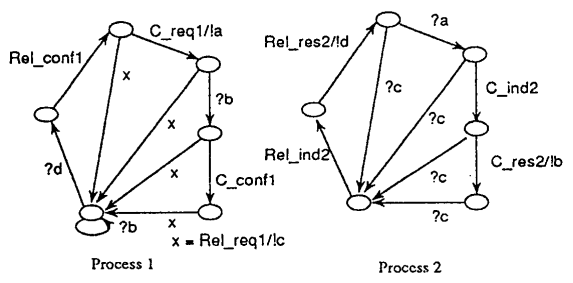

Figure 3 Example of a protocol specification

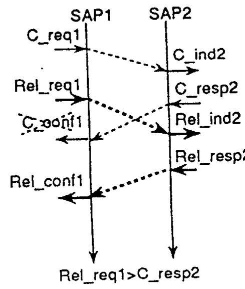

Figure 4 A sequence chart representing a message collision

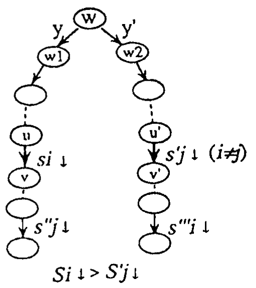

Figure 5 Conditions for finding MC primitives