Automatic Conversion from CCM to HCM in State Transition Model

Akira Takura
Department of Career Planning and Information Studies
Jumonji University
Niiza, Japan
e-mail: takura@jumonji-u.ac.jp

Tadashi Ohta
IEICE Fellow

Tokorozawa, Japan
e-mail: myupapa@jcom.home.ne.jp

***Abstract*—As a program processing model for network software systems, the state transition model is well known. In the state transition model, there are two models: a central call model and a half call model. Each model has merits and demerits. So, if automatic conversion between the two models can be used, it would be convenient for system developers to design the system by adopting the appropriate model for the purpose of the design. A method of automatically converting from a half call model to a central call model has already been proposed. This paper proposes a method of automatically converting from a central call model to a half call model. The proposed method was applied to a three-way call service and it was confirmed that the conversion was correctly carried out.**

Keywords-state transition model; central call model; half call model; automatic conversion

I. INTRODUCTION

As a program processing model for network software systems, the state transition model has well been used [1][2]. The state transition model contains two models. One is called a central call model (abbreviated as CCM) where states of all terminals receiving a service are described in one state. The other model is called a half call model (abbreviated as HCM) where states of different terminals are described in different states, respectively. As states of all terminals receiving the service are described as one state in the CCM, service is easily understood. However, when the number of combinations for states of each terminal increases, the number of states in the CCM increases, resulting in increasing the size of programs implemented based on the CCM.

If an automatic conversion between CCM and HCM is established, the appropriate model can be applied for the purpose of the design. A method for automatic conversion from HCM to CCM has been proposed in [3]. In this paper, a method for automatically converting from CCM to HCM is proposed. The proposed method was applied to a three-way call service, call transfer service, call waiting service, and VoIP service, and it was confirmed that all state transition diagrams were correctly converted from CCM to HCM.

N. D. Grifeth proposed the method for creating a state transition diagram based on signals input to and output from the system whose function is unknown [4]. The created state transition diagram is based on CCM. S. K. Chakrabarti proposed the method for creating a state transition diagram to analyze functions of API [5]. Shimokura proposed the program described in a rule-based language , which is based on state transitions, for controlling networked robots [6]. The embedded program in AIBO, which is a robot made by Sony and was used in Shimokura's system, was designed based on state transition diagrams. Other researches of using a rule-based language are proposed by S. Sen, X. Fei, and L. Dongliang [7-9]. M. Ohba proposed the method for creating state transition diagrams based on CCM from programs described in a rule-based language [10].

In Section II, formal description of states is described so that computer processing is possible. In Section III, problems in converting from CCM to HCM such as conversion of states, state synchronization and reduction of states are described. In Section IV, solutions for the problems described in Section III are proposed. In Section V, an example of applying the proposed method to a three-way call service, main part, is described.

II. A FORMAL EXPRESSION OF STATE

For a computer to handle a state, the state of a state transition diagram has to be described formally. This is how it is done: Each state of the state transition diagram is described as a set of state description primitives (called primitive) which represent states of terminals which are receiving a service at that time [11][12]. A primitive consists of a primitive name which represents a state of a terminal and arguments which represent the terminals related to the primitive name. For example, when terminal A is in an idle state, it is described as idle(A). When users of terminals A and B are talking, it is described as talk(A,B). 'idle' and 'talk' are primitive names, and A and B are arguments, respectively. When terminals A and B are arguments of the same primitive, it is said that terminals A and B are connected. If terminals A and B are connected and terminals B and C are connected, then terminals A and C are connected. Thus, all terminals described in the same state of a state transition diagram are connected.

In a state transition model, where a state transition is decided based only on a current state and an event, the number of states becomes large. So, in the state transition model generally used, the state transition is decided not only on the current state and the event but also on state transition

 ISBN: 978-1-61208-186-1

conditions. State transition conditions are described, in the state transition diagram, in an analysis block below the current state. In this paper, state transition conditions are also described as a set of primitives. A primitive which is used only as a state transition condition and never used as an element of any state in the state transition diagram, is called, in this paper, a 'condition primitive'. On the other hand, a primitive which is used as an element of a state is called, in this paper, a 'state primitive'.

An example of a state transition diagram with two analysis blocks for a call waiting service is shown in Fig. 1. Four states and two analysis blocks are involved in Fig. 1. States are described as ovals, each of which has a set of primitives: dialtone(A), calling(A,B) {cw-calling(A,B), talk(B,C)}, and busy(A), respectively. Analysis blocks are described as triangles which have conditions: idle(B) and {m-cw(B), talk(B,C)}, respectively. When event dial(A,B) occurs at state dialtone(A), the next state is decided according to condition idle(B) described in the first analysis block. If idle(B) holds, the next state is calling(A,B). If idle(B) does not hold, the next state is decided according to condition {m-cw(B), talk(B,C)} described in the second analysis block.

III. PROBLEMS IN CONVERSION FROM CCM TO HCM

A. Conversion of States

In CCM, states of all terminals receiving a service are described in one state. But, in HCM, states of different terminals are described in different states, respectively. So, states in CCM should be converted so that states of different terminals are described in different states.

B. State Synchronization

In HCM, an event occurred in the system is independently accepted in individual state of state transition diagram related to a terminal which creates the event. More than one event might occur at one time. So, when the event is accepted in the state transition diagram, there is no guarantee that all terminals are at states which correspond to the same state of CCM. Thus, a mechanism which guarantees that all terminals are at states which correspond to the same state of CCM is needed.

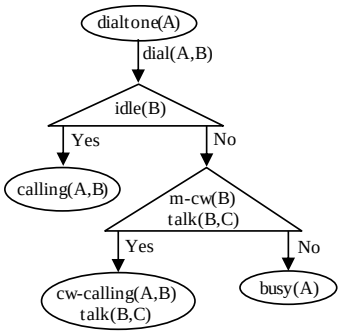

Figure 1. An Example of Two Analysis Blocks

C. Reduction of States

There is a case where, while a state of CCM is changed but a corresponding state of an HCM is not changed; there is a reduction of states. For example, in a three-way call service, when a user of terminal A, who is talking with a user of terminal B, wants to let a user of terminal C join the conversation, the user of terminal A pushes a flash button. Then terminal B is in hold state and is kept in the hold state until the three-way call is established. The state of CCM is changed but a state of terminal B in an HCM is not changed. Then, one state of terminal B in HCM corresponds to several states in CCM.

When the terminal, whose state corresponds to several states in CCM, creates an event, in HCM for terminal B, it has to ask all terminals which are receiving the service to distinguish to which state of CCM the state of terminal B in HCM corresponds.

IV. SOLUTIONS

A. Conversion of States

Each state in CCM is described as a set of primitives as mentioned in Section II. So, to convert a state in CCM to a corresponding state of terminal X in HCM, select primitives which have terminal X as an argument from the state in CCM. Thus the corresponding state of terminal X in HCM is gained as a set of the selected primitives. Primitives, that have more than one argument, appear in states of each argument in HCM. See Fig. 2.

B. Synchronization of States

First, a signal sequence for asking states of each terminal is proposed. Then, methods to synchronize states of each terminal are proposed based on the proposed signal sequence, in two cases; the state in the HCM where an event occurs is decided, and the state is not decided because of state reduction. The solution in the case that the state in HCM is reduced is described in *C*.

1) Signal sequence

In an HCM where an event occurs it has to ask current states of all terminals which are receiving the service by sending a signal to all terminals without a repetition. But, terminals to which the signal is sent are limited to terminals which are described in the current state of the HCM. In other word, the signal can be sent between terminals which are arguments of the same primitive described in the current state in the HCM.

First, based on a set of primitives in the current state of CCM, make a directed graph, by drawing arrows originating

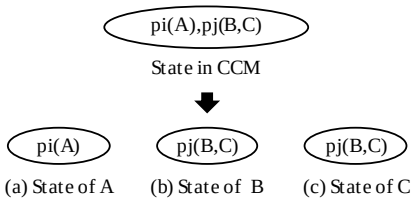

Figure 2. Converted States in HCM

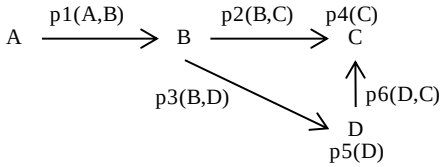

Figure 3. Graph Corresponding to a CCM State

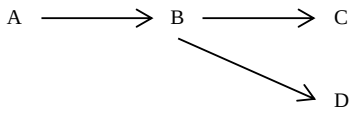

Figure 4. Extracted Tree from the Graph

from the terminal, which creates the event, to terminals which are described as arguments in the primitives where the originating terminal is also described as an argument [13]. As mentioned in Section II, all terminals described in the current state of CCM are connected. So, all terminals can be described as a node of the graph. Suppose the current state of CCM is {p1(A,B),p2(B,C),p3(B,D),p4(C), p5(D), p6(D,C)} and terminal A creates an event. In this case, the graph is shown in Fig. 3. A primitive, which has one argument, is described above or under the argument. A primitive, which has two arguments, are described above the arrow drawn between the two arguments.

Next, as the signal should be sent to each terminal only once, delete unnecessary arrows from the graph. Then delete primitives. Now, a tree which shows a signal sequence is obtained as shown in Fig. 4.

By sending the signal from the terminal which creates an event based on this tree, the signal can be sent to all terminals with certainty and without repetitions.

2) State of CCM is decided

In the case that the current state of CCM is decided, the HCM, where an event occurs, checks that all other terminals are in the states which correspond to the decided state, and synchronize states of all terminals. To realize the synchronization, the HCM corresponding to the root node in the tree shown in Fig. 4 sends an inquiring signal (REQ) to the next node and transits to a wait state. When the node, which receives REQ, is not in the corresponding state it sends back a signal NG to the former node which sent REQ. In this case, since there are no states which receive REQ, NG is sent from state analysis program. When the node which receives REQ is in the corresponding state and is not a leaf node, called an inner node in the tree, it transmits REQ to the next node and transits to a wait state to keep synchronization. If the node which receives REQ and is in the corresponding state is a leaf node, it sends back a signal OK to the former node which sends REQ, and transits to a wait state to keep synchronization.

The node, which receives NG, sends a reset signal to all the next nodes that have sent OK, sends NG to the former node, and transits to the current state. The node, which receives the reset signal and is an inner node, sends the reset signal to all the next nodes and transits to the current state. If the node is a leaf node, it transits to the current state.

When a node receives OK from the next node, if there is the next node to which REQ has not been sent, the node sends REQ to the next node. If there is not the next node to which REQ has not been sent there are two cases: the node is an inner node or the node is a root node. If the node is an inner node it sends OK to the former node. When the node is a root node, it sends a signal ST, which represents an instruction of state transition, to all the next nodes in parallel and transits to the next state. When an inner node receives ST it retransmits ST to all the next nodes in parallel and transits to the next state instructed by ST. When a leaf node receives ST it transits to the next state instructed by ST.

Note: The signal which instructs a state transition is not sent to the terminal which does not make a state transition nor retransmit the signal to another terminal.

C. State in HCM is Reduced

1) Process for synchronization

Information that each state in HCM corresponds to which state in CCM is saved in the process for converting a current state in CCM to individual states in HCM. For all terminals described in a current state of CCM, referring to the saved information, make an AND set of states in CCM to which a current state corresponds. Thus, the state in CCM, to which each state of individual terminals corresponds, can uniquely be decided. The signal to decide the unique state is sent based on the tree described in *B 1)*.

More precisely, the process is explained as follows: The root node sends REQ, which is a set of states in CCM corresponding to the current state of the node, to the next node as an inner event, and transits to wait state. An inner node which receives REQ, makes an AND set of received REQ and a set of states in CCM corresponding to the current state of the inner node to make a new REQ. Then, the inner node sends the REQ to the next node as a new inner event, and transits to wait state. If the AND set is null, a vacant set, the inner node sends back NG to the former node and transits to the current state. A leaf node which receives REQ, makes an AND set in the same way as the inner node. If the AND set is not null, the leaf node sends back the AND set as ANS to the former node, and transits to wait state. If the AND set is null, the leaf node sends back NG to the former node, and transits to the current state. The process hereafter is the same as one described in *B 2)* in this section under the condition of replacing signal OK with ANS and replacing "sending REQ" with "making new REQ and sending it". Thus, finally the root node can decide the corresponding state in CCM and can send ST to all the next nodes.

2) Example of signal sequence

Concrete examples of signal sequence are explained in the case where the current states of CCM are as follows:

S1: {p1(A,B),p2(B,C),p3(B,D),p4(C),p5(D),p6(D,C)}
S2: {p1(A,B),p2(B,C),p4(C)}
S3: {p1(A,B),p3(B,D),p5(D)}
S4: {p1(A,B),p2(B,C),p7(C)}

Terminals A, B, C and D are in states corresponding to S1, S2, S3, or S4 in CCM. But, for certain terminals, some

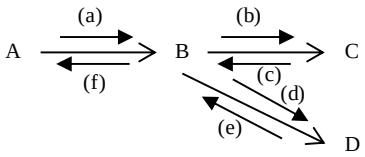

Figure 5. Tree Expression for S1

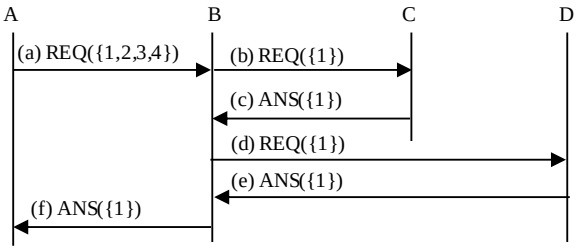

Figure 6. The Signal Sequence for S1

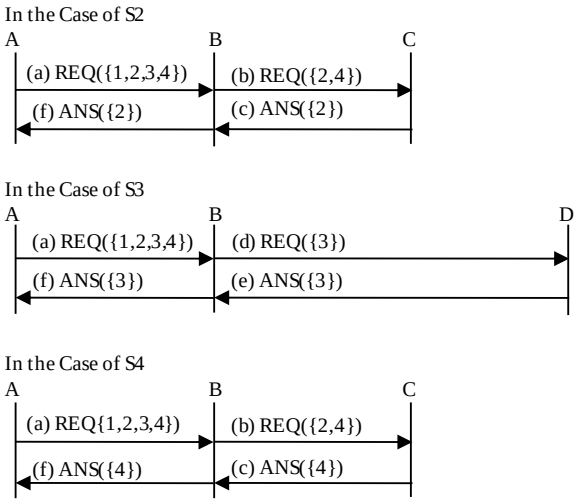

Figure 7. The Signal Sequence for S2, S3 and S4

states in HCM might be the same state. In this case, the terminal may not be able to decide to which state in CCM the current state of the terminal corresponds. In this example, terminal A is in the same state in HCM for S1, S2, S3, and S4 in CCM. Terminal B is in the same state in HCM for S2 and S4 in CCM. Terminal C is in the same state in HCM for S1 and S2 in CCM. So, when terminal A creates an event, in HCM terminal A sends {S1,S2,S3,S4} to the next node based on the tree described in *B 1)* in this section. Terminals A, B and C behave as described in the second paragraph in *C 1)* in this section. Thus, terminal A can distinguish to which state in CCM the current state of terminal A corresponds.

The tree expression for S1 is shown in Fig. 5. Signals are sent based on this tree. The signal sequence is shown in Fig. 6. (a): Since terminal A is in the state which corresponds to S1, S2, S3 and S4, terminal A sends {1,2,3,4} as REQ to terminal B. (b): As terminal B is in the state corresponding to S1, an AND set of REQ sent from terminal A and {1} is made, resulting in gaining {1} as new REQ. Terminal B sent {1} as REQ to terminal C. (c): Since terminal C is in S1, an AND set of REQ sent from terminal B and {1} is made, resulting in gaining {1}. As terminal C is a leaf node, it sends {1} as ANS to terminal B. (d): Since the node of terminal B has a branch to terminal D, terminal B sends {1} received as ANS from terminal C to terminal D as REQ. (e): By making an AND set of received REQ and its state which corresponds to S1, terminal D gains {1}. Terminal D is a leaf node, so it sends {1} as ANS to terminal B. (f): Terminal B sends {1} received from terminal D to terminal A as ANS. Thus, terminal A can recognize its state as S1.

Signal sequences for S2, S3 and S4 are shown in Fig. 7. Consequently, each state can be recognized.

D. Conversion for an Analysis Block

Primitives for plural terminals can be described in the analysis block. As far as the authors know, transition conditions for at most one terminal, other than a terminal which creates an event, is described in most of conventional telephone services. So, in this paper, the way of converting the analysis block is described in the case where transition conditions for at most one terminal, other than a terminal which creates an event, is described in the analysis block. The proposed method can be easily extended to the case where the analysis block has transition conditions for more than two terminals.

Analysis blocks of CCM are converted after conversion described in B and C in this section. In an analysis block conversion, if arguments of primitives represent the terminal where the event occurred, the analysis block is described at just below the current state in the HCM corresponding to the terminal. Otherwise, i.e. arguments of primitives in the analysis block are not the terminal, the way of converting the analysis block depends on whether primitives described in the analysis block are state primitives or condition ones as described below.

1) In the case of a state primitive

When the primitives described in the analysis block are state primitives, there is an argument which appears in arguments of the event. So, in the HCM where the event occurs, inquiry signal, REQ, to the HCM designated by the argument of the event is described below the event. In the HCM where REQ is received, REQ as an input signal and OK as a reply signal are described just below the state containing the primitive described in the analysis block of the CCM. And then, transition to wait state is described. Note that NG signal is not described in the HCM. This is because when the current state of the HCM cannot accept REQ, NG is sent by a state analysis program as mentioned in Section IV *B 2)*. In the HCM which receives OK, the OK signal is described as an input signal just below the wait state, and an instruction of state transition to the HCM, which sent the OK signal, is described.

Fig. 8 shows an example of converting a state transition diagram, where a state primitive p2(B) is described in the analysis block, from CCM to HCM by the proposed method.

 ISBN: 978-1-61208-186-1

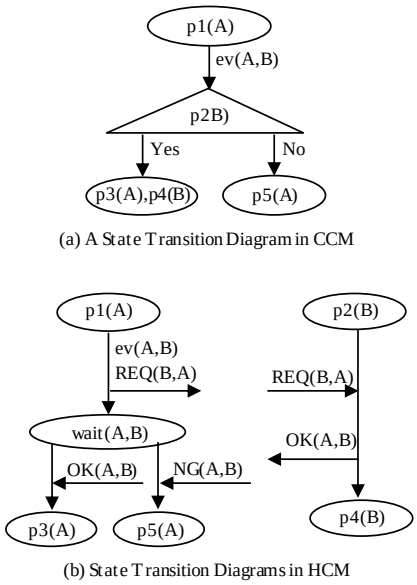

Figure 8. When a State Primitive is Described

2) In the case of a condition primitive

The analysis block is described just below the current state in the HCM corresponding to the first argument of the condition primitives. This current state of the HCM corresponds to that of the CCM. When the condition described in the analysis block is satisfied, OK is described as a reply signal to the HCM that sent REQ. When the condition described in the analysis block is not satisfied, the conversion is done in the following two cases: If, in CCM, the state of the terminal which received REQ does not change, in HCM, a transition to the current state is described. Otherwise, in HCM, a transition to wait state is described.

Fig. 9 shows an example of converting a state transition diagram, where a condition primitive pc(B) is described in the analysis block, from CCM to HCM. When terminal B is not in the state of {p2(B)}, NG1 is sent to terminal A. But, in HCM of terminal B, since NG1 is sent from the state analysis program NG1 does not appear in HCM of terminal B in Fig. 9.

3) In the case of both primitives

When both primitives, state primitives and condition ones, are described in the analysis block, the analysis block is converted in different ways depending on whether the both primitives have the same argument or not. If the both primitives do not have the same argument, the analysis block is described in the same way as mentioned in *1)* and *2)* above. If both primitives have the same argument, the conversion is carried out in two stages. First, REQ as an input signal is described just below the state consisting of the same state primitives described in the analysis block. Second, the analysis block is described just below the input signal, and the condition primitives, which have the argument corresponding to the HCM, are described in the analysis block. Input and output signals are described as described in *1)* and *2)*.

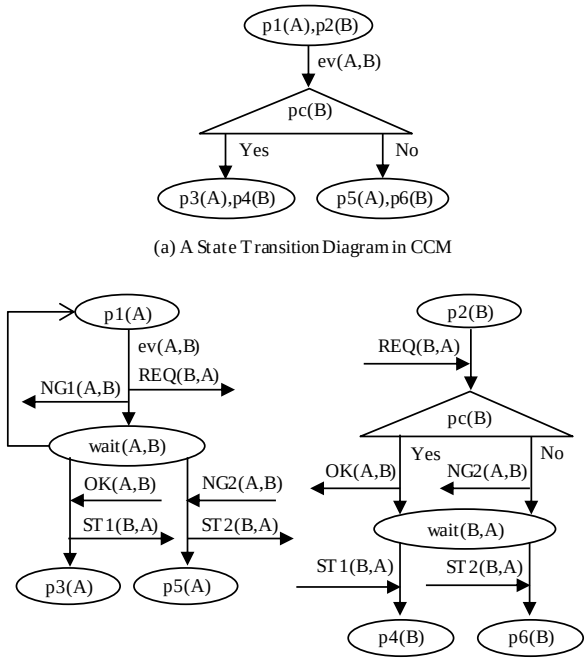

Figure 9. When a Condition Primitive is Described

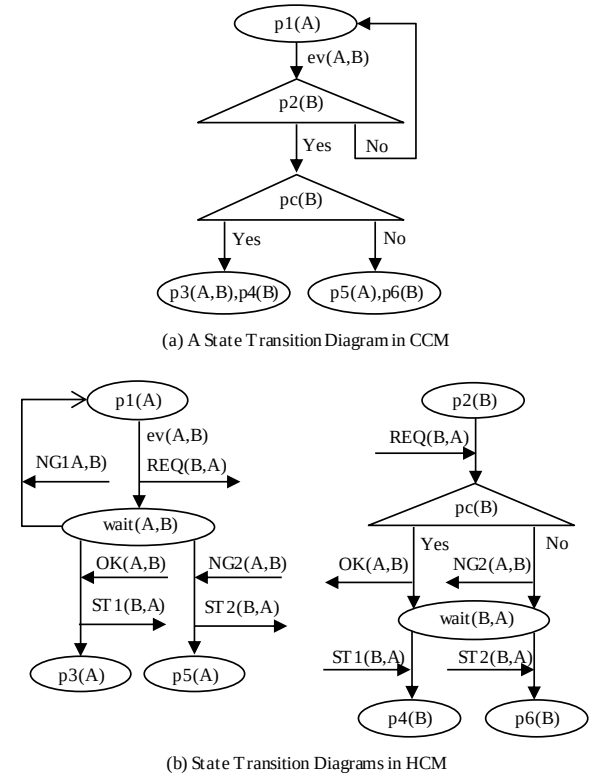

Figure 10. When State and Condition Primitives are Described

Fig. 10 shows an example of converting a state transition diagram, where a state primitive p2(B) and a condition

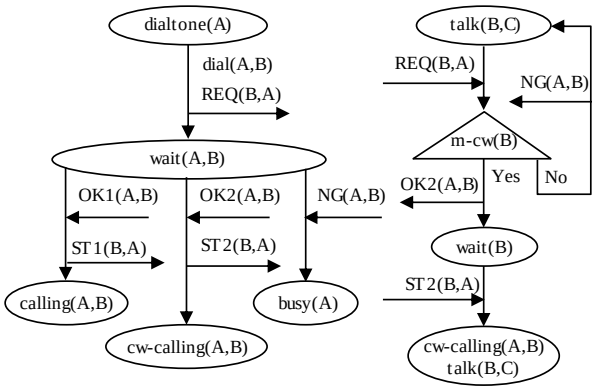

Figure 11. State Transition Diagrams in HCM (Call-Waiting Service)

primitive pc(B) are described in the analysis block, from CCM to HCM. When terminal B is not in the state of {p2(B)}, NG1 is sent to terminal A. But, in HCM of terminal B, since NG1 is sent from the state analysis program NG1 does not appear in HCM of terminal B in Fig. 10.

Fig. 11 shows the HCMs converted from the CCM of a call waiting service described in Fig. 1. Since, in Fig. 1, only the state of terminal A is described in the current state of CCM, the synchronization process described in *B* and *C* in this section is not described. OK1 is sent to terminal A when terminal B is in idle state. Namely this is a state transition for POTS. So, OK1 does not appear in HCM of terminal B in Fig. 11.

V. An Example of Conversion of State Transitions

An example of conversion for the three-way call service based on the method described in Section IV is shown.

A. State Transitions for Three-Way Call Service

The main part of state transitions for the three-way call service is described. Fig. 12 shows state transitions from two-party talk state (talk(A,B)) to three-way talk state (twctalk(A,B,C)) in the form of text.

Condition primitive m-twc(A) represents that terminal A subscribes to the three-way call service, flash(A) represents that flash button is pressed, hold(A,B) represents that terminal A holds terminal B, dialtone(A) represents that terminal A receives dial-tone, idle(C) represents that terminal C is in idle state, calling(A,C) represents that terminal A is calling terminal C, not[idle(A)] represents that terminal C is not in idle state, busy(A) represents that terminal A receives busy tone, dial(A,C) represents that a user of terminal A dials the number of terminal C, offhook(C) represents tresents that a user of terminal C hangs up the receiver.

```
S1 -> S2: {talk(A,B)} flash(A): {m-twc(A)} {hold(A,B),dialtone(A)}
S2 -> S3: {hold(A,B),dialtone(A)} dial(A,C): {idle(C)}
                    {hold(A,B),calling(A,C)}
S2 -> S5: {hold(A,B),dialtone(A)} dial(A,C): {not[idle(C)]}
                    {hold(A,B),busy(A)}
S3 -> S4: {hold(A,B),calling(A,C)} offhook(C): {} hold(A,B),talk(A,C)}
S4 -> S5: {hold(A,B),talk(A,C)} onhook(C): {} {hold(A,B),busy(A)}
S5 -> S1: {hold(A,B),busy(A)} flash(A): {} {talk(A,B)}
S4 -> S6: {hold(A,B),talk(A,C)} flash(A): {} {twctalk(A,B,C)}
          Syntax: {current state} event: {analysis block} {next state}
```

Figure 12. State Transitions of a Three-Way Call Service in CCM

The states of CCM described in Fig. 12 are the following six states. Note that S1 is a state of POTS (plain old telephone service).

S1 = {talk(A,B)},
S2 = {hold(A,B),dialtone(A)},
S3 = {hold(A,B),calling(A,C)},
S4 = {hold(A,B),talk(A,C)},
S5 = {hold(A,B),busy(A)},
S6 = {twctalk(A,B,C)}.

B. Conversion of Current States in HCM

Fig. 13 shows the converted current states in the HCM for terminal A, B and C from the current states S1 … S5 described in the CCM as mentioned in Section IV *B.*

C. Conversion to Tree Representation

Tree representations obtained from the current states S1 … S5 described in Section IV *B* are shown in Fig. 14. In the representation, the root represents the terminal where an event occurs. Terminal A, B, and C are represented as nodes.

D. Conversion of State Transitions

The state transition diagrams in HCM converted from the state transition diagram in CCM described in Section V *B* are shown in Fig. 15. State transitions for synchronization are omitted.

Since there are various types of state transitions in Fig. 12, in addition to processing of conversion of states and synchronization of states, some conversion methods described in Section V are applied.

a) In the state transition S1 -> S2: Since a condition primitive m-twc(A) is described in the analysis block, the conversion method described in IV *D 2)* is applied.

	A	B	C
S1:	talk(A,B)	talk(A,B)	-
S2:	dialtone(A),hold(A,B)	hold(A,B)	-
S3:	hold(A,B),calling(A,C)	hold(A,B)	calling(A,C)
S4:	hold(A,B),talk(A,C)		talk(A,C)
S5:	hold(A,B),busy(A)	hold(A,B)	-

Figure 13. Converted States in the HCM for Terminal A, B and C

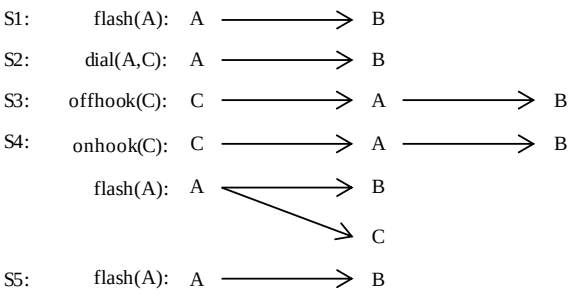

Figure 14. Tree Representation Obtained from States S1 … S5

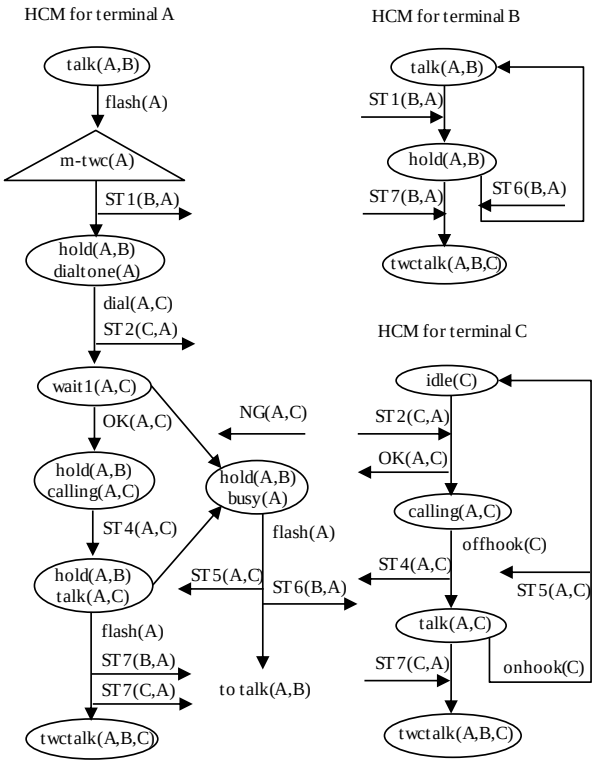

Figure 15. State Transition Diagrams in HCM Converted from CCM

b) In the state transition S2 -> S3: Since a state primitive idle(C) is described in the analysis block, the conversion method for the signal OK described in IV *D 1)* is applied.

c) In the state transition S2 -> S5: A state primitive not[idle(C)] which means the condition idle(C) is not satisfied is described in the analysis block. So, the conversion method for NG described in IV *D 1)* is applied.

d) In the state transition S4 -> S5: A branch appears in the tree expression for S4. So, the method of processing signals at the branch described in IV *B 2)* is applied.

Each state transition in Fig. 12 is converted to state transitions in HCM's in Fig. 15. State transition S1 -> S2 in Fig. 12 is converted to the state transition from talk(A,B) to {hold(A,B), dialtone(A)} in HCM for terminal A, and the state transition from talk(A,B) to hold(A,B) in HCM for terminal B, respectively. Suppose terminal A is in talk(A,B) state. At this moment, when an event flash(A) occurs and terminal A subscribes three-way call service, m-twc(A), signal ST1(B,A), which is an instruction to transit to hold(A,B), is sent to HCM for terminal B. In the HCM for terminal B, the state of terminal B transits to hold(A,B) if and only if ST1(B,A) is received when the state of terminal B is in at talk(A,B). Other six state transitions in Fig. 12 are also converted to state transition diagrams described in Fig. 15.

Thus, the proposed method can convert all types of state transitions in the three-way call service. Consequently, the proposed methods can give a good prospect for converting state transition diagrams of various services from CCM to HCM.

VI. Conclusion and Future Work

A conversion method from CCM to HCM was proposed, and by applying the method to the three-way call service it was confirmed that the proposed method can properly convert the state transition diagrams. Consequently, the proposed methods can give a good prospect for converting state transition diagrams of various services from CCM to HCM. Future work is to verify the validity of the proposed method by applying the method to various services including non-telephone services.

References

[1] H. Kawashima, K. Futami, and S. Kano, "Functional specification of call processing by state transition diagram," IEEE Trans. on Com., vol. COM-19, 1971.

[2] N. D. Grifeth, R. Blumenthal, J. C. Gregoire, and T. Ohta, "Feature interaction detection contest of the fifth international workshop on feature interactions," Computer Networks, vol. 32, pp. 487-510, Apr. 2000.

[3] A. Nakashima and T. Ohta, "Automatic conversion from HCM to CCM in telecommunication service simulation," Proc. ATS04, pp. 29-34, Apr. 2004.

[4] N. D. Griffeth, Y. Cantor, and C. Djouvas, "Testing a Network by Inferring Representative State Machines from Network Traces," ICSEA2006, Nov. 2006.

[5] S. K. Chakrabarti, and Y. N. Srikant, "Specification Based Regression Testing Using Explicit State Space Enumeration," ICSEA2006, Nov. 2006.

[6] M. Shimokura, S. Nakanishi and T. Ohta, "Networks for a symbiotic Human Life with Robots," Proc. of ROBOCOM2007, Oct. 2007.

[7] S. Sen and R. Cardell-Oliver, "A Rule-Based Language for Programming Wireless Sensor Actuator Networks using Frequency and Communication," Third Workshop on Embedded Networked Sensors (EmNets 2006), 2006.

[8] X. Fei and E. Magill, "Rule Execution and Event Distribution Middleware for PROSEN-WSN," 2008 Second International Conference on Sensor Technologies and Applications, pp. 580-585, Aug. 2008.

[9] L. Dongliang, Z. Kanyu, and L. Xiaojing, "ECA Rule-based IO Agent Framework for Greenhouse Control System," ISCID, 2008 International Symposium on Computational Intelligence and Design, vol. 1, pp. 482-485, Oct. 2008.

[10] M. Ohba, K. Matsuoka, and T. Ohta, "Eliciting State Transition Diagrams from Programs described in a Rule-based Language," ISAST Transaction on Computers and Intelligent Systems, No. 2, Vol. 1, pp. 58-66, Jan. 2009.

[11] Y. Hirakawa and T. Takenaka., "Telecommunication service description using state transition rule," Int. Workshop on Software Specification and Design, Oct. 1991.

[12] T. Yoneda and T. Ohta, "The declarative language STR," Proc. of FIREworks workshop, pp. 197-212, May 2000.

[13] A. Takura, T. Ohta, and K. Kawata, "Process specification generation from communications service specification," Automated Software Eng., No. 2, pp. 167-182,1995.

 ISBN: 978-1-61208-186-1