A Verification Procedure via Invariant for Extended Communicating Finite-State Machines

Masahiro Higuchi* Osamu Shirakawa* Hiroyuki Seki*

Mamoru Fujii** Tadao Kasami***

* Dept. of Information and Computer Sciences, Osaka University
Toyonaka, Osaka 560, Japan

** College of General Education, Osaka University
Toyonaka, Osaka 560, Japan

*** Advanced Institute of Science and Technology, Nara
Ikoma, Nara 630–01, Japan

e-mail: (higuchi, sirakawa, seki, fujii, kasami)@ics.osaka-u.ac.jp

Abstract. This paper presents a method for verifying safety property of a communication protocol modeled as two extended communicating finite-state machines with two unbounded FIFO channels connecting them. In this method, four types of atomic formulae specifying a condition on a machine and a condition on a sequence of messages in a channel are introduced. A human verifier describes a logical formula which expresses conditions expected to be satisfied by all reachable global states, and a verification system proves that the formula is indeed satisfied by such states (i.e. the formula is an invariant) by induction. If the invariant is never satisfied in any unsafe state, it can be concluded that the protocol is safe. To show the effectiveness of this method, a sample protocol extracted from the data transfer phase of the OSI session protocol was verified by using the verification system.

1 Introduction

For implementing reliable communication software, it is important to verify the communication protocol formally. Communicating finite-state machines (CFSMs) are used as a model for verifying communication protocols. If the boundedness of the communication channels is guaranteed, many important properties for CFSMs are decidable[1] in principle, and some decision procedures have been proposed[2][3]. However, even though channel boundedness is guaranteed, the decision procedures based on channel boundedness are not feasible for most practical protocols because of state space explosion.

Furthermore, for practical protocols, protocol machines are usually defined as extended communicating finite-state machine(ECFSM)s whose state is rep-

resented by a state of finite control and values of context variables. In fact, two formal description techniques Estelle[4] and SDL[5] for communication protocols are based on extended finite-state machine model. In this paper, a verification method for a class of ECFSMs in which the channel boundedness is not guaranteed is proposed.

For such a class of protocols, the set of global states reachable from the initial global state is potentially infinite and therefore traditional state exploration techniques which enumerate reachable global states cannot be used. Instead, we propose a method based on a verification via invariant using similar techniques to those adopted by such systems as theorem provers[6].

The proposed method is summarized as the following (1) and (2):

(1) Find a logical formula on global states, say F, which is expected to satisfy (a) $RS \subseteq GS(F)$ and (b) $GS(F) \subseteq SAFE$, where RS is the set of reachable global states, $GS(F)$ is the set of those global states which satisfy F and $SAFE$ is the set of safe global states. Although only safety property is considered in this paper, the proposed method can be extended to verify liveness property. F is written as a propositional formula which consists of the following atomic formulae,
   (i) conditions on states of finite controls of ECFSM,
   (ii) regular expressions which specify message type sequences in the channels,
   (iii) conditions on sequences of integers (parameters of messages in the channel) such as 'monotonically increasing', and
   (iv) linear inequalities on integers which specify the relations to hold for the values of context variables of ECFSMs and parameters of messages in the channels.

(2) Verify that the above (a) and (b) hold. Verification of (b) is easy. The above (a) is verified by structural induction on event sequences. Verification in the inductive step is reduced to the inclusion problem for given two regular expressions (for (ii) above), the problem to find the normal form of a given term in the term rewriting system which represents the definition of protocol machines, inductive hypothesis and properties of sequences of integers (for (iii)), and the problem to decide whether a given ordered pair of expressions belongs to the transitive closure of given inequalities (for (iv)).

A verification system which implements the proposed procedure and a verification example of OSI session protocol are also described in this paper.

As related works, a verification method is proposed for ECFSMs with queues of length one[7]. For a protocol for which channel boundedness is not guaranteed, Finkel[8] studies a class of protocols in which the set of message sequences in the channels is exactly expressed by regular expressions, and gives decidable results on some verification problems. However, protocol machines considered in [8] are assumed to be finite. The protocol model discussed in this paper assumes neither finiteness of protocol machines nor channel boundedness. Systematic verification methods for such a class of protocols have been scarcely reported.

2 Basic Definitions

2.1 Protocol Model

Two-extended communicating finite-state machines(2-ECFSMs) are a protocol model which consists of two protocol machines modeled as extended communicating finite-state machines and two unbounded FIFO channels connecting them. Formally, it is defined as below.

A **protocol machine** PM is a 4-tuple (S, Σ, δ, si), where

(M1) $S = \langle SF, r \rangle$ defines a set of **states**, where SF is the state set of finite control part of the machine and r is the number of registers(context variables) which store nonnegative integers. Let $\mathcal{N}$ denote the set of nonnegative integers. The state space of the protocol machine is $SF \times \mathcal{N}^r$.

(M2) $\Sigma = \Sigma_- \cup \Sigma_+$: a finite set of **message types**. Σ_- is a set of message types which PM can send and Σ_+ is a set of message types which PM can receive. Σ_- and Σ_+ are supposed to be disjoint. For $d \in \Sigma$ and $n \in \mathcal{N}$, $\langle d, n \rangle$ is called a **message** and n is called the parameter of the message. The number of parameters of a message is assumed to be exactly one only for simplicity. In the following, for a message sequence u, $type(u)$ and $parameter(u)$ denote the message type sequence of u and the parameter sequence of u respectively. The set of events EV of the protocol machine is defined in connection with the set of messages sent or received by the machine, i.e. $EV = \{-\langle d, n \rangle | d \in \Sigma_-, n \in \mathcal{N}\} \cup \{+\langle d, n \rangle | d \in \Sigma_+, n \in \mathcal{N}\}$. The former subset is the set of **sending events** and the latter one is the set of **receiving events.**

(M3) δ : a partial **state transition function** from $SF \times \mathcal{N}^r \times EV$ to $SF \times \mathcal{N}^r$. For $s \in SF \times \mathcal{N}^r$ and $e \in EV$, if $\delta(s, e)$ is defined, then an event e is said to be **executable** in the state s.

(M4) $si \in SF \times \mathcal{N}^r$: an **initial state.**

For $PM_A = (\langle SF_A, r_A \rangle, \Sigma_A, \delta_A, si_A)$ and $PM_B = (\langle SF_B, r_B \rangle, \Sigma_B, \delta_B, si_B)$, if $\Sigma_{B-} = \Sigma_{A+}$ (denoted Σ_{BA}) and $\Sigma_{A-} = \Sigma_{B+}$ (denoted Σ_{AB}), then $\Pi = (PM_A, PM_B)$ is called a **protocol**. A 4-tuple $(s_A, s_B, ch_{BA}, ch_{AB}) \in (SF_A \times \mathcal{N}^{r_A}, SF_B \times \mathcal{N}^{r_B}, \langle \Sigma_{BA}, \mathcal{N} \rangle^*, \langle \Sigma_{AB}, \mathcal{N} \rangle^*)$ is called a **global state** of protocol Π. s_A and s_B denote states of PM_A and PM_B respectively. ch_{BA} and ch_{AB} denote message sequences in the channel from PM_B to PM_A and that from PM_A to PM_B respectively. $gs_I = (si_A, si_B, \varepsilon, \varepsilon)$ (ε is the empty sequence) is called the **initial global state** of Π.

A global state $gs' = (s'_A, s'_B, ch'_{BA}, ch'_{AB})$ is said to be **transitable** from $gs = (s_A, s_B, ch_{BA}, ch_{AB})$(denoted by $gs \rightarrow gs'$) iff one of the following conditions is satisfied for some $d \in \Sigma_{BA} \cup \Sigma_{AB}$ and $n \in \mathcal{N}$:

(TA1) $s'_A = \delta_A(s_A, -\langle d, n \rangle)$, $s'_B = s_B$, $ch'_{BA} = ch_{BA}$, $ch'_{AB} = ch_{AB} \cdot \langle d, n \rangle$;
(TA2) $s'_A = \delta_A(s_A, +\langle d, n \rangle)$, $s'_B = s_B$, $\langle d, n \rangle \cdot ch'_{BA} = ch_{BA}$, $ch'_{AB} = ch_{AB}$;
(TA3) $s'_A = s_A$, $s'_B = \delta_B(s_B, -\langle d, n \rangle)$, $ch'_{BA} = ch_{BA} \cdot \langle d, n \rangle$, $ch'_{AB} = ch_{AB}$;
(TA4) $s'_A = s_A$, $s'_B = \delta_B(s_B, +\langle d, n \rangle)$, $ch'_{BA} = ch_{BA}$, $\langle d, n \rangle \cdot ch'_{AB} = ch_{AB}$.

If (TA1) holds, the relation is also denoted $gs-(-\langle d,n\rangle, A) \rightarrow gs'$. This extended notation is also used for (TA2), (TA3) and (TA4). The transitive reflexive closure of the relation "$\rightarrow$" is denoted by "$\stackrel{*}{\rightarrow}$". If $gs \stackrel{*}{\rightarrow} gs'$, then the global state gs' is said to be **reachable** from gs.

2.2 Safety Property

For a protocol $\Pi = (PM_A, PM_B)$, the set of reachable global states from the initial global state is called the **reachability set** of Π. If the reachability set of Π does not contain following unsafe states, Π is said to be **safe**.

Deadlock state: A global state $gs = (s_A, s_B, ch_{BA}, ch_{AB})$ is said to be a deadlock state if $ch_{BA} = ch_{AB} = \varepsilon$ and any sending event is not executable in s_A and s_B respectively.

Unspecified reception state: A global state $gs = (s_A, s_B, ch_{BA}, ch_{AB})$ is said to be an unspecified reception state if either $ch_{BA} \neq \varepsilon$ and $\delta_A(s_A, +head(ch_{BA}))$ is not defined or $ch_{AB} \neq \varepsilon$ and $\delta_B(s_B, +head(ch_{AB}))$ is not defined, where $head(\alpha)$ denotes the first element of a nonempty sequence α.

3 Verification Method

If a logical formula F on global state on Π is satisfied by all global states in the reachability set of a protocol Π, F is called an **invariant** in Π. If an invariant F in Π is not satisfied by any deadlock state or unspecified reception state, then Π is safe. We present a method for verifying a given formula in a disjunctive normal form $F = P_1 \vee P_2 \vee \ldots \vee P_n$ to be an invariant in Π. In the following, for a formula F, $GS(F)$ denotes the set of global states which satisfy F.

3.1 Description of a Logical Formula

Every disjunct P_i of formula F is a conjunction of **atomic formulae** (or simply **atoms**) of the following four types. Figure 1-A (a) shows an example of P_i.

(AF1) A formula $\langle SSF_A, SSF_B\rangle$, where $SSF_A \subseteq SF_A$ and $SSF_B \subseteq SF_B$, is an atom which holds for a global state $(s_A, s_B, ch_{BA}, ch_{AB})$ iff the finite control part of s_A and s_B belong to SSF_A and SSF_B respectively.

(AF2) A class of regular expression to express an infinite set of message type sequences in a communication channel is introduced as below. The regular expression is restricted to be ε (the empty sequence) or a concatenation of subexpressions of the following types:

R1: A choice of Σ_{BA} (or Σ_{AB}), i.e. $m_1 + m_2 + \ldots + m_n$ for $m_k (1 \leq k \leq n)$ in Σ_{BA} (or Σ_{AB});

R2: Positive closure t^+ of a choice t of Σ_{BA} or Σ_{AB}.

For two restricted regular expressions r_{BA} and r_{AB}, a formula $\langle r_{BA}, r_{AB}\rangle$ is an atom which holds for a global state $(s_A, s_B, ch_{BA}, ch_{AB})$ iff $type(ch_{BA}) \in L(r_{BA})$ and $type(ch_{AB}) \in L(r_{AB})$, where $L(r)$ is the set of sequences denoted

by the regular expression r. We assume that exactly one AF2 type atom appears in every P_i. In the following, if a global state $(s_A, s_B, ch_{BA}, ch_{AB})$ satisfies an AF2 type atom "$\langle u_1 \cdot u_2 \cdot \ldots \cdot u_n,\ v_1 \cdot v_2 \cdot \ldots \cdot v_m \rangle$", $BA[k](1 \leq k \leq n)$ denotes a message sequence such that $BA[1] \cdot BA[2] \cdot \ldots \cdot BA[n] = ch_{BA}$ and $type(BA[j]) \in L(u_j)(1 \leq j \leq n)$ and, $AB[k](1 \leq k \leq m)$ denotes the message sequence such that $AB[1] \cdot AB[2] \cdot \ldots \cdot AB[m] = ch_{AB}$ and $type(AB[j]) \in L(v_j)(1 \leq j \leq m)$.

(AF3) A predicate on a message sequence on a communication channel is also an atom. For instance, "$step1(AB[1])$" (in Figure 1-A (a)) states that the parameter sequence $parameter(AB[1])$ satisfies the predicate "$step1$". "$step1$" means that the parameter sequence is an increasing sequence such that the difference of every adjacent elements is one. Predicates which appear in an AF3 type atom are defined in terms of rewrite rules and inequalities. For example, a conditional rewrite rule in Figure 1-A (b) "$|seq| \geq 1, step1(seq), t = last(seq)+1 : step1(seq \cdot \langle type, t \rangle) \Rightarrow true$" asserts that if a message sequence seq of length 1 or more satisfies the predicate $step1$ and t is equal to the parameter of the last message of seq plus 1, then the parameter sequence $parameter(seq) \cdot t$ of the message sequence $seq \cdot \langle type, t \rangle$ also satisfies the predicate $step1$. The conditions of conditional rewrite rules and inequalities are also assumed to be written in the form of an AF3 or AF4 type atom.

(AF4) A linear inequality which represents the relation on the values of registers of protocol machines and the parameter values of messages in a channel. The expressions appearing on the both sides of inequalities are restricted to the form of "$v+C$" where v is a term which denotes either the value of a register of a protocol machine or a parameter value of a message in a channel and C is a constant value of integer. For instance, "$Vm(A) = last(AB[1]) + 1$" in Figure 1-A (a) is an AF4 type atom which states that the value of the register Vm of PM_A is equal to the parameter of the last message in the channel from PM_A to PM_B plus 1 at the global state under consideration.

3.2 Verification Procedure

A given logical formula $F = P_1 \vee P_2 \vee \ldots \vee P_n$ is shown to be an invariant in Π by structural induction on event sequences of Π as follows.

Inductive basis: Prove that the initial global state of Π satisfies F.

Inductive step: Prove that

$$\forall_{gs \in GS(F)} \forall_{gs \rightarrow gs'} \{gs' \in GS(F)\}. \qquad (* 1)$$

Observe that $GS(F) = GS(P_1) \cup GS(P_2) \cup \ldots \cup GS(P_n)$. Therefore, (* 1) is equivalent to

$$\forall_{i(1 \leq i \leq n)} \forall_{gs \in GS(P_i)} \forall_{gs \rightarrow gs'} \exists_{j(1 \leq j \leq n)} \{gs' \in GS(P_j)\}. \qquad (* 2)$$

Thus (* 1) is proved by executing the following IS1 and IS2 for each $P_i(1 \leq i \leq n)$.

IS1 Identify all events(pairs of a local event $(\pm\langle d, n\rangle)$ and a machine) executable in global states in $GS(P_i)$.

IS2 For every executable event $(\pm\langle d, n\rangle, X)$ obtained by IS1, show

$$\forall_{gs\in GS(P_i)}\forall_{gs-(\pm\langle d,n\rangle,X)\rightarrow gs'}\exists_{j(1\leq j\leq n)}\{gs' \in GS(P_j)\}. \qquad (*\ 3)$$

The inductive basis and IS1 are easily examined from the form of each P_i. To explain the procedure for examining IS2, we consider the following example $EX1$.

$\underline{EX1}$: • Let P_i in (*3) be shown in Figure 1-A (a).
• Let $(\pm\langle d, n\rangle, X)$ in (*3) be $(-\langle \mathrm{MIP}, Vm(A)\rangle, A)$. □

The definition of state transition on the event is shown in Figure 1-A (b). In the following and in Figures 1-A and 1-B, all terms with primes denote the values of the corresponding terms without primes after the transition. The definition of the state transition tells the followings:

(1) If the state of finite control of the machine is STA713, the machine can send MIP with parameter value equal to the value of register Vm;
(2) The finite control still stays at STA713 after sending $\langle \mathrm{MIP}, Vm(A)\rangle$;
(3) The value of register Vm is incremented by one;
(4) The value of register Va is not changed.

Let $P_j = PF_j \wedge PI_j (1 \leq j \leq n)$, where PF_j is the conjunction of AF1 and AF2 type atoms and PI_j is the conjunction of AF3 and AF4 type atoms. Since $GS(P_j) = GS(PF_j) \cap GS(PI_j)$, IS2 is refined as the following (I) and (II) for each i.

(I) Identify all PF_j such that

$$\forall_{gs\in GS(PF_i),gs-(\pm\langle d,n\rangle,X)\rightarrow gs'}\{gs' \in GS(PF_j)\}. \qquad (*\ 4)$$

In general, it can be checked for gs' to satisfy an AF1 type atom from the definition of δ directly, and to satisfy an AF2 type atom by reducing the satisfaction problem to the inclusion problem for two regular sets. The restrictions **R1** and **R2** simplify the decision procedure for this inclusion problem. Consider the case $P_j = P_i$ in $EX1$. As the message type sequence in the channel from PM_B to PM_A and that from PM_A to PM_B at gs' are required to be in $L(\varepsilon)$ and $L(\mathrm{MIP}^+ \cdot \mathrm{MIP})$ respectively, the problem to decide whether every gs' satisfies $\langle \varepsilon, \mathrm{MIP}^+\rangle$ is reduced to the inclusion problems $L(\varepsilon) \subseteq L(\varepsilon)$ and $L(\mathrm{MIP}^+ \cdot \mathrm{MIP}) \subseteq L(\mathrm{MIP}^+)$.

Then the next step is as follows:

(II) Show that $gs' \in GS(PI_j)$ for some j which satisfies (*4).

To show (II),at first, the rewrite rules to express the message sequences in gs' in terms of the message sequences in gs and the message sent by executing the event are generated.

For example, suppose that the event is $-(\langle d, p\rangle, A)$. The rewrite rules are generated as follows. Let $\langle r_{BA,i}, r_{AB,i}\rangle$ and $\langle r_{BA,j}, r_{AB,j}\rangle$ be AF2 type atoms in P_i and P_j respectively. Let $r_{AB,i} = u_1 \cdot u_2 \cdot \ldots \cdot u_n$ and $r_{AB,j} = u'_1 \cdot u'_2 \cdot \ldots \cdot u'_m$, where u_k and u'_l are choices of Σ_{AB} or positive closures of choices of Σ_{AB}, for $1 \leq k \leq n$ and $1 \leq l \leq m$. Since the atom $\langle r_{BA,j}, r_{AB,j}\rangle$ holds for every gs' by (I), i.e. $L(r_{AB,i} \cdot d) \subseteq L(r_{AB,j})$, it follows from the restrictions **R1** and **R2** that there exists a mapping φ such that $L(u_1 \cdot \ldots \cdot u_{\varphi(l)}) \subseteq L(u'_1 \cdot \ldots \cdot u'_l)$ and $\varphi(l-1) \leq \varphi(l)$ for every $1 \leq l \leq m-1$, and $\varphi(0) = 0$. Then the rewrite rule "$AB[l]' \Rightarrow AB[\varphi(l-1)+1] \cdot \ldots \cdot AB[\varphi(l)]$" is generated for $1 \leq l \leq m-1$ and the rewrite rule "$AB[m]' \Rightarrow AB[\varphi(m-1)+1] \cdot \ldots \cdot AB[n] \cdot \langle d, p\rangle$" is generated. In our example, "$AB[1]' \Rightarrow AB[1] \cdot \langle \mathrm{MIP}, Vm(A)\rangle$" is generated. If the mapping φ is not uniquely determined, then for every possible φ, the rewrite rules for φ are generated and the procedure to check (II) is executed. If (II) holds for some φ, it can be concluded that IS2 for given $(-\langle d, n\rangle, A)$ holds.

To check (II), the condition parts of all conditional rewrite rules and conditional inequalities are evaluated with assigning the values in gs to the free variables in the condition. If the condition of a conditional rewrite rule (or inequality) is shown to be true for the assignment, then the rewrite rule (or inequality) instantiated by the assignment is added to the assumption. In our example, $|AB[1]| \geq 1$, $step1(AB[1])$, and $Vm(A) = last(AB[1]) + 1$ are shown to be true, and the rewrite rule
"$step1(AB[1] \cdot \langle \mathrm{MIP}, Vm(A)\rangle) \Rightarrow true$" are added as (5'). The procedures for evaluating AF3 and AF4 type atoms (and the conditions of conditional rewrite rules and conditional inequalities) are described below.

- (AF3) Show that the atom can be rewritten as constant term "true" under the term rewriting system[9] consisting of assumed relations. Figure 1-A (b) shows the example. Figure 1-B (d) shows a process in which "$step1(AB[1]')$" is rewritten to "true".
- (AF4) For an atom "$a\ rel\ b$" ($rel \in \{=, \geq\}$), find the normal forms of a and b under the rewriting system described above, i.e. rewrite a and b to $norm(a)$ and $norm(b)$ respectively until $norm(a)$ and $norm(b)$ can not be rewritten to any terms. In Figure 1-B (e), "$Vm'(A) \geq Va'(A)$" is rewritten to "$Vm(A)+1 \geq Va(A)$". And decide whether $norm(a)\ rel\ norm(b)$ belongs to the transitive closure (over the set of expressions of the form "$v + C$") of the assumed inequalities (Figure 1-B (c)). In our example, let $GE = \{(a, b) \mid a \geq b$ is an assumed inequality$\}$, then $(Vm(A)+1, Vm(A))$ and $(Vm(A), Va(A))$ belong to GE. The transitive closure of GE contains $(Vm(A)+1, Va(A))$. We can conclude that "$Vm(A)+1 \geq Va(A)$"(Figure 1-B (f)).

4 A Verification System

We implemented a verification system based on the verification method described in Section 3. The verification system provides the procedures for executing a state transition, deciding the inclusion problem on given two regular sets, rewriting a term under the given term rewriting system, and deciding whether a given pair

of expressions belong to the transitive closure of given relations. The system executes the inductive step of the proposed verification method by conducting the above procedures.

An input to the verification system consists of the definition of protocol machines, properties of predicates on sequences of integers such as *step*1 explained in the example in Section 3, and a logical formula F to be shown an invariant. The system constructs a state transition table, unfolds an input formula to obtain a disjunctive normal form, and executes the verification procedure described in Section 3. If there exists a pair of global states gs and gs' such that $gs \rightarrow gs'$, $gs \in GS(F)$ and $gs' \notin GS(F)$, then the system always detects the fact and reports relevant information on such global states and a transition. If there exists a deadlock or unspecified reception state which satisfies F, then the system also detects that and reports it.

The verification system was implemented by using C, lex, and yacc on the UNIX environment. The size of the source code of the system is about 10,000 lines.

5 An Experimental Result

To show the usefulness of the proposed verification method, we performed an experiment on a part of OSI session protocol[10].

5.1 Extracting a Sample Protocol

We extract the protocol for *data transfer* phase of *kernel, duplex, minor synchronize* and *major synchronize* functional units from OSI session protocol. For simplification, we omit some *PDU*(message)s which have no effect on any registers and we assume that the rights to send *MIP*(*MINOR SYNC POINT*) and *MAP*(*MAJOR SYNC POINT*) are transferred simultaneously from a protocol machine to the other machine by sending a token named *ma-mi* token instead of using two tokens *ma* and *mi*. For the extracted protocol $\Pi_{ses} = (PM_{ses_A}, PM_{ses_B})$, PM_{ses_A} and PM_{ses_B} are the same protocol machines except their initial states, i.e. PM_{ses_A} owns *ma-mi* token while PM_{ses_B} does not at the initial states. The size of the states of finite control and the number of registers of the protocol machine are 10 and 2 respectively. And the number of message types used in the protocol is 10.

5.2 Verification Result

For the protocol Π_{ses}, the set of global states expected to be reachable has been divided into 60 subsets by considering the possible combinations of the states of finite control parts of two protocol machines. We have described a logical formula based on these subsets of global states. The numbers of AF1 through AF4 type atoms in the described formula are 180, 60 46, and 297 respectively.

The properties on sequences of integers used for verification were provided in terms of 10 conditional rewrite rules and 7 conditional inequalities.

In the process of describing a formula, a human verifier often misses some reachable global state and describes an incorrect formula, i.e. a described formula is not an invariant in Π_{ses}. In such a case, the verification system detects a global state which does not satisfy the described formula and reports relevant information about the global state. A human verifier revised the formula using the information reported by the verification system.

The described formula was verified to be an invariant and the protocol was verified to be safe by the verification system on a UNIX workstation (Solbourne Series 5/600, 2CPU 48MB). The CPU time and memory storage used in the execution are 12.0 seconds and 816 KBytes respectively. The input formula was expanded to a disjunctive normal form consisting of 170 conjunctive terms. The number of considered state transitions was 428, and the number of AF3 and AF4 type atoms checked to hold in some global states were 222 and 1600 respectively.

6 Discussion

It seems that in a verification procedure via invariant for extended communicating finite-state machines, how to cope with integral registers dominates its verification power and efficiency. In this paper, by restricting the expressions appearing on both sides of inequalities to the form of "$v + C$", the problem whether a given inequality is implied by a given set of inequalities can be decided by an efficient procedure. In most of practical protocols, the operations on integral registers in the definition of a state transition function of a protocol machine are limited to simple types, e.g. "store some value", "add a constant number to the current value", or "clear to 0", and the restriction on the form of inequalities does not affect the verification power on such protocols. If more general operations on integral registers e.g. "summation of register values" are used in a protocol, then a more general procedure, e.g. the decision procedure for Presburger formula which is known to be intractable in general, may be required to deal with such a protocol.

In this paper, we put a restriction on regular expression. This greatly simplifies the procedure for deciding whether a given regular set includes another regular set. We are extending the verification system to allow the following interleaving operator " $\|$ " on regular expressions without loss of simplicity. For regular expressions r_1 and r_2,
$L(r_1 \| r_2) = \{w_1x_1w_2x_2...w_kx_k \mid w_1w_2...w_k \in L(r_1)$ and $x_1x_2...x_k \in L(r_2)\}$.

As a practical protocol such as OSI session protocol provides many kinds of services, the definitions of protocols tend to be enormous and any verification method suffers from state space explosion. To facilitate the design and analysis of such a protocol, the authors have proposed a method for composing a safe protocol from a safe protocol defining a priority service and that defining an ordinary service[11]. Furthermore, several composition techniques have been proposed within the framework of CFSM[12]~[14]. It is desirable to fit these

techniques to the protocol model discussed in this paper. Currently, we are conducting an experiment to show the safety property of OSI session protocol using these techniques.

References

1. Brand D., and Zafiropulo P.: "On Communicating Finite-State Machines", Journal of ACM, vol.30, pp.323-342, 1983-04.
2. Kakuda Y., Wakahara Y., and Norigoe M.: "A New Algorithm for Fast Protocol Validation", Proc. of Compsac-86, pp.228-236, 1986.
3. Yuang M.C., and Kershebaum A.: "Parallel Protocol Verification: The Two-Phase Algorithm", Proc. 9th Intern. Symp. on PSTV, pp.339-353, 1989.
4. ISO: "Information Processing Systems-Open Systems Interconnection-Estelle: A Formal Description Technique Based on an Extended State Transition Model", ISO/DIS 9074, 1987.
5. CCITT: "Specification and Description Language(SDL)", Recommendation Z100, 1989.
6. Gordon M.J.C.: "A Proof Generating System for Higher-Order Logic" in "VLSI Specification, Verification and Synthesis", Kluwer Academic Publishers, pp.73-128, 1987-01.
7. Sarikaya B., Bochmann G.V., and Koukoulidis V.: "Method of Analysing Extended Finite-State Machine Specifications", Computer Communications, vol.13, no.2, pp.83-92, 1990-03.
8. Finkel A.: "A New Class of Analyzable CFSMs with Unbounded FIFO Channels", Proc. 8th Intern. Symp. on PSTV, pp.283-294, 1988.
9. Huet G., and Oppen D.:"Equations and Rewrite Rules A Survey" in "Formal Language: Perspectives and Open Problems", R. Book eds., Academic Press, pp.349-405, 1980.
10. ISO: "Basic Connection Oriented Session Protocol Specification", ISO 8327.
11. Higuchi M., Seki H., and Kasami T.: "A Method of Composing Communication Protocols with Priority Service", to appear in IEICE Trans. Commun., 1992-10.
12. Choi T.Y., and Miller R.E.: "A Decomposition Method for the Analysis and Design of Finite State Protocols", Proc. of 8th ACM/IEEE Data Comm. Symp., pp.167-176, 1983.
13. Lin H.: "Constructing Protocols with Alternative Functions", IEEE Trans. Comput., vol.40, pp.376-386, 1991-04.
14. Chow C., Gouda M.G., and Lam S.S.: "A Discipline for Constructing Multiphase Communication Protocols", ACM Trans. on Computer Systems, vol.3, pp.315-343, 1985-11.

(a) a conjunctive formula

$$
\begin{aligned}
P_i = \langle\ &\{\mathrm{STA713}\},\ \{\mathrm{STA713}\}\ \rangle && \texttt{(AF1)}\\
\wedge\ \langle\ &\varepsilon,\ \mathrm{MIP}^{+}\ \rangle && \texttt{(AF2)}\\
\wedge\ &step1(AB[1]) && \texttt{(AF3)}\\
\wedge\ &Vm(A) = last(AB[1]) + 1 && \texttt{(AF4)}\\
\wedge\ &Vm(B) = head(AB[1])\\
\wedge\ &Va(A) = Va(B)\\
\wedge\ &Vm(A) \geq Va(A)\\
\wedge\ &Vm(B) \geq Va(B)
\end{aligned}
$$

(b) term rewriting system

definition of δ_A:

event: $-\langle \mathrm{MIP}, Vm(A)\rangle$

$\langle STA713 \Rightarrow STA713\rangle$

$Vm'(A) \Rightarrow Vm(A) + 1$ (1)

$Va'(A) \Rightarrow Va(A)$. (2)

$\vdots$

properties of defined predicates:

$last(seq \cdot \langle type, n\rangle) \Rightarrow n$ (3)

$|seq| \geq 1$:

$head(seq \cdot \langle type, n\rangle) \Rightarrow head(seq)$ (4)

$|seq| \geq 1, step1(seq), t = last(seq) + 1$:

$step1(seq \cdot \langle type, t\rangle) \Rightarrow true$ (5)

$\vdots$

added rules by evaluating
the conditions of conditional rewrite rules:

$step1(AB[1] \cdot \langle type, Vm(A)\rangle) \Rightarrow true$ (5')

inductive hypotheses:

$step1(AB[1]) \Rightarrow true$ (6)

$\vdots$

relation between message sequences:

$AB[1]' \Rightarrow AB[1] \cdot \langle \mathrm{MIP}, Vm(A)\rangle$ (7)

Figure 1-A. A formula and a process of verification of $EX1$
(All terms with primes denote the values after transition.)

(c) assumed inequalities

inductive hypothesis

$$Vm(A) = last(AB[1]) + 1$$
$$Vm(B) = head(AB[1])$$
$$Va(A) = Va(B)$$
$$Vm(A) \geq Va(A)$$
$$Vm(B) \geq Va(B)$$

properties of defined functions

$$\vdots$$

(d) a process of rewriting a predicate

$$\begin{array}{ll} & step1(AB[1]') \\ \rightarrow & step1(AB[1] \cdot \langle \mathrm{MIP}, Vm(A) \rangle) \text{ by (7)} \\ \rightarrow & true \qquad \text{by (5')} \end{array}$$

(e) a process of rewriting an inequality

$$\begin{array}{ll} & Vm'(A) \geq Va'(A) \\ \stackrel{*}{\rightarrow} & Vm(A) + 1 \geq Va(A) \qquad \text{by (1),(2)} \end{array}$$

(f) the transitive closure

$$\underline{Vm(A)+1} \quad \geq \quad Vm(A) \quad \geq \quad \underline{Va(A)}$$

Figure 1-B. A formula and a process of verification of $EX1$
(All terms with primes denote the values after transition.)