A Way to Suppress Errors in Communication Protocols

César G. VIHO

I.R.I.S.A.*- Campus de Beaulieu - 35042 Rennes cedex - FRANCE
E-mail: viho@irisa.fr

Abstract

Because of the growth in performance required by networks users, communication protocols to be used in recent networks become today more complex to develop. Thus, designing error-free protocols becomes by the same almost impossible. A lot of works have been done in protocol verification to allow detection of errors in protocol but now, the real need of protocol designers is more solutions to correct the detected errors than errors detection. In this paper, we develop a method to help protocols designers in verification steps, such that the considered protocols are error-free and correspond as much as possible to the expected service.

The communicating entities are modelised by Finite State Machine (FSM) and the validation step is based on the construction of the reachability graph. After a correction oriented analysis of this graph, our method proposes sets of transitions of one or another FSM. The suppression of transitions in any of these sets eliminates the considered error without generating new errors. Then, criteria to choice sets based on preserving as much as possible the specified service are proposed. We show how our method works by applying it to correct errors in a data transfer protocol.

Keywords : Protocol, Service, Specification, Verification, Finite State Machine, Reachability Graph, Errors, Correctione.

*Institut de Recherche en Informatique et Systèmes Aléatoires

1 Introduction

The goal of formal specification and verification steps are respectively to describe protocols and to ensure their correctness before their implementation.

Today, languages such as ESTELLE [1], SDL [2] and LOTOS [4] allow complete description of many of all mechanisms which can occur in a protocol.

From the established results in protocol verification [11, 12], tools allowing protocols errors detection have been developped during the last years. The contribution brought by such tools is certainly important but still much more has to be done by designers to correct the detected errors. This difficulty leads some designers to pass over the verification step which is yet required for error-free protocol development. Some designers use techniques based on trial and error methods. Protocols implemented may work partially or be entirely unusable. The cost of implementation of a protocol is such that "*it is better to correct its specification rather than its implementation*".

Our aim is to develop an approach to cope with this problem. We detect errors and propose solutions to correct the detected errors in protocols according to their specified expected service.

2 Formalism and error detection

Protocols are composed by communicating entities called **processes** exchanging messages troughout FIFO **channels**.

2.1 Process

A process P is a labeled directed graph with two types of edges called **sending** and **receiving egdes**. A sending (resp. receiving) edge is labeled -m (resp. +m) for some message m in a finite set M of messages. Each node in P has a distinct label and a unique label. One of the nodes in M is identified as its initial node, and each node in P is reachable by a directed path from the initial node. Thus, (e,α,e') reprensents the unique edge from node e to node e' labeled α. Fig. 1 shows two processes representing a simple data transfer protocol.

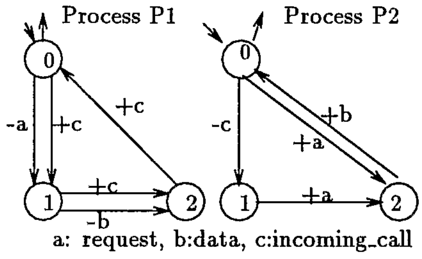

Figure 1: A simple data transfer protocol

2.2 Reachability graph

Let $(P_i)_{i\in\{1,2,...,n\}}$ be n processes with the same set M of messages. A **global state** s of $(P_i)_{i\in\{1,2,...,n\}}$ $s = < e_1, \ldots, e_n, c_{12}, c_{21}, \ldots, c_{(n-1)n} >$ is an n_tuple where $(e_i)_{i\in\{1,2,...,n\}}$ are the labels of n nodes in $(P_i)_{i\in\{1,2,...,n\}}$ and $(c_{ij})_{i,j\in\{1,...,n\}}$ are strings of messages from the set M. Informally, the global state s represents a situation of the communication between the n processes where each process P_i reachs node e_i while their input channels have the sequences $(c_{ji})_{j\in\{1,2,...,n\}}$, respectively.

The initial global state is $s^0 = < e_1^0, .., e_n^0, \varepsilon, .., \varepsilon >$ where $(e_i^0)_{i\in\{1,..,n\}}$ are the labels of the intial nodes of the processes and ε is the empty string.

Let $s = < e_1, \ldots, e_n, c_{12}, c_{21}, \ldots, c_{n(n-1)}, c_{(n-1)n} >$ be a global state of $(P_i)_{i\in\{1,2,...,n\}}$ and let t be an output edge of P_i from node e_i to node e_i'. A global state s' is said to **follow s over t** iff one of the following conditions is satisfied:

1. t is an edge from e_i to e_i' in P_i and is labeled -m, and there exists $i \in \{1, 2, .., n\}$ such that $s' = < e_1, .., e_i', .., e_n, c_{12}, .., c_{ij}.m, .., c_{(n-1)n} >$

2. t is an edge from e_i to e_i' in P_i and is labeled +m, and there is $j \in \{1, \ldots, n\}$ such that $c_{ji} = m.c_{ji}'$ and $s' = < e_1, .., e_i', .., c_{12}, .., c_{ji}, .. >$

A global state s' **follows** another global state s iff there is an edge t in a process $(P_i)_{i\in\{1,2,...,n\}}$ such that s' follows s over t. A global state s' is **reachable from** another global state s iff there exists r $(r \geq 2)$

global states $s^1, \ldots, s^r$ such that $s = s^1$, $s' = s^r$ and s_{i+1} follows s^i for $i = 1, \ldots, r-1$.

$Rch(s)$ is the set of global states reachable from a global state s.

A global state s is **reachable** iff it is reachable from the initial global state s^0, that means $s \in Rch(s^0)$.

The verification procedure generates the **reachability graph.** This graph is a labeled directed graph where nodes are initial global state and all the reachable global states: $\{s^0\} \cup Rch(s^0)$. In this graph, an edge from a node s to s' is labeled by the edge in the corresponding process over which s' follows s. Figure 2 below, shows the reachability graph obtained from the example given in figure 1.

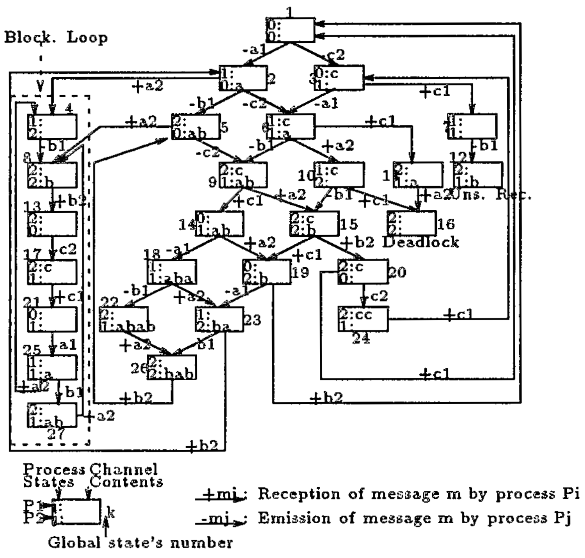
Block. Loop
Deadlock
Uns. Rec.
Process Channel States Contents
Global state's number
+mi: Reception of message m by process Pi
-mj: Emission of message m by process Pj

Figure 2: Reachability graph of the example

The problem of combinatory explosion caused by these construction is not our main concern (see [10] for reduction methods). We assume that all reachable global states are generated (or can be obtained when desired). We are particularly interested by those which represent errors in the communication protocol.

2.3 Errors detected

In this paper, we will consider three types of error:

1. An **Unspecified Reception** in a process P_i for a message m sent by a process P_j which occurs in a global state s is iff:

   - $c_{ji} = m.v$, where v is a sequence of message.
   - No output edge of the current node of process P_i is labeled +m.
   - for all global states in $Rch(s)$, the process P_i stays in the same node.

   As consequences, the only way for P_i to progress is to receive the message m from its input channel and there is any receiving edge labeled $+m$.

2. A **Deadlock** which is a reachable global state s where any of the processes can't progress.

   That means, $Rch(s)$ is empty.

3. A **Locking Loop** is a set of reachable global states different from the initial global state that form a cycle in the reachability graph.

   For two different global states s and s' in a locking loop and every global state s^1 not in this locking loop, we have:

   - $s \neq s^0$, $s' \neq s^0$, $s' \in Rch(s)$, $s \in Rch(s')$ and
   - $s^1 \notin Rch(s)$ and $s^1 \notin Rch(s')$.

Other errors can be detected (Unbounded Communication, Nonexecutable Transitions,...) but the method we propose in the next section, relates only to the three errors we have defined above.

3 Problems

Errors are represented by global states in the reachability graph. To eliminate errors, we have to suppress those global states. But, the reachability graph is built upon processes. Thus, the suppression of global states cannot be made without consideration of those processes.

In fact, suppressing a node in a graph consists in the suppression of all edges which leads to this node and all nodes issued from it. In a reachability graph, the label of an edge is a edge of one of the processes. Thus, the label of an edge which leads to a global state can be the label of other edges in the reachability graph. Consequently, the suppression of an edge in the reachability graph causes the suppression of all edges which have the same label. This makes the *suppression of global state much more complex* than a suppression of a state in a simple graph.

Moreover, by suppressing edges in a reachability graph, we can suppress reachability to global states other than those concerned by this suppression. According to the definition of the reachability graph, suppressing edges to eliminate an error can generate other errors. This problem, called the **non-regression problem**, obliges to make judicious choice of edges to be suppressed in the processes.

Another important problem comes from the fact that a protocol is designed to provide a service. Suppressing errors modify the service provided by the protocol. How can we ensure that the corrected protocol provides the initial service for which it has been designed ? We call this the **service problem** and discuss it later.

4 Solutions

Any solution to the non-regression problem requires a study of the cause-and-effect relationship between the suppression of edges or nodes in a process and the suppression of edges or global states in a reachability graph.

In the following section, gradually as we give fundamental steps that permit us to obtain solutions to the problem of error's suppression without generating new error. We give definitions and some results allowing the understanding of our solutions.

4.1 Fundamental steps of resolution

A global state is said to be an **error** if it is either a deadlock, an unspecified reception or a locking loop.

A global state s is said **resettable** iff the initial global state s^0 is reachable from s; otherwise s is **unresettable**. RS (resp. $\overline{RS}$) is the set of resettable (resp. unresettable) global states.

Thus, $Rch(s^0) = RS \cup \overline{RS}$.

A global state in $\overline{RS}$ is either an error or leads necessarily (within a directed path in the reachability graph) to an error.

Let E be a set of reachable global states representing the detected error to be suppressed.

In order to make the different steps developped below understandable, edges in the reachability graph are still called edges and, edges in processes are called transitions.

Step 1: Frontier Cut Plane

The analysis of the reachability graph obtained after verification step gives a first set of transitions which suppression eliminates the error. This set can suppress more than what is needed to suppress the considered error. So, more analysis is needed.

Let $NR(E)$ be the set containing E and all global states in $\overline{RS}$ which are predecessor of at least one global state in E.

We constitute the set $Fr(E)$ of edges of the reachability graph that are from global states in RS to global states in $NR(E)$.

We then build a set $FCP(E)$ of transitions called the **Frontier Cut Plane** of E. $FCP(E)$ contains the transitions which are the labels of the edges in $Fr(E)$.

Because it suppresses reachability to global states of $NR(E)$, it is easy to prove that the suppression of $FCP(E)$, suppresses all reachability to every global states of E [9] (cf the definition of reachability graph in section 2.2). Consequently, the error is suppressed.

But, the suppression of transitions induces the suppression of all edges which are labeled by this transitions. Thus, we can eliminate more global states than needed to suppress the considered error. The next step resolves this problem.

Step 2: Reduced Cut Plane

Reductions of the Frontier Cut Plane generate minimal sub-sets which suppression also eliminates the considered error. This sub-sets minimise the number of global states suppressed.

Remark that every global state in $NR(E)$ follows (over a edge in $FCP(E)$) a global state of the reachability graph. Let $pred(NR(E))$ be the set of such global states, then the suppression of $pred(NR(E))$ also eliminates all reachability to E. This means, there are several solutions to suppress an error.

We say that two sets T and T' of transitions are **equivalent for the suppression** of a set G of global states iff the suppression of T or T' eliminates all reachability to global states of G. We note this $T \equiv_G T'$. Otherwise, we note $T \not\equiv_G T'$.

We add to $FCP(E)$, all the labels (transitions) of all the edges in the reachability graph which are on a path from initial global state to a global state of $pred(NR(E))$. $AT(E)$ denotes the set we obtain in this manner.

Any set of transitions such that its suppression eliminates the error E is added to $AT(E)$.

From $AT(E)$, we build sub-sets of $AT(E)$, called **Reduced Cut Plane** of E and denoted $RCP(E)$. Every $RCP(E)$ satisfies the two following conditions:

1. $RCP(E) \equiv_{NR(E)} FCP(E)$
2. $\forall t \in RCP(E), RCP(E) \not\equiv_{NR(E)} RCP(E) \setminus \{t\}$

According to the definition of the $\equiv_G$ relation and of $NR(E)$, the suppression of any Reduced Cut Plane of E also suppresses E in the reachability graph.

As explained in section 3, this suppression can generate other errors. That is why another step is needed.

Step 3: Total Cut Plane

The resolution of the non-regression problem consists in adding to every Reduced Cut Plane, transitions intended to eliminate the errors generated by the suppression of the transitions in this Reduced Cut Plane. The suppression of any of the sets of transitions constructed in this manner eliminates the considered error without generating others.

Let us consider a Reduced Cut Plane $RCP(E)$. Two situations can be observed:

- **The suppression of $RCP(E)$ doesn't introduce new errors in the reachability graph.**

  Non-regression problem is solved and the suppression of RCP is the solution to our initial problem.

- **The suppression of the considered $RCP(E)$ introduces new errors in the reachability graph.**

  We denote $Rgr(RCP(E))$ the set of such new errors introduced by the suppression of $RCP(E)$.

  It is clear that the reachability graph obtained after the suppression of $RCP(E)$ contains less global states and less edges than $Rch(s^0)$.

  As we apply step 1 and step 2 to suppress E, we can do so to suppress the set $Rgr(RCP(E))$ in the smaller reachability graph obtained after the suppression of $RCP(E)$.

  The sets we obtain in this way are called **First Non-regression** set of transitions associated to $RCP(E)$.

  Every set of transitions formed by a first non-regression set of transitions added to $RCP(E)$ is called **First Non-Regression Cut Plane** of E: $FNR(E)$.

  For each $FNR(E)$ which suppression introduces other errors, we can iterate the same procedure applied to suppress $Rgr(RCP(E))$, until there is no more error introduced or there is no more transition to be suppressed (this is the worst situation we can obtain).

  The sets obtained in this manner are called **Total Cut Plane** of E. We prove that the suppression of a Total Cut Plans either suppresses the considered error without generating other errors or suppresses all the reachability graph. This last case means that we don't have solution and particularly that the protocol is "very badly" specified.

Finally, the service problem (of conserving provided service) is discussed later, but here and now we can propose two criteria to choose between the Total Cut Planes that contain a minimum number of transitions or that suppresses a minimum of global states. Note that, solutions obtained with this two different criteria can bring about equal consequences on the considered protocol.

4.2 Proofs

Before showing how our method works on an example, we give here main ideas (see [9] for details) proving its termination and availibility.

4.2.1 Termination

Communicating entities are finite state machines and we have supposed all the reachability graph constructed. At all steps of our method, we propose suppression.

Thus, suppression method in a finite domain ensure that our method always terminates.

4.2.2 Availability

According to the partition of reachable states we have made ($Rch(s^0) = RS \cup \overline{RS}$) in section 4.1, two situations are to be considered:

1. RS is empty.

   This means that all global states are or lead to error. Here, I would like to say that "it is difficult to correct automatically a protocol where everything goes wrong".

   This is also true for our method, and in this case it proposes the suppression of all the protocol meaning by this that no solution is found.

2. RS is not empty.

   We have also proved that our method proposes at least one Total Cut Plane such that its suppression preserves global states in RS. This can be seen as the minimal service the protocol should provide.

   This is due to the notion of frontier between RS and global states in $NR(E)$ we consider in step 1 of our method.

4.3 Application to an example

The example of figure 1 is choosen because it contains the three errors described in section 2.3 and it is well done to show how our method works.

Errors	Global States To Be Suppressed
Blocking loop	{4, 8, 13, 17, 21, 25, 27}
Uns. Reception	{12}
Deadlock	{16}

In the two tables below, (-mi,j) (resp. (+mi,j)) represents the emission (resp. reception) of a message m in state j by the process Pi. A number in the field "Global States to be suppressed" represent the corresponding global states in the reachability graph in figure 2.

Locking loop's suppression			
Steps	Transitions To Suppress	Global States Suppressed	Errors Generated
1	{ (+a2, 0) }	{4, 8, 13, 17, 21, 25, 27}	no error
2	{ (+a2, 0) }	/	/
3	{ (+a2, 0) }	/	/

In this case, there isn't non-regression problem and the solution proposed is the suppression of the "Request" message reception in the initial node of process P_2.

Now, let us consider the deadlock. The result obtained are resumed in the following table.

Deadlock's suppression			
Steps	Transitions To Suppress	Global States Suppressed	Errors Generated
1	{ (+a2,1), (+c1,1) }	{10, 11, 15, 16, 19, 20, 23, 24, 26 }	{22} Un. Rec.
2	{ (+a2,1) }	/	{22, 11} Un. Rec.
	{ (+c1,1) }	{11, 16}	no errors
3	all	all	/
	{ (+c1,1) }	{11, 16}	/

This method has been implemented in a tool [7, 8], and has given interesting results on real protocols such as "Packed Radio Network Management" [6] and a simplified version of X.25 protocol.

5 The Service Problem

By suppressing transitions in the processes we modify the protocol and by the same we modify the service it provides. Thus, what we can do is to give a measure allowing a comparison between the service provided by the corrected protocol and the expected one. This section discusses about how to do this.

As well as we can specify processes of a protocol with FSM , we can also specify service as a Finite State Machine. The difference is that what we can obtain and treat formally can only be an abstraction of this service. For example, figure 3 shows the abstract service of the protocol of figure 1.

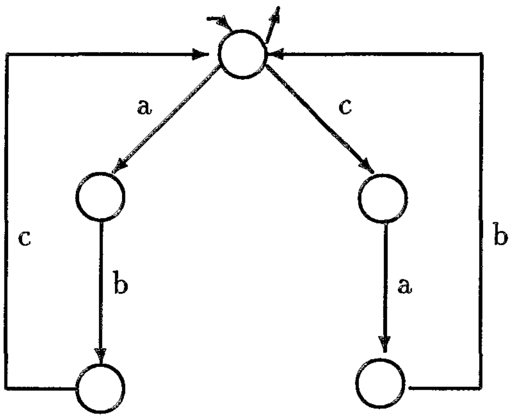

Figure 3: An Example of the Expected Service

We make projections and equivalences transformations [9] on the reachability graph obtained after the suppression of the proposed transitions contained in the Total Cut Plan. This produces another graph which represents a abstract vision of the protocol according to the service. With algorithms allowing the verification of observationnal equivalence (in the sense of recognized languages) we compare this transformated graph with the abstract service graph.

This is yet another criterion for the choice of the Total Cut Plan which is semantical whereas the two previous (see section 4.2) are rather syntactical. Notice that this criterion is the most usefull because it expresses what the designer wanted the protocol to do. That is why we assume that it is possible to correct errors in protocols while preserving the service for which it has been designed.

As an exemple, the minimal and deterministic abstract service of the transformated graph obtained after the suppression of the errors is shown in figure 4.

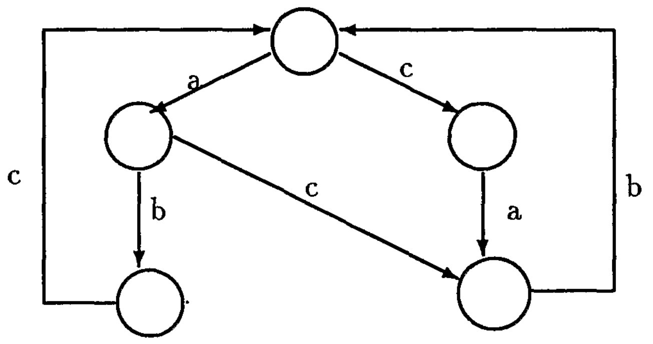

Figure 4: Global Abstract Service Provided by the Corrected Protocol

We can see that the expected service is included in the service provided by the corrected protocol. Considering that this is sufficient depends on the designer. But in the case that we have the reverse (the service provided by the corrected protocol is included or different from the expected one), we need to add transitions rather than suppress.

6 Conclusion

We have proposed in this paper a contribution to the resolution of the error correction problem which is a recognized difficult and not really investigated problem [3, 5, 12]. The difficulty comes from the fact that we have to use the reachability graph which is generally very large.

We have presented a method, which allows correction of given errors without generating other errors. Criteria based on projections towards the specified expected service are proposed to make the service provided by the corrected protocol correspond as most as possible to the initially expected one.

Finally, our method is based on suppression but sometimes transitions addition is needed to correct errors in protocols. Methods based on addition are more difficult [5]. Meanwhile, our current work concerns this area but results we have in this domain are to be improved. Used in combination, transitions suppression and addition would provide complete solution to correct errors in protocol specification and verification steps, and by the same, provide the error-free transmission of informations required by network users.

References

[1] ISO-IS-9074, "ESTELLE, a Formal Description Technique Based on an Extended State Transition Model", ISO International Standard 9074, 1989.

[2] CCITT/SGXI/WPWP3-1, "Specification and Description Language", CCITT Recommandations Z100-Z104, May 1983.

[3] P. Guitton, "Description, validation et tests de conformité de protocoles de communication", Thèse no. 1941, Université Bordeaux I, Jan. 1984.

[4] ISI/IEC-JTC1/SC21/WG1/FDT/C, "IPS - OSI - LOTOS, A Formal Description Technique Based on the Temporal Ordering of Observational Behavior", IS 8807, Feb. 1989.

[5] C. Jard et M. Raynal, "De la nécessité de spécifier des propriétés pour la vérification des algorithmes distribués", Rapport de rech. no. 590, INRIA - Rennes, Dec. 1986.

[6] C. V. Ramamoorthy, S.T. Dong and Y. Usuda, "An Implemantation of an Automated Protocol Synthesizer (APS) and Its Application to the X.21 Protocol", *IEEE Trans. on Soft. Eng.*, vol. SE-11, No. 9, pp 886-908, Sep. 1985.

[7] G. Viho, "Manuel d'utilisation de l'outil de validation de protocoles VAP", Rapport interne No. I-8916, LaBRI - Université Bordeaux I, Avr. 1989.

[8] G. Viho, B. Cousin, R. Castanet and O. Rafiq, "Protocol Correction: Methods and Tools", Computer Networks'91, pp 247-276, Wroclaw - POLAND, Jun. 1991.

[9] G. Viho, "Une contribution à la suppression des anomalies dans les protocoles de communication", Thèse de Doctorat de l'Université Bordeaux I no. 666, Sep. 1991.

[10] L. Cacciari, O. Rafiq "On Improving Reduced Reachability Analysis", FORTE'92, Fith Int. Conf. Workshop on Formal Description Techniques, Lannion, Oct. 1992.

[11] C. West, "General Techniques for Communications Protocol Validation", in Proc. Comput. Network Protocols Symp., Liège, Belgium, Feb. 1978.

[12] P. Zafiropulo, C. West, H. Rudin, D.D. Cowan and D.Brand, "Towards Analyzing and Synthesizing Protocols", IEEE Trans. Com., vol. COM-28, pp. 651-661, Apr. 1980.