MTA SZTAKI Közlemények 33/1985 p. 37-45

AUTOMATED PROTOCOL VERIFICATION

László KOVÁCS

Computer Network Department
Computer and Automation Institute of the
Hungarian Academy of Sciences

1. INTRODUCTION

Distributed systems and computer networks are widely used all over the world. A protocol represents a set of rules which control the inner communication of a computer network. The correct operation of this algorithm fundamentally influences the operation and the realiability of the whole computer network. The use of an erroneous protocol can result in incalculable consequence so that at present the protocol designers specify protocols with great care. Formal techniques to check the protocol correctness are called verification methods.

The precondition of a protocol verification is the existence of the formal description of the protocol and its communication services. In this paper we present an automated protocol verification method which analyses all possible behaviour of a protocol system described in a high level specification language. This verification method is one of the family of "global view" methods because of its use of global state information. The "local view" methods (e.g. program proving approach) are based on the analysis of the interactions between the protocol entities and their local encironments. [Bochmann-Sunshine 80]

2. PROTOCOL PROPERTIES

Verifiable protocol properties may be classified into two categories: ([Sunshine 79]) specific and general properties. Specific properties are those that characterize the protocol individually. The most important specific protocol property is that of the functional correctness. It means that the protocol meets its service specification. The general properties are those that are common to all protocols. The author considers that the properties described below are the most essential general properties.

The completeness of the protocol means that the protocol entities detect all interactions arrived from their environments and produce output interactions when required.

Protocols are expected to operate free from deadlock. Deadlock is a system state from which there is no exit. In the case of deadlock there is a logical loop of protocol entities in which each entity is waiting for interaction of the preceding one.

The liveness property means that every protocol system state is reachable from each reachable system state.

The progress (or tempo-blocking freeness) protocol property is the absence of cyclic behaviour where no useful activity takes place. In other words the protocol never gets into any non-productive looping.

The recoverability concept is based on the view of dividing the protocol system state space into two parts: the normal and the abnormal spaces. The protocol is recoverable if after the occurence of a temporary aberration the protocol will return to the normal operation space in a finite time [Merlin-Farber 76].

The protocol is stable if the step's number to recover the protocol does not increase in the operation time.

3. ABSTRACT MODEL OF PROTOCOLS

3.1. THE STATE TRANSITION APPROACH

We make a proposal as to the abstract conceptual model of protocol systems. To model the protocols we present a non-deterministic partitioned named transition system which is a modified version of the Keller model of parallel computation [Keller 76].

Definition 1. A partitional transition system is a quadrople (Q,R,N,P) where Q is a set of states, R is a binary relation on Q, N is a set of (action) names and P is a binary partition of actions.
$(p \in P \wedge \forall i<>j(p[i] \cap p[j] = \emptyset) \wedge \forall i(p[i]<>\emptyset))$ @

Denote $q \rightarrow q'$ the R relation of $q,q' \in Q$ states which is called state transition. The system in the case of $q \rightarrow q'$ executes an indivisible (atomic) action. (Notation:

$q \xrightarrow{p.n} q'$, $n \in N$, $p \in P$)

Definition 2. Let (Q,R,N,P) be a partitional transition system. The set of possible next states is defined to be
$E(q) = (q' : q \rightarrow q' \wedge q,q' \in Q)$ @

Definition 3. The set of enabled actions is defined to be

$V(q) = \{n : n \in N \wedge q' \in E(q) \wedge q \xrightarrow{p.n} q'\}$ @

Definition 4. The $S = q[0], q[1], \ldots q[i], \ldots$ is a state sequence of the (Q,R,N,P) system if and only if
$\forall i\ (q[i] \in Q \wedge i>0(q[i] \in E(q[i-1])))$ @

Definition 5. The $T = n[1], n[2], \ldots n[i], \ldots$ is computation on the (Q,R,N,P) system if and only if
$\exists S\ (\forall i\ (n[i] \in V(q[i-1])))$ @

Definition 6. The (Q,R,N,P) system is nondeterministic if $\exists q\ (q \in Q \wedge \#E(q)>1)$ @
(Notation: # cardinality)

In protocol systems there are parallel overlapped actions. To describe protocol we apply the nondeterministic (Q,R,N,P) system so that we transform the parallel systems into that kind of system. In the transformation we represent the parallel overlapped actions with their sequential execution of optional order.

Definition 7. The total protocol specification is the definition of the (Q,R,N,P) transition system. @

To form a method which can be used in practice let K be a predicate and let L be a state transition function on objects representing Q space. The correspondence between the (K,L) and the (Q,R,N,P) systems is the following.

$q,q' \in Q,\ \ p \in P,\ \ n \in N\ \ ((q \xrightarrow{p.n} q')$ if and only if $(K(p\cdot n,REP(q)) \wedge REP(q') = L(p.n,REP(q))))$.
The REP function refers to the objects representing Q space.

Definition 8. The protocol specification is the description of the (K,L) representation. @

The main difference between the two definitions (7. and 8.) is that (K,L) is a "human executable" and (Q,R,N,P) is a "machine executable" model. In the [Kovacs 82] report the author published a high-level language implementation of the (K,L) system.

3.2. FORMULATIONS OF PROTOCOL PROPERTIES

In order to formulate the protocol properties presented in the second section of this paper define the concept of the reachability relation (R^*). Let R^* be the reflexive and transitive closure of R (Keller definition 3.1). (Notation: $\xrightarrow{*}$

Definition [Keller definition 3.2) *9.* Let I be a predicate on Q. I q[O]-invariant in (Q,R,N,P) if and only if $\forall$ q (q[O] $\xrightarrow{*}$ q $\wedge$ I(q)) @ (q[O] is the start state of the system).

In the formal description of deadlock freeness protocol property we follow the extention of the Keller's definition to the (Q,R,N,P) model.

Definition 10. Let ACTIV be a predicate on Q. It is defined to be ACTIV(q) if and only if #E(q))>= 1 @

Definition 11. The protocol is deadlock free if and only if ACTIV(q) q[O]-invariant @

Formulation of the liveness property corresponds to the informal description of the 2. section.

Definition 12. Let REACHABLE be a predicate on Q. REACHABLEq'(q) if and only if q $\xrightarrow{*}$ q' @

Definition 13. The protocol has liveness property if and only if (REACHABLEq q[O]-invariant) q[O]-invariant. @

In this formalism it appears to be very difficult to construct a formal definition of the progress property since it requires the formulation of the usefulness of the protocol cycles so we do not try it.

4. PROTOCOL VERIFICATION

4.1. IMPROVED REACHABILITY ANALYSIS

The improved reachability analysis verification method check the q[O]-invariance of a predicate. It is based on the reconstruction of (Q,R,N,P) transition system from (K,L) protocol representation (language model). This reconstruction consists of building a G graph of (Q,R,N,P) system. By the G graph of (Q,R,N,P) system we mean an oriented graph whose nodes correspond to the Q states and whose arrows correspond to the state transitions labelled by action names. In other words this means that there is an arrow from the node indicated by q to the node q' labelled by p·n if and only if there exists $q \xrightarrow{p \cdot n} q'$ relation in (Q,R,N,P) system. The correspondence between the structural (geometrical) properties of the G graph and the protocol properties defined in the 3.2 section makes it possible to map the protocol verification to the structural analysis of the G graph.

The node in the G graph from which no arrows descend represents a protocol deadlock state (definition 11.). Protocol has liveness property if the G graph is strongly connected (definition 13.). Several algorithms were published to examine the strong connectivity of an oriented graph (e.g. [Aho-Hopcroft-Ullman 75]). The protocol cycles correspond to the loops of G graph. Examination of protocol cycles to determine the useless loops (progress property) concerning the human designers demands the knowledge of a concrete algorithm. Other protocol properties formulating in the form of predicate q[O]-invariance can be examined with the evaluation of predicates in some nodes of G graph.

In order to put this method into practice we have to define the system state in the case of concrete language representation of (K,L) system. It means the definition of the REP^{-1} function. The language representation of (K,L) system was published in [Kovacs 82] uses entities communicating with each other to describe the protocol. In that case the REP^{-1} function was defined a q = <e[1],e[2],...e[n]> where q is equal to a vector (n-tuple) of the states of entities.

4.2. COMPUTER-AIDED VERIFICATION OF PROTOCOLS

The automated protocol verification system described below is an interactive software tool realizing the improved reachability analysis of 4.1 section. In the first step the protocol designer describes the protocol by means of the specification language. In that process the designer uses the language constructions and abstract data types predefined in the open abstract data type library of the system. The specification language makes it possible to express the protocol properties demanded by the designer in the form of predicates. The language specification of the protocol is the input of the verification system. In the first pass the system compiles the specification then it starts the construction of G graph and the examination of it's structural properties. The designer can visualize the system informations in the way of the progress of the verification by means of a display. The job of designer sitting before the display is to "cut" the infinite branch of G graph in the case when no predicate signals it. The user of the system can stop the verification process by an interactive command and can ask the actual values of protocol variables.

The speed of system operation is determined to the greatest extent by the fact that the system can't keep the whole G graph in memory because of it's size. It is necessary to swap the parts of the graph to the disk. As a result of this fact, a protocol verification process exceeds the average session time.

The first experiences of verifying system operation - despite the awkwardness of the system - strengthen our resolution which can lead to the interactive automated systems for analyzing and synthesizing protocols in the near future.

REFERENCES

[Aho-Hopcroft-Ullman 75] Aho V.A., Hopcroft J.E., Ullman J.D., "The Design and Analysis of Computer Algorithms," Addison-Wesley Publ. Comp. 1975.

[Bochmann-Sunshine 80] Bochmann G.V., Sunshine C.A., "Formal Methods in Communication Protocol Design," IEEE Trans. on Comm. Vol. COM-28, No. 4. Apr. 1980.

[Keller 76] Keller R.M., "Formal Verification of Parallel Programs," Comm. of the ACM Vol. 19. No. 7. Jul. 1976.

[Kovacs 82] Kovacs L., "Formal Specification and Verification of Computer Network Protocols," Technical Report, MTA SZTAKI Tanulmányok 138/1982. (In Hungarian)

[Merlin-Farber 76] Merlin Ph.M., Farber, D.J., Recoverability of Communication Protocols - Implications of . Theoretical Study," IEEE Trans. on Comm. Vol. COM-24, S[illegible]. 1976.

[Schultz-Rose-West-Gray] Schultz G.D., Rose D.B., West C.H., Gray J.P., "Executable Description and Validation of SNA," IEEE Trans. on Comm. Vol. COM-28, No. 4. Apr. 1980.

[Schwabe 81] Schwabe D., "Formal Techniques for the Specification and Verification of Protocols," Technical Report, UCLA Apr. 1981.

[Sidhu 82] Sidhu D.P., "Rules for Synthesizing Correct Communication Protocols," Comp. Comm. Rev. Vol. 12. No. 1. Jan. 1982.

[Sunshine 79] Sunshine C.A., "Formal Techniques for Protocol Specification and Verification," Computer, Sep. 1979.

Automatikus protokoll verifikálás

Kovács László

Összefoglaló

A dolgozat a számitógép-hálózati protokollok verifikálására, helyességének ellenőrzésére szolgáló automatikus módszert mutat be. Definiálja a protokollok egy absztrakt matematikai modelljét. Az absztrakt modell lehetőséget teremt a protokollok legfontosabb tulajdonságainak formális értelmezésére. A közölt módszer a protokollok specifikációs nyelvü modelljét visszavezeti az absztrakt modellre, tehát az implementált verifikáló rendszer képes a protokollok tulajdonságait felderiteni.

АВТОМАТИЧЕСКАЯ ВЕРИФИКАЦИЯ ПРОТОКОЛОВ

Л. Ковач

Р е з ю м е

В статье описывается автоматический метод проверки правильности и верификации протоколов вычислительных сетей. Определяется абстрактная математическая модель протоколов. Абстрактная модель дает возможность для формального толкования важнейших характеристик протоколов. Описанный метод сводит модель со спецификационным языком к абстрактной модели протокола, следовательно реализованная система верификации способна определить характеристики протоколов.