Test result analysis and validation of test verdicts

G. v. Bochmann, D. Desbiens, M. Dubuc, D. Ouimet and F. Saba

Université de Montréal

August 1990

Abstract

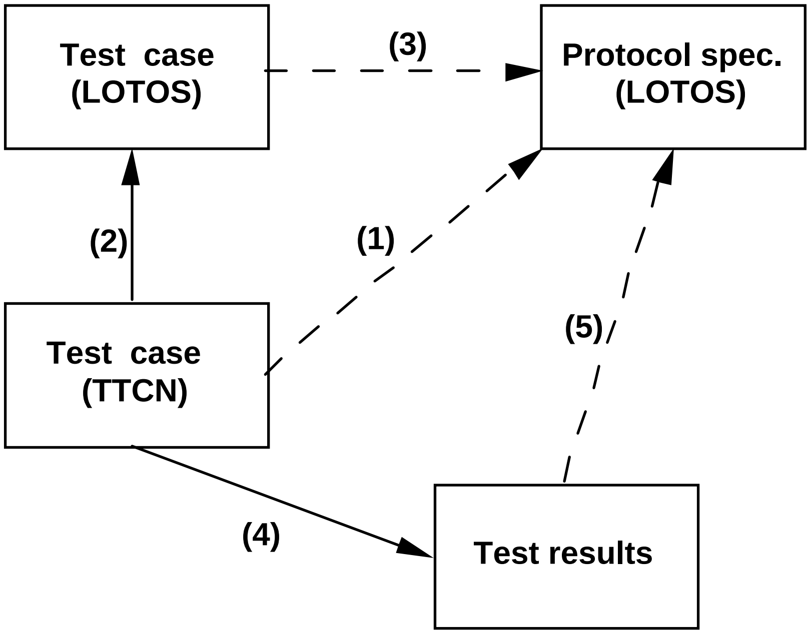
Formal description techniques (FDT's) are useful in the protocol development cycle, particulary in the conformance testing area. In this paper, we present TETRA, a test and trace analysis tool based on the LOTOS FDT which can be used to automatically compare the specified verdicts of a conformance test case with a protocol specification, or to analyse results of a test run with the reference specification. We also describe our experience with this tool for the validation of a X.25 TTCN test suite and for the testing of an ACSE implementation.

1. Introduction

In recent years, ISO and CCITT have developed various languages for the description of communication protocols and services [Boch 90g]. On the one hand, so-called formal description techniques (FDT's) [Este 89, Loto 89, SDL 87] are intended for writing formal specifications for the OSI protocols and services to be used during the protocol development process [Boch 87c]. On the other hand, semi-formal methods, such as ASN.1 [ASN1], are used in many existing OSI standards and proposals. Another language, TTCN [ISO C3], is used for the description of OSI conformance test suites. We consider in this paper the use of an FDT tool for two different aspects of conformance testing: (1) the validation of TTCN test suites, (2) and test trace analysis in respect to the corresponding protocol specification described in an FDT language.

Conformance testing of an implementation under test (IUT) usually involves a test suite consisting of a large number of test cases, each defining a certain number of execution paths, in the following called scenarios. Each scenario contains a so-called verdict which indicates whether the IUT "passes" or "fails" the test in question. A third verdict called "inconclusive" is also possible which indicates that the observed behavior of the IUT satisfies the rules of the specification, but the particular behavior to be tested could not be observed.

Although certain methods for automatically developing a test suite have been described (see for instance [Sari 89c]), most test suites are developed manually. It can therefore be expected that proposed test suites contain certain errors which should be detected as soon as possible. In particular, the verdicts of the test cases should be consistent with the protocol specification. This means that any scenario with verdict pass or inconclusive should correspond to a sequence of observed input and output interactions which is valid according to the protocol specification, and the sequence of interactions corresponding to a scenario with verdict "fail" should not be valid according to the specification. This is indicated in Figure 1 by the arrows (1) and (3). The translation of a TTCN test case into an equivalent LOTOS test case (arrow (2)) will be addressed in Section 3.



Explanations:

(1) Checking consistency of test case with protocol specification
(2) Translation from TTCN into LOTOS
(3) Same as (1) for test case described in LOTOS
(4) Execution of test case
(5) Analysis of test results in respect to protocol specification

Figure 1. Validation Steps

In addition to test suite validation discussed so far, we consider in this paper also the analysis of test results during the testing of an IUT . Again, the protocol specification can be used as reference, this time for the analysis of the observed interaction trace, as shown by arrow (5) in Figure 1. In the case of standardized test cases including verdicts, the analysis of the test results can be performed based on the verdicts, however, such an approach is not possible when other test cases are used which may be required for additional test coverage or the testing of implementation-dependent features. In all such cases, including random test inputs, the test results can be analysed directly in respect to the specification using automated tools [Boch 89m].

In Section 2 we describe a tool, called TETRA, which can be used to automatically compare the specified verdicts of a conformance test case (arrow (3) in Figure 1) with protocol specification, or to analyse results of a test run (arrow (5) in Figure 1) with the reference specification. The latter is especially useful if non-standard test cases are used which do not contain verdicts [Boch 89j].

In Section 3, we describe an experiment of validating the verdicts of a real OSI conformance test suite with a formal protocol specification. A large number of the test cases of the ISO/CCITT test suite for the LAP-B of X.25 was validated against a protocol specification written in LOTOS [Guer 89a]. The result of this experiment is threefold: (1) The translation of the test cases into LOTOS turned out to be relatively straightforward, as discussed in [Dubu 90], (2) the TETRA tool was debugged and improved, and (3) a number of errors were detected, not only in the translated test cases in LOTOS, but also in the original TTCN test case definitions [ISO 8882] and the formal protocol specification .

In Section 4, we describe an experiment with test result analysis for an Application layer protocol. The purpose of this experiment was at the same time to demonstrate tools for ASN.1 which were developed for implementation support in conjunction with the FDT Estelle [Boch 90f], and for the support of ASN.1 in relation with LOTOS [Boch 89h]. The Association Control protocol (ACSE) was used in this experience because of its relative simplicity. The experiment consisted of having two ACSE implementations communicate with one another and having the exchanged PDU’s observed and automatically analysed by the trace analysis tool TETRA, as shown in Figure 2.

Section 5 contains a discussion of our conclusions based on the practical results obtained in the two described experiments and on another experiment with the OSI Transport protocol. We also comment on the feasability of our approach for more complex protocols.

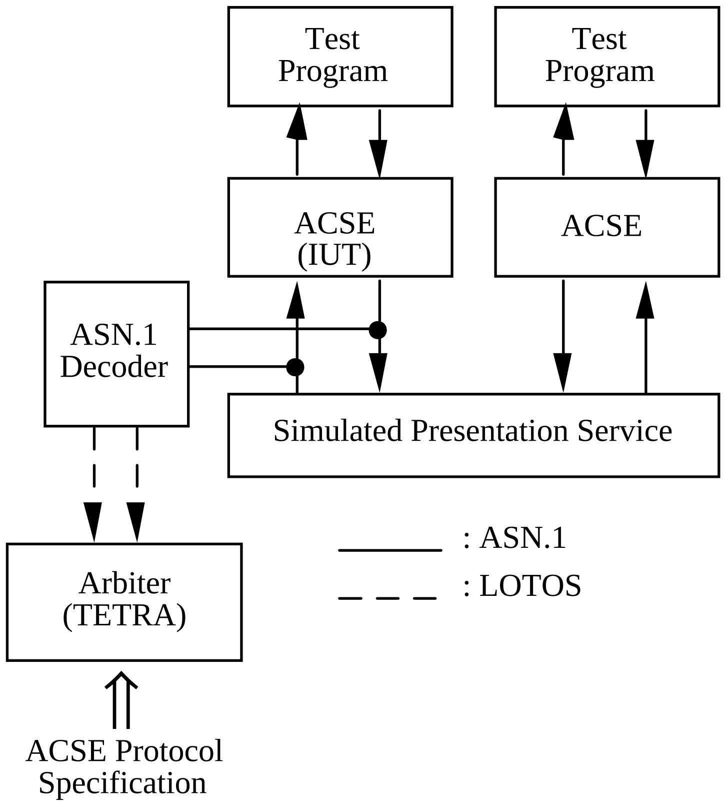


Figure 2. ACSE Test Experiment

2. The TETRA tool

We have modified an existing LOTOS interpreter [Logr 88] in such a way that it checks whether a given trace of observable interactions g_1, ..., g_n is valid in respect to a given reference specification S[g_1, ..., g_n]. The original interpreter requires interactions with the user for the selection of the next event to be interpreted, which may be an observable interaction at the gates g_1 through g_n, the internal event i, or an internal interaction at one of the gates hidden by the specification. Another version of the interpreter generates an overview of all possible sequences of observable interactions using backtracking over all possible internal events in case that several non-deterministic choices exist [Guil 89]. However, this latter version does not handle any interaction parameters. Our modified interpreter for test trace analysis, called TETRA, takes interaction parameters into account and uses backtracking to determine whether the tree of possible execution histories defined by the specification includes a history which gives rise to the trace T of observed interactions.

Backtracking is necessary for all those occasions where the reference specification allows for non-deterministic choices which are not directly visible. These choices relate to one of the following cases:

(a) Execution of internal events, i.e. event i or internal interactions on hidden gates, which appear as alternatives within a choice of subexpressions.

(b) Idem, when appearing within a **choice** statement relating to alternative gates.

(c) Selection of a data value associated with a **choice** statement.

(d) Expansion of recursive definitions in case of non well-guarded expressions, i.e. alternatives not beginning with a visible interaction.

In the cases (a) and (b), only a finite (usually small) number of alternatives are available, and they can be explored by the trace analysis interpreter, through backtracking, without excessive loss of efficiency. For case (c), however, the number of possible choices is sometimes not even bounded, as for instance in the case of the choice of a natural number. The trace analyzer has to determine whether a suitable choice of value exists which makes the given trace acceptable by the specification.This problem is in general undecidable.

TETRA also has an option for checking the consistency of a test case, written in LOTOS, with a reference specification. For this purpose, first the scenarios, or branches, of the test case are identified, and then each scenario is checked against the specification. In order to limit the number of scenarios when the test case contains a loop, the loop is expanded only a limited number of times. Each scenario is checked like an interaction trace for conformance with the specification. The test case is consistent with the specification if each scenario with verdict "pass" or "inconclusive" conforms to the specification, and each scenario with verdict "fail" does not conform.

Since our initial experiments with TETRA [Boch 89j], the system has been improved in several respects (for more details, see [Boch 90h]). First, an on-line version has been built which analyses the trace of interactions one by one [Saba 90]. Second, certain options allow the processing of specifications with unlimited number of choices, as mentioned above, by arbitrarily limiting the depth of exploration of the tree of alternatives unless an observable interaction is involved.

In addition, the system has been modified in order to instantiate interaction parameters as late as possible. Input parameters of test case scenarios, as well as parameters of internal interactions, may not be defined directly. In this case, the analysis must be performed for all possible parameter values. The situation is similar for the variable representing the value selected by a **choice** statement. The improved system leaves such parameters or variables uninstantiated until their value can be determined from the constraints of the specification and/or subsequent observed input or output values. This avoids backtracking over all possible values, which would make the processing much more complex and inefficient.

Finally, we mention that TETRA provides error diagnostics in the case that an error has been found during trace analysis or test case consistency checking [Boch 89j]. During the optional error diagnostic phase, various fault hypothesis are checked for consistency with the given test trace (or scenario) and the specification. Each consistent hypothesis gives rise to a diagnostic message, which may be of the form "The second interaction is wrong and should be such and such", "The third interaction of the scenario should be absent", or "The verdict should be ’pass’ ". Each diagnostic message represents a possible interpretation for the reason of the problem.

3. Validation of X.25 test cases

We have validated the verdicts of a large number of the ISO/CCITT conformance test cases for the link layer of X.25 [ISO 8882] against a specification of the corresponding protocol, the LAP-B, as mentioned in the introduction. We used for this purpose an existing LOTOS specification of LAP-B which had already been validated through extensive simulations [Guer 89a]. This is a quite sizable specification of approximately 2500 lines of LOTOS code.

It is difficult to directly compare a test case written in TTCN with a protocol specification written in another formalism. In order to automate such a comparison, it is necessary that first, the protocol be specified formally, and second, that the language used to specify the test case, e.g. TTCN, be comparable with the language used for the formal protocol specification. In the case that the protocol specification is written in LOTOS, as assumed in this paper, we have to relate TTCN and LOTOS. A methodology for translating TTCN into LOTOS is described in [Dubu 90]. The translation turned out to be relatively straightforward.

As shown in Figure 3, the test cases are validated according to a simulated remote test architecture. We chose this architecture because the test cases of the ISO document were conceived to be executed within such an architecture. The test cases only describe the interactions with the lower tester (the upper tester is "empty"). The medium is a reliable full duplex queue.

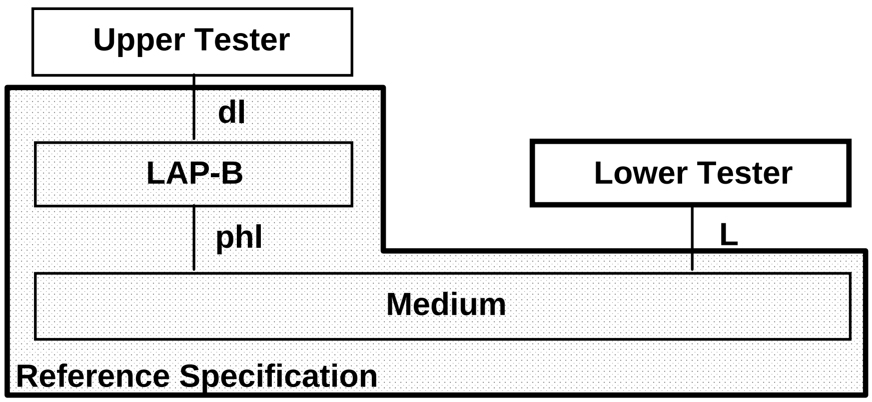


Figure 3. Remote Test Architecture

The complexity of the behaviour tree increases (dramatically) when we model the medium between the specification and the lower tester explicitly in the form of two FIFO queues in LOTOS. For simple test cases, the analysis time increases by a factor of up to ten. In more elaborated test cases, the analysis aborts due to a lack of memory. In order to obtain our results, we have bypassed the queue. Table 1 shows preliminary results for the validation time of some of the test cases. Note that TETRA is written in PROLOG and that these results were obtained on a Sun 4/330 with 32 Mb of RAM. The following paragraphs discuss particular aspects of our results.

Depth of the Behaviour Tree	Group DL1 (Disconnected)	Group DL2 (Link Disconnection)	Group DL4 (Information Transfer)
5	About 1 min	About 30 min	About 1 h
6	-	> 1 day	> 4 h
7	-	-	> 3 days

Table 1. Statistics on Validation Time

(a) Detected error in LAP-B test suite: We have found an error in one of the test cases of the original TTCN document. Test case DL1_306 says that the trace:

L ! DISC (P:=1)
   L ? DM [F=1]
      L ! UA (F:=1)
         L ? DISC [P=1]

should have a fail verdict, but it is accepted by the specification. We found that this sequence of actions is valid with respect to the LAP-B standard. We believe that the last test step of the test case DL1_306 (L ?Otherwise) should have an inconclusive verdict instead of a fail verdict.

(b) Detected error in the specification: Test case DL1_207 indicates an error in the specification which does not include all details concerning error processing. The branch:

L ! DISC (P:=1)
   L ? DM [F=1]
      L ! Hex (string:='03F??'H)
         L ? Otherwise

has a fail verdict, but is accepted by the specification (string '03F??'H is a SABM/P=1 with an non-empty information field).

(c) Preambles for arbitrary initial states: We were surprised to see that TETRA rejected certain branches of preambles which are considered valid according to the test suite

[ISO 8882]. For instance the branch:

L ! DISC [P:=1]

  L ? UA [F=1]

of the subtree DL1_STATE is not accepted by the LAP-B specification. Later we noticed that the ISO test cases do not necessarily assume that the IUT is initially in the disconnected state. For instance, the above branch is valid starting in the data transfer phase.

(d) Error diagnostics: As an option, TETRA provides diagnostics for locating the fault if an error is detected. While this facility works well for smaller specifications, we found that in our case the number of fault hypothesis (each indicated by a diagnostic message) was often too large to be useful. For instance, a fault in the first actions of a branch could lead to about hundred diagnostics messages.

4. Test result analysis for the ACSE protocol: Support for ASN.1

The parameter data structures of Application layer PDU's are usually described with ASN.1. The various FDT's use different notations for the description of data structures, which must also be used for the description of PDU's in a formal specification of the protocol in a given FDT. For the application of FDT's to the description of OSI Application layer protocols, it is therefore necessary to define a mapping from ASN.1 to the FDT language elements used for the description of data structures. Such mappings have been defined for Estelle [Boch 90f] and LOTOS [Boch 89h] in such a manner that the data structure definitions resulting from the ASN.1 translation are relatively readable. Example 1 shows the ASN.1 representation of a particluar ACSE APDU and the corresponding representation in the form of a LOTOS expression. A similar translation exists between data

Example 1: An ASN.1 ARRQ PDU and its LOTOS translation

The ACSE AARQ PDU expressed in ASN.1 :

```
{
    aarq {
        protocol-version                {version2(1)},
        application-context-name        "a",
        called-AP-title                 "d",
        called-AE-qualifier             "e",
        calling-AP-title                "b",
        calling-AE-qualifier            "c",
        user-information                {"f"}
    }
}
```

Corresponding LOTOS description :

```
ACSE_apdu(ACSE_apdu_generated_0(AARQ_apdu(
        protocol_version(Bit(0)+Bit(1)),
        application_context_name(a),
        called_AP_title(d),
        called_AE_qualifier(e),
        called_AP_invocation_id(Not_Present),
        called_AE_invocation_id(Not_Present),
        calling_AP_title(b),
        calling_AE_qualifier(c),
        calling_AP_invocation_id(Not_Present),
        calling_AE_invocation_id(Not_Present),
        implementation_information(Not_Present),
        user_information(f+ <>)
)))
```

type definitions in the two languages. Tools for the automatic translation of specifications and the generation of ASN.1 encoding and decoding routines in conjunction with related FDT tools have also been described. In order to demonstrate these tools, and at the same time demonstrate the automatic analysis of test results for an Application layer protocol, we have performed the experiment described below.

As shown in Figure 2, two ACSE protocol entities were connected over a (simulated) Presentation service. These implementations written in C were obtained through the automatic translation of an ACSE Estelle specification which contained the PDU definitions automatically obtained through translation from the original ASN.1 definitions found in the standard [ISO 8650]. The implementation also contained automatically generated PDU encoding and decoding routines.

The exchanged PDU’s were recorded into a trace file and at the same time analysed on-line by the TETRA tool using as the reference an ACSE LOTOS specification which also contained PDU definitions automatically obtained through translation from the ASN.1 definitions. Before being analysed by the TETRA tool, which accepts the analysed interactions in LOTOS action format, the ASN.1 encoded PDU’s were translated into the form of LOTOS expressions by the ASN.1 Decoder (see Figure 2) automatically generated by our ASN.1/LOTOS tool.

The ACSE protocol specification is relatively simple. Nevertheless, the length of the PDU definitions in ASN.1 is 100 lines. This is translated into 200 lines of Estelle type definitions and 1000 lines of LOTOS data type definitions. The total size of the ACSE LOTOS specification used by TETRA as a reference is 2400 lines. The control part is small (approximatively 200 lines) which gives a simple behaviour tree. The on-line version of TETRA validates each interaction in matters of seconds. A certain number of test scenarios were run and the resulting traces of PDU’s were analysed. TETRA detected one error in the implementation and one error in the specification. In all those cases, the diagnostic part located the erroneous behaviours. Example 2 shows an execution trace of TETRA which points out an error in the implementation.

Example 2: Execution trace of TETRA (on-line version)

Observed new interaction ->
P !Input:IO !PCONind : primitive !ACSE_apdu(AARQ_apdu(**protocol_version(Bit(0)+Bit(1))** ,
application_context_name(a), called_AP_title(d), called_AE_qualifier(e), called_AP_invocation_id(Not_Present),
called_AE_invocation_id(Not_Present), calling_AP_title(b),
calling_AE_qualifier(c),calling_AP_invocation_id(Not_Present), calling_AE_invocation_id(Not_Present),,
implementation_information(Not_Present), user_information(f+ <>))) : ACSE_apdu

... Valid ...

Observed new interaction ->
P !Out : IO !PCONrspAcceptance : primitive !ACSE_apdu(AARE_apdu(**protocol_version(Bit(1))**,
application_context_name(a), **result(0)**, Associate_source_diagnostic(acse_service_user(0)),
responding_AP_title(Not_Present), responding_AE_qualifier(Not_Present),
responding_AP_invocation_id(Not_Present), responding_AP_invocation_id(Not_Present),
implementation_information(Not_Present), user_information(Not_Present)))) : ACSE_apdu

... Invalid ...

--- Next Possible Actions -- ---

<1>- P !Out:IO !PCONrspUserRejection:primitive
!ACSE_apdu(ACSE_apdu_genere_1(AARE_apdu(protocol_version(**protocol_version(Bit(1))**,
application_context_name(a), **result(Succ(0))**, Associate_source_diagnostic(acse_service_provider(Succ(Succ(0)))),
responding_AP_title(Not_Present), responding_AE_qualifier(Not_Present),
responding_AP_invocation_id(Not_Present), responding_AP_invocation_id(Not_Present),
implementation_information(Not_Present), user_information(Not_Present))) : ACSE_apdu

<2>- P !Out:IO !PUABreq : primitive !ACSE_apdu(ABRT_apdu(abort_source(Succ(0)),
user_information(Not_Present))) : ACSE_apdu

<3>- ...

In this example, we wanted to test if the called implementation reacts properly when an AARQ APDU is sent with an unsupported protocol version (first interaction of Example 2). In this test scenario, the called implementation should respond with a rejection, i.e. AARE APDU (Result = 1), but instead, it responds with result = 0 (acceptance, second interaction of Example 2). In a first analysis phase, TETRA diagnoses an invalid interaction from the implementation. In a second phase, TETRA provides a list of actions that could have taken place instead of the erroneous behaviour. This includes the AARE APDU (Result = 1) shown as last interaction in Example 2 (note that "Succ(0)" is the notation for 1 in LOTOS).

One of our test cases (collision of two release requests) highlights a problem with remote and distributed testing architectures, which may also apply to local testing. According to the specification, the IUT may generate a RLRQ (release request) PDU just before receiving a RLRQ from its peer, as shown for the initiator in Figure 3, but not after it has received such a PDU. However, the latter sequence may be observed by an observer that resides at the peer site, or somewhere between the two communicating entities, as shown by the dashed line in Figure 4.

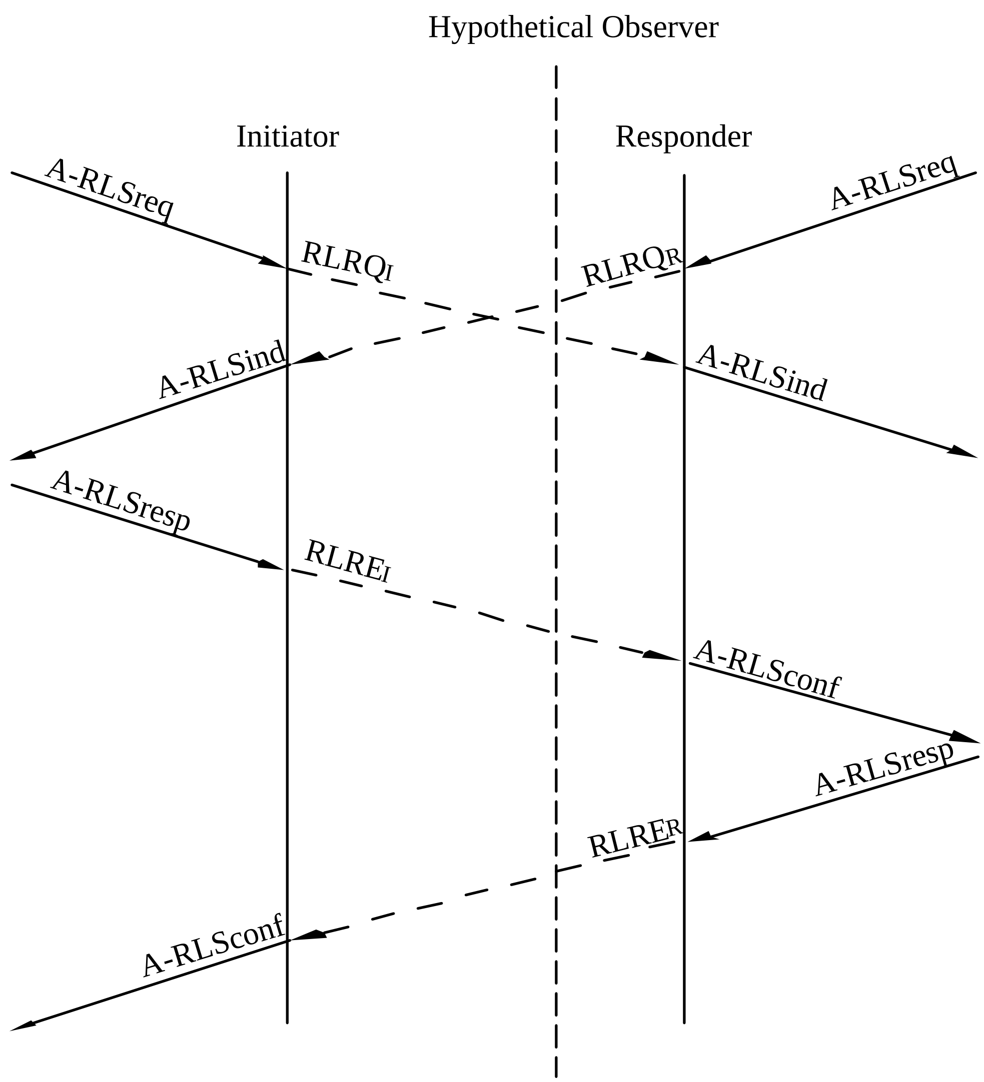


Figure 4. Sequence of ASPs and APDUs in the release collision scenario

Although we performed local observation, as shown in Figure 2, we sometimes observed the wrong sequence of PDU's. This is due to the fact that our point of control and observation (PCO) was at the Presentation service interface, and included queues for PDU buffering within the ACSE entity implementation. It seems that this is a quite normal situation. Unfortunately, this makes the observation of certain timing and ordering errors, such as the one above, difficult to observe and diagnose [Dsso 90].

The presence of such queues between the PCO and the state machine of the protocol implementation can be taken into account during the test result analysis by including queues in the reference specification used for the analysis, as discussed for X.25 in Section 3. We did not do this exercise in the case of ACSE.

5. Concluding discussion

This paper describes our experience with a trace analysis tool, called TETRA, for the validation of OSI conformance test cases, and for the analysis of conformance test results in respect to the reference specification. Another experience involving the validation of a Transport protocol in respect to the OSI Transport service written in LOTOS is described in [Saba 90]. These experiences show that TETRA is a practical tool for the analysis of test result and for the validation of test verdicts. Although there are still some limitations when the behaviour tree of the specification is complex, it is possible to handle real-life protocol specifications which cover several thousand lines of LOTOS code. We hope that further optimizations of the analysis algorithm will lead to an improved tool which can handle all the test cases which are encountered in OSI conformance testing. The diagnostics facility should also be improved.

The errors found during our experiences indicate, as could be expected, that even well studied specifications still contain a few errors. This shows that the automatic checking of conformance test cases in respect to the corresponding protocol specification is a useful activity for increasing the confidence in the OSI specifications. It is important to note that the automation of this activity is only possible when a formal specification of the protocol is available. Unfortunately, at present, there are only few formal specifications of OSI protocols or services that have been generally recognized to faithfully represent the OSI standards.

Acknowledgements: The authors would like to thank Fabrice Lavier for writing the first draft of the ACSE LOTOS specification, O. Bellal for helpful discussions concerning the development of TETRA, L. Logrippo for the understanding of the LOTOS specification for LAP-B and Pierre Mondain-Monval for helpful discussions on protocols of the Application layer. The projects described in this paper were mainly funded under the IDACOM-NSERC-CWARC industrial research chair on communication protocols. Financial support from the Ministry for Education of Quebec is also gratefully acknowledged.

References

[ASN1] ISO 8824 (1987) *Specification of Abstract Syntax Notation One (ASN.1)*.

[Boch 87c] G. v. Bochmann, *Usage of protocol development tools: the results of a survey*, (invited paper), 7th IFIP Symposium on Protocol Specification, Testing and Verification, Zurich, May 1987, pp.139-161.

[Boch 89h] G. v. Bochmann and M. Deslauriers, *Combining ASN.1 support with the LOTOS language*, Proc. IFIP Symp. on Protocol Specification, Testing and Verification XI, June 1989, North Holland Publ.

[Boch 89j] G. v. Bochmann and O. Bellal, *Test result analysis in respect to formal specifications*, Proc. 2nd Int. Workshop on Protocol Test Systems, Berlin, Oct. 1989, pp. 272-294.

[Boch 90f] G. v. Bochmann, D. Ouimet and G. Neufeld, *Implementation support tools for OSI application layer protocols*, submitted to Software Practice and Experience.

[Boch 90g] G. v. Bochmann, *Protocol specification for OSI*, Computer Networks and ISDN Systems 18 (April 1990), pp. 167-184.

[Boch 90h] G. v. Bochmann, O. Bellal, F. Saba and M. Dubuc, *Automatic test result analysis*, in preparation.

[Dubu 90] M. Dubuc, G. v. Bochmann, O. Bellal and F. Saba, *Translation from TTCN to LOTOS and the validation of test cases*, submitted to FORTE ’90 (IFIP), Madrid, (November 1990).

[Este 89] ISO 9074 (1989), *Estelle: A formal description technique based on an extended state transition model*

[Guer 89a] D. Gueraichi, *Derivation of test cases for LOTOS LAP-B specification*, M.Sc. Thesis, University of Ottawa, 1989.

[Guil 89] R. Guillemot and L. Logrippo, *Derivation of useful execution trees from LOTOS by using an interpreter*, in "Formal Description Techniques", (Ed.) K.J.Turner (Proceedings of the First International Conference on Formal Description Techniques Stirling, Scotland, 6-9 September, 1988), pp.311-325.

[ISO 8650] ISO 8650 (1988) *Protocol Specification for the Association Control Service Element (ACSE)*.

[ISO 8882] ISO DP 8882-2 (1989) *X.25 DTE Conformance Testing, Part 2: Data Link Layer Conformance Tests.*

[Logr 88] L. Logrippo and e. al., *An interpreter for LOTOS: A specification language for distributed systems*, Software Practice and Experience, Vol. 18 (4), pp.365-385, April 1988.

[Loto 89] ISO 8807 (1989), *LOTOS: a formal description technique*.

[Saba 90] F. Saba, *Validation en ligne de traces d'execution appliquee au protocole de Transport*, M.Sc. thesis, Universite de Montreal, summer 1990.

[Sari 89c] B. Sarikaya, *Conformance Testing: Architectures and Test Sequences*, Computer Networks and ISDN Systems 17 (1989), pp. 111-126.

[SDL 87] S. X. CCITT, *Recommendation Z.100*, (1987).