ETSI TECHNICAL REPORT

ETR 049

October 1992

Source: ETSI TC-ATM

Reference: DTR/ATM-1006

ICS: 33.020

Key words: Test specification methods

Advanced Testing Methods (ATM); State of research in the area of formal test specification methods

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Foreword 5

PART 1: The fundamental problems in conformance testing 7
- 1.1 Protocol specification 7
  - 1.1.1 Some basics about protocols; the OSI principles 7
  - 1.1.2 Formalism versus non-formalism 8
- 1.2 Protocol testing 9
  - 1.2.1 Theory behind testing 9
    - 1.2.1.1 Assumptions about testing 9
    - 1.2.1.2 One definition of "conformance" 10
  - 1.2.2 A framework for conformance testing 11
    - 1.2.2.1 Terminology and common practice 11
    - 1.2.2.2 Various test types 12
    - 1.2.2.3 The standardised testing methods 13
    - 1.2.2.4 The Ferry principle 16
    - 1.2.2.5 The "Ferry Clip" approach 17
- 1.3 Generation of abstract test suites 18
  - 1.3.1 Finite State Machine formalism 18
  - 1.3.2 Problems with Extended Finite State Machines 19
  - 1.3.3 Problems with non-deterministic specifications 19
- 1.4 Establishing a testing environment 21
- 1.5 Test execution and verdict 21

PART 2: Research approaches in conformance testing 22
- 2.1 Standardised specification languages 22
  - 2.1.1 Specification and Description Language (SDL) 22
  - 2.1.2 LOTOS 23
  - 2.1.3 Estelle 24
  - 2.1.4 Research in common semantics 24
- 2.2 Test suite generation approaches 25
  - 2.2.1 Transition tour method 25
  - 2.2.2 Distinguishing sequences 27
  - 2.2.3 Characterising sequences 29
  - 2.2.4 Unique input/output sequences 29
  - 2.2.5 The canonical tester approach 31
    - 2.2.5.1 An overview of the approach 31
    - 2.2.5.2 The CO-OP method 32
  - 2.2.6 The extended EFSM (chart) approach 34
  - 2.2.7 The Asynchronous Communication Tree (ACT) approach 38

PART 3: Dictionary 39
- 3.1 Abbreviations 39
- 3.2 Vocabulary of research related terms used in this paper 41
- 3.3 References used in this paper 41
- 3.4 Research centres, principal researchers and approaches 43
  - 3.4.1 AT&T Bell Laboratories, USA 43
  - 3.4.2 National Institute of Standards, USA 44
  - 3.4.3 National Physical Laboratory, United Kingdom 45
  - 3.4.4 University of British Columbia, Canada 45
  - 3.4.5 University of Montreal, Canada 46
  - 3.4.6 University of Ottawa, Canada 47
  - 3.4.7 University of Twente, The Netherlands 48
  - 3.4.8 PTT Research, The Netherlands 50

History 52

Blank page

Foreword

ETSI Technical Reports (ETRs) are informative documents resulting from ETSI studies which are not appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or I-ETS.

This ETR was produced by the Advanced Testing Methods (ATM) Technical Committee of the European Telecommunications Standards Institute (ETSI). It presents an overview of the current research problems and approaches in the field of Protocol Conformance Testing by, firstly, introducing the general problems and then by introducing the current research approaches that are applied to tackle the problems.

This ETR has been produced as a basis for standardisation work within ETSI TC-ATM with the purpose of presenting a background for discussions about standardisation opportunities in the field. An interesting question is how mature the technology is for standardisation from a scientific as well as from an industrial point of view. This ETR is concerned with the scientific aspect. It intends to present the relevant basic problems and the scientific approaches to tackle them. For those interested in deeper studies, a listing is provided which tries to cover the most relevant actors, as well as papers produced, in the field.

The structure of this ETR tries to mirror this with the first element looking at fundamental problems, the second element detailing how those problems are treated by researchers and, finally, who these researchers are, where they are and what results have they produced.

Blank page

PART 1: The fundamental problems in conformance testing

1.1 Protocol specification

1.1.1 Some basics about protocols; the OSI principles

The seven-layer Open Systems Interconnection (OSI) reference model has been defined by ISO to represent the interconnection of heterogeneous systems. Each layer entity in this model, called (N)-Entity, provides certain services, called (N)-services, to its upper entity (figure 1). In order to provide these services to the (N+1)-Entity, an (N)-Entity uses the services provided by its lower entity (i.e. the (N-1)-Entity). An (N)-Entity together with the services provided to the (N+1)-Entity are called an (N)-service provider. The abstraction of each entity and its services allows flexible implementations of the OSI reference model protocols, since the implementation details of the upper and lower entities are isolated.

Two peer (N)-Entities use (N)-Protocol Data Units (abbreviated (N)-PDUs) to communicate with each other through an (N)-Service provider. An (N)-protocol is the set of rules and conventions that define the communication between the two peer entities.

Each (N)-PDU that becomes known to an (N)-protocol can - at each state of the protocol - be classified to one of the set {valid, inopportune, illegal}. A PDU is valid at a state if it is specified to occur at that particular state of a protocol. A PDU is inopportune at a state if it is a member of the total set of PDUs, but is not defined at that state. A PDU is illegal if it does not exist in the total set of PDUs.

Thus a specification ex ante can only contain valid PDUs. The presence of an inopportune PDU is a sign that the sending (N)-entity is out of phase with the receiving (N)-entity. Illegal PDUs do not meet the syntax requirements of the (N)-protocol, and are signs of a more serious error than being out of phase.

A complete (N)-protocol specification is defining all actions that have to be taken for the cartesian product of {states} x {valid, inopportune and illegal PDUs}. Thus a complete protocol specification does not contain undefined states.

However, there are two problems involved:

1) specifying all inopportune PDUs leads to very large specifications;

2) illegal PDUs cannot be specified, because if so done, they are immediately legal in a logical sense. (Although not desired). Therefore, illegal PDUs are covered under a generic "otherwise" Clause.

An (N)-Entity communicates with its upper and lower entities by means of Abstract Service Primitives (ASPs). It communicates with its upper (N+1)-Entity by means of (N)-ASPs, and it communicates with its lower (N-1)-Entity by means of (N-1)-ASPs. In an (N)-Entity implementation, the ASPs are implemented at the Points of Control and Observation (PCOs) at the lower and upper service boundaries (see figure 1). These two sets of ASPs define the externally controllable and observable behaviour of the (N)-Entity. The objective of the conformance testing of an (N)-Entity implementation is to determine whether the behaviour of the implementation is in accordance with the protocol specification. During the conformance testing, the ASPs that are defined as inputs are sent to the (N)-Entity implementation and the ASPs generated by the implementation are observed. This approach in conformance testing, which considers only the externally visible behaviour of an (N)-Entity implementation, is called black box testing.

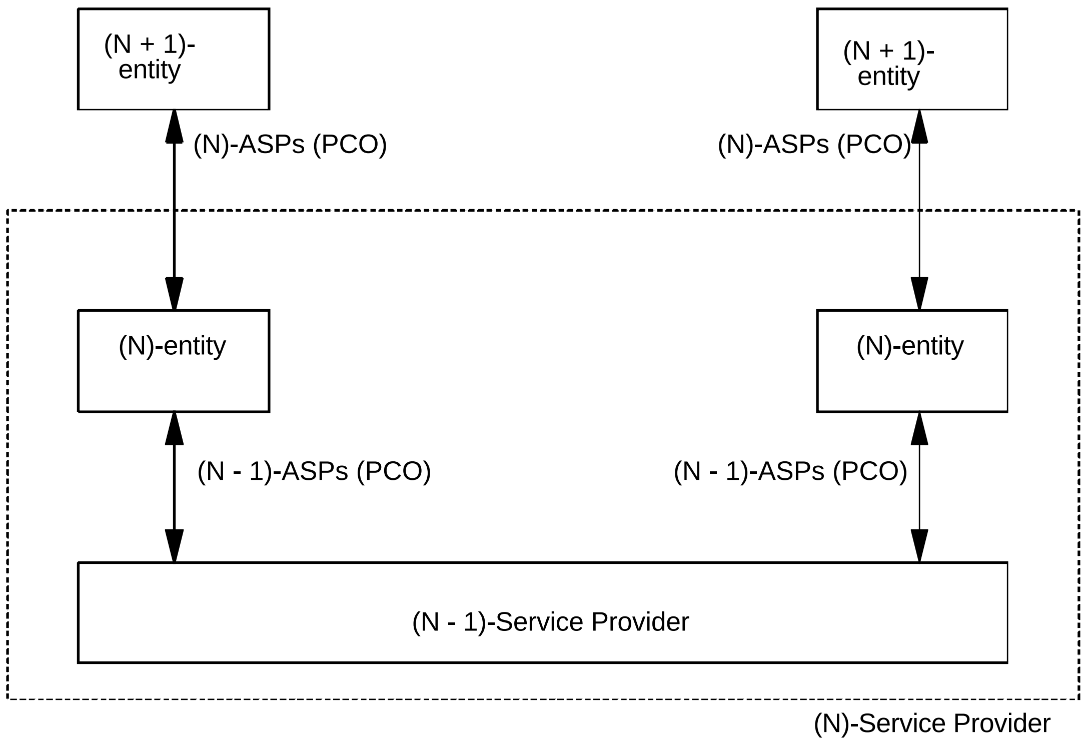

Figure 1: Abstraction of an (N)-entity in a multi-entity OSI reference model

1.1.2 Formalism versus non-formalism

There has been some debate whether a protocol should be specified formally or informally. "Formally" means using a language with well defined syntax and semantics. The syntax may be separated into concrete and abstract syntax. The abstract syntax defines the syntactical entities and their relationships. The concrete syntax defines how the syntactical entities are represented as character strings (or graphical symbols). Thus it is possible to decide whether a certain character string is a well-formed formula in a particular concrete syntax. The formula may however, although well formed, be meaningless from a semantical point of view. A language with well defined abstract syntax may have several equivalent concrete syntaxes. It may for instance be allowable to have keywords in different languages (English, French, etc...). Although this has been tried for a language like COBOL, nowadays it is avoided. There is a tendency to keep to one concrete syntax defined in terms of the standard ASCII character set.

The definition (CCITT Recommendation Z.110 [1]) of a Formal Description Technique (FDT) is the following.

> "A FDT is a specification method based on a description language using rigourous and unambiguous rules both with respect to developing expressions in the language (formal syntax) and interpreting the meaning of these expressions (formal semantics)".

It is obvious that a formal language can only express aspects of a specification allowed by the language. Every other aspect is undefined, from the language point of view.

There is an intermediate language category, semi-formal languages, which have well defined concrete syntax but informally defined semantics. It may work at the expense of quite verbose informal semantic specifications.

The proponents for non-formal languages argue that the language will not restrict the ability to express aspects of a specification. Although it is not a valid argument, it may be easier to have different people agree on a specification with different possible interpretations. The main problem with informal specifications is, in fact, their ambiguity, i.e. different interpretations may be possible. This leads to interworking problems when implementations based on two different implementations have to interwork.

The definition (CCITT Recommendation Z.110 [1]) of an informal description is the following.

> "A natural language description is an example of an informal description technique using one of the languages used by CCITT to publish Recommendations. It may be supplemented with mathematical and other accepted notation, figures, etc".

The general consensus in standardisation bodies is that formal specification languages should be used, as far as the semantics of the languages allow it. Only when formal languages are applied, can unambiguous test data be produced.

The official CCITT and ISO approach to the use of FDTs are formulated in CCITT Recommendation Z.110 [1] and ISO Resolution ISO/IEC JTC 1/N 145 respectively.

1.2 Protocol testing

1.2.1 Theory behind testing

1.2.1.1 Assumptions about testing

(Acknowledgement: part of this text is derived from "Conformance testing based on SDL specifications SDL'89: The language at work"; Kristofferson, Finn [2] and remarks received from Jan Kroon, RNL.)

Behind the idea of testing lie the following assumptions:

1) The behaviour of the system S (the specified system) is fully known. This includes possible silent transitions.

2) The behaviour of the system I (the implemented system) is a priori fully unknown. Any knowledge of its behaviour shall be collected through probes into the system, in a controlled manner.

3) The system I is considered to be stable in the sense that it is not changing its behaviour rules.

Although the last assumption cannot be guaranteed for a particular I, it would not be meaningful to perform any test of a system that silently changes its behaviour.

This is not violating the assumption that silent transitions that are permitted according to S will occur also in I.

There may be stronger or weaker equivalence relationships between S and I, represented as Labelled Transition Systems (LTS). The equivalences work if both S and I are defined as LTSs, i.e. I is not an implemented but unknown system in the sense of 1), 2), and 3) above.

The testing literature is presenting the following equivalences:

- bi-simulation;
- failure;
- observation;
- testing;
- trace.

Bi-simulation equivalence means that systems mutually simulate the behaviour of each other. Thus S and I are bi-simulation equivalent if S simulates I and I simulates S.

Failure equivalence holds between I and S if I conforms to S (see definition under subclause 1.2.1.2) and I does not contain any transition that is not specified in S. That means briefly that I fails exactly where S fails. Failure sets (communications that cannot happen) are equal at all nodes of LTSs.

The latter three are of primary interest. Observation equivalence is stronger than testing equivalence, which in turn, is stronger than trace equivalence. Thus, if observation equivalence is proven, then the others are proven by implication. Testing equivalence implies trace equivalence.

The problem with observation equivalence from a testing point of view is that it cannot be decided by testing. Observation equivalence is so strong that it is undecidable for unknown systems. To decide observation equivalence requires full insight into all possible silent transitions of the implemented system, and full insight means that the behaviour of the implemented system is, a priori, fully known. Therefore testing is not needed at all, since stronger analytical methods can be applied (for details see "Calculus of communication systems - lecture notes in Computer Science" [3]).

Two systems S and I are testing equivalent only under the two following conditions:

a) if I may satisfy an observer, S may also satisfy the observer;

b) if I shall satisfy an observer, S shall also satisfy the observer.

A system may satisfy an observer if there exists at least one test that runs successfully. A system shall satisfy an observer if all tests run successfully. In practice it means, given any test, if it runs on S it shall also run on I, and if it runs on I then it shall also run on S. As will be shown later, testing equivalence between I and S demands not only that I conforms to S but also that the set of traces of I is a subset of the set of traces of S. (In fact, the sets shall be identical due to the conformance requirement, "Conformance testing based on SDL specifications SDL'89: The language at work" [2]).

Two systems are trace equivalent if the set of traces they can produce are identical. Trace equivalence is the weakest because it abstracts away silent transitions. A silent transition is recognisable in a LTS as a situation where a node is followed by two different subtrees through the same vertex label. This means that the LTS is "choosing" one of the subtrees based on the same stimuli, or no stimuli at all (if the subtrees are identical, then one of the subtrees can be deleted without semantical change of the LTS).

From this follows that for a test sequence, applied to systems I and S the following is known a priori:

1) if S allows for silent transitions, I can contain silent transitions at the same nodes, and consequently testing equivalence shall be looked for;

2) if S does not allow for silent transitions, and I contains incorrectly silent transitions and I is fair during test, and the test is repeated enough, then the incorrect silent transitions will eventually be detected;

3) if S does not allow for silent transitions, and I contains incorrectly silent transition and I is unfair during test, then the incorrect silent transitions will not be detected, regardless the length of the test.

A system is fair if, with a free choice of paths, it can eventually choose both of the alternatives. A system is unfair if, with a free choice of paths, it shall never choose one of the alternatives.

From this it can be concluded that nothing can be done about an unfair I containing incorrect silent transitions. These errors will only show up occasionally, since there is no relationship between the number of test runs and the probability of error detection. Normally, with a choice of 2 equally frequent alternatives, the likelihood of one choice getting undetected is $1/(2^n)$ where n is the number of test runs.

One interesting question is whether there exists a test derived from an S that allows for silent transitions? The answer is yes; all alternative traces are generated. If there is no choice in S, the test does not include choice. When there is a choice in S, the test includes a choice, and consequently I shall succeed with either of the alternatives (no test sequence can be derived from I, so that problem never occurs).

1.2.1.2 One definition of "conformance"

According to Brinksma in "A theory for the derivation of tests Proceeding of the IFIP WG 6.1 VIII International Symposium on Protocol Specification, Testing and Verification (1988)" [4], a process I conforms to a process S, if and only if, a deadlock that occurs in I also occurs in S, tested with the same trace that is defined for S.

Or in other words, take any trace σ of S (derived from S) member of the set Tr(S). This trace σ may deadlock in S, if S allows for silent transitions. If that is the case, it may also deadlock in I.

If S does not allow for silent transitions, then σ shall always terminate in S, and it shall also terminate in I.

The definition of conformance does not cover traces of I that are not members of Tr(S). Intuitively, it can be realised that this restriction is necessary, because the behaviour of I is, a priori, unknown, and, therefore, the potential set of traces of I is infinite. Any trace taken by chance can be applied.

Pragmatically, we can also say that S specifies the minimal behaviour of any system I that claims to fulfil S. Whatever I can do more is irrelevant for the re-usability problem. Therefore a user need not bother with it.

Testing equivalence is thus stronger than conformance relation. If a system I is testing equivalent to S, it is also conforming. However testing equivalence between I and S also demands that the set of traces of I is a subset of the set of traces of S. In fact, the sets shall be identical due to the conformance requirement.

From a pragmatical viewpoint, testing equivalence is an overkill, since it excludes the use of various implementations with extra functionality.

Since the behaviour of a conforming I that is outside of the scope of testing is unknown and possibly infinite, whatever I will do with stimuli outside of the scope of the specification is undefined and unknown. This is, however, a problem of robustness that cannot be solved through conformance testing. It is up to the user to make sure that undefined stimuli does not reach the implementation.

1.2.2 A framework for conformance testing

1.2.2.1 Terminology and common practice

The basic framework for conformance testing in practice is standardised in ISO/IEC 9646 [5] "Information processing systems - Open Systems Interconnections - OSI conformance testing methodology and framework - Parts 1 to 5". The text below is based on ISO/IEC 9646 [5].

An (N)-Entity implementation that is being tested is called the Implementation Under Test (IUT). Similarly, a system of which one or more layers are under test is referred to as the System Under Test (SUT). Conformance testing of an IUT is performed as black box testing by means of a test suite that simulates the peer (N)-Entity. An abstract test suite defines the functionalities of the test suite without giving any implementation details. In an abstract test suite, the Lower Tester (LT) is responsible for the control and observation of the (N-1)-ASPs at the PCO either below the IUT or at the remote end. The lower tester is the peer entity of the IUT. The Upper Tester (UT) controls and observes the (N)-ASPs at the upper service boundary of the IUT. Typically, the LT is considered to be the master of the UT. The co-ordination between the upper and lower testers' actions are provided by the Test Co-ordination Procedures (TCPs). In some architectures, a specific protocol, called a Test Management Protocol (TMP) that uses specific PDUs (TM-PDUs), is defined for the upper and lower tester co-ordination.

When an (N)-Entity is installed within a multi-layer product, due to design restrictions or by choice, some (or all) of the ASPs become externally inaccessible at the PCOs. The availability of the ASPs in an IUT defines the degree of controllability and observability capabilities that the tester has on an IUT. The conformance testing standards define various methods of abstract test suite organisations based on the accessible PCOs in an IUT, which is discussed in subclause 1.2.2.3.

Test events are atomic interactions between the IUT and an upper or lower tester. The test events are ordered and described in an abstract conformance test suite. Test events are grouped into test steps, which in turn are grouped into test cases. Each test case is a conceptually self-contained test unit that is based on a test purpose. A test purpose describes the objective of the corresponding test case. The test cases are ordered into test groups, based on a logical ordering of the execution of test cases. Test groups may recursively contain test groups. The highest level set of test groups constitutes a test suite.

In each abstract test case, there are three major components: test preamble, test body and test postamble. The preamble defines the necessary steps to bring the IUT into the desired starting state of the test case. The test body defines the test steps that are needed to achieve the test purpose. The PDU exchanges to verify the new state of the IUT before the test case completion are also part of the test body. The test postamble is used to put the IUT into a stable state after a test body is run. In order to achieve efficient run time, one may execute the preambles and postambles conditionally (i.e. only if a previous test fails or an individual test is to be run) by concatenating the test bodies in a manner of tour.

An abstract test notation, called the Tree and Tabular Combined Notation (TTCN) is defined to represent the abstract test suites in a standard fashion. In TTCN, the behaviour and actions of an IUT during the execution of the test cases (i.e. sending and receiving various PDUs defined by the test cases) are described in a tree format. Research is in progress to generate machine-executable versions of the abstract test suites that are written in TTCN.

Before a conformance test is conducted, the options and capabilities that are implemented in an IUT have to be known by the test laboratory in order to select the applicable test cases and eliminate the irrelevant ones. This information is described in a Protocol Implementation Conformance Statement (PICS). A test laboratory additionally requires a Protocol Implementation eXtra Information for Testing (PIXIT) in order to match the testing environment to the capabilities of an IUT.

Test suite generation can be automated to a large extent based on a model of a protocol expressed as an Finite State Machine (FSM). It is a precondition that the specification can take the form of a directed graph that is free of deadlocks and livelocks. A deadlock is a lack of continuation, and a livelock is a never-terminating loop.

Test suites based on directed graphs will trigger transitions when applied to an IUT. It is desirable to generate test suites that are economical, i.e. will cover relevant transitions by a minimal number of test cases. For that purpose it is important to minimize redundancy. The way to minimize redundancy is to merge preamble and postamble such that the postamble for case "i" is merged with preamble for case "i + 1". This problem turns out to be a search (or tour) problem, where the problem is to find a tour of the directed graph that fully covers all transitions with a minimal number of node visits (see details in subclause 2.2.1).

Once the test suite is generated, each test case follows the following procedure:

Step 1: Put the IUT in the assumed state s_i;

Step 2: Apply input a_k and observe output o_l;

Step 3: Verify that the new state is the assumed s_j.

Two principal problems are involved here. The first problem is <u>controllability</u>, to really be sure that the IUT is in a state that corresponds to the state s_i in the FSM. The second problem is <u>observability</u>, to observe output o_l and/or verify that s_j was actually reached. The problem of controllability increases dramatically where τ-events in the IUT can occur e.g., as internal actions caused by non-visible timers. The problem of observability can be very hard if the IUT is not revealing in which state it is.

Testing is greatly facilitated if the specification includes a function that for each state responds to probes by sending back a status message. The sole purpose of the status message is to identify the state of the computation that matches a named node in the directed graph of the specification. Thus the tester need not use Unique Input/Output (UIO) sequences or other techniques to find if the correct node was visited.

1.2.2.2 Various test types

Conformance testing is, in principle, any black box testing that examines to what extent an implementation conforms to the specification. Therefore, the extent of the specification delimits the extent of the conformance test. The maximal conformance test is a test that fully examines all requirements in the specification. A test that examines all requirements of a specification has 100% coverage. This is, due to technical and economical realities, seldom achieved.

It is however, customary to perform simple and cheap tests before more complex and expensive tests are applied. If the IUT fails in the simple test, there is no use in applying a complex test.

The following test types are generally defined:

- Basic Interconnection Test.

  A limited test to ensure that the IUT can establish a basic interconnection. If this test fails, more complex tests cannot be applied.

- Capability Test.

  A test, essentially based on the description of the IUT, that the static conformance requirements are met. These requirements include necessary options, necessary PDUs, parameter values, etc.

- Behaviour Test.

  This test has the purpose of testing that the IUT behaves according to the dynamic conformance requirements. For this purpose it is necessary to put the IUT in a particular state and input particular PDUs. The test will reveal the real response comparable to that of the specification. It is therefore assumed that the specification is complete (see subclause 1.1.1).

- Conformance Resolution Test.

  This is a particular case of behaviour test, where specific states are tested. These states are those that are known to be problematic.

- Performance Test.

  If the specification includes performance requirements, this is the kind of test for it. It includes the generation of a real-time situation for the IUT, combined with traffic generation and performance measurement. It is assumed that the behaviour is already tested to be correct.

- Stress test.

  A test that extends the traffic to the extreme, in order to study behaviour under heavy load.

1.2.2.3 The standardised testing methods

ISO and CCITT have defined the basic concepts related to the conformance testing of OSI protocol implementations. Standardisation efforts, which resulted in a 5 part Standard (ISO/IEC 9646 [5] which is still being refined), have been mainly devoted to the issues regarding the implementation of test suites in a test laboratory. The subjects that are covered in ISO/IEC 9646 [5] are as follows:

Part 1: General concepts of conformance testing;

Part 2: Abstract test suite specification;

Part 3: The Tree and Tabular Combined Notation (TTCN);

Part 4: Test realisation;

Part 5: Requirements on test laboratories and clients for conformance assessment process.

The formal methods for generation of conformance tests are not included in this effort since that problem area is still in the research domain.

ISO/IEC 9646-2 [5] defines abstract test methods for test suite specifications. These methods are independent of the implementation details of a specific testbed and the definitions are based on the availability of the ASPs at the PCO. For single-layer testing, each method is defined as follows:

- Local Testing Method.

  The PCOs are defined at the upper and lower service boundaries of the IUT (figure 2) assuming that (N)-PDUs, (N)- and (N-1)-ASPs can be controlled and observed by the upper and lower testers. The UT is located within the test system. The service provider is local.

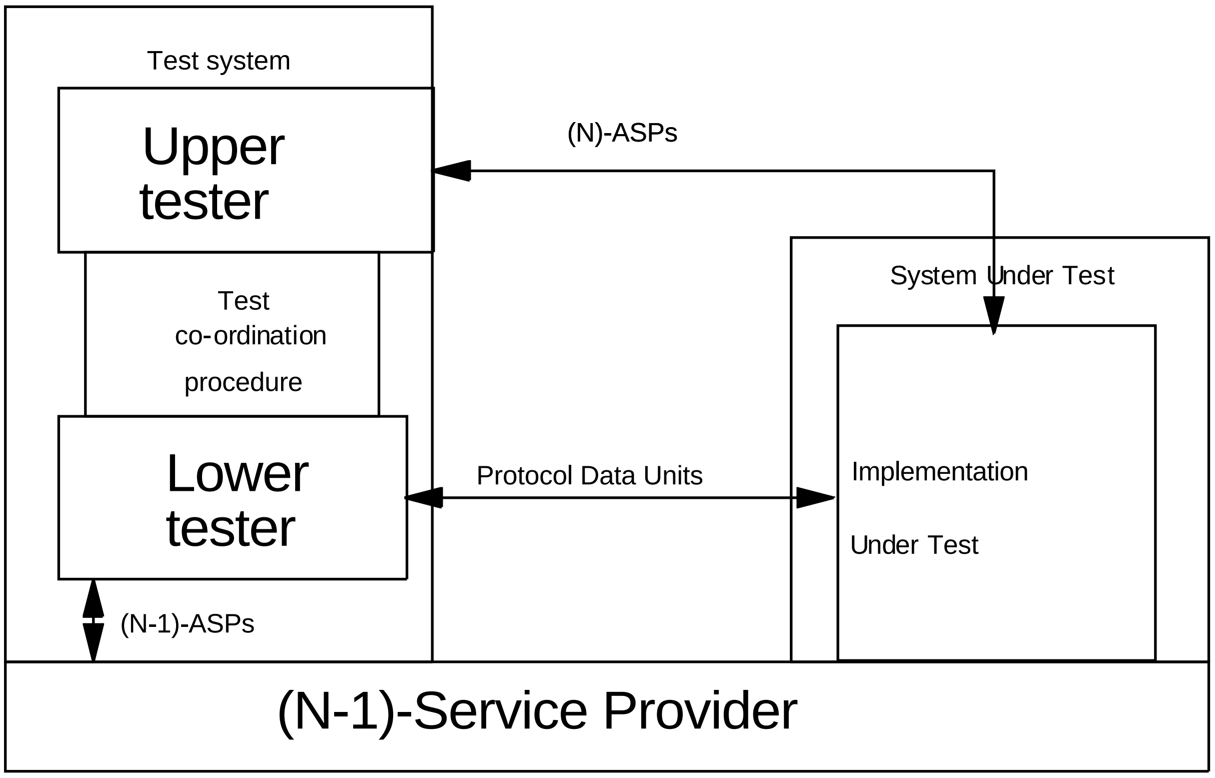

Figure 2: Local single-layer testing method

- Distributed Testing Method.

  In this method, the IUT does not have a PCO at the lower service boundary. Therefore, (N)-ASPs and (N)-PDUs are controlled and observed indirectly at the remote end of the (N)-Service Provider (figure 3) allowing the upper and lower testers to reside in physically separate locations. The UT is located within the system under test. There is no final solution for design of test co-ordination procedure.

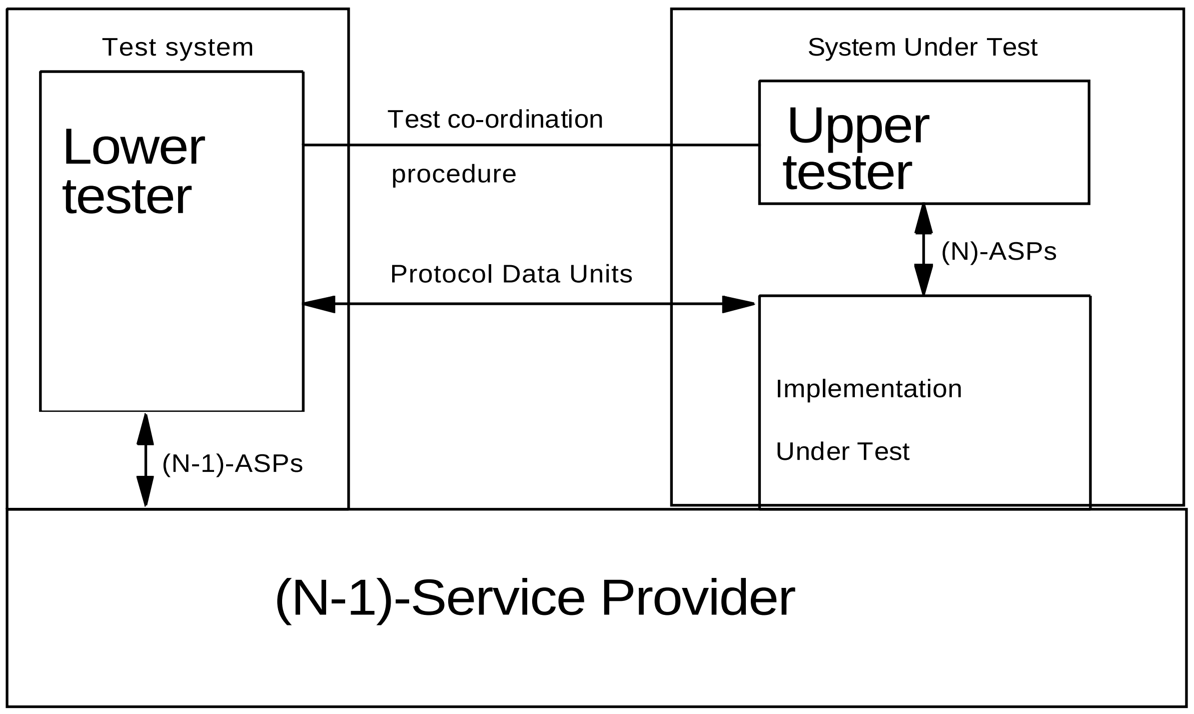

Figure 3: Distributed single-layer testing method

- Co-ordinated Testing Method.

  This method is an enhanced version of the distributed method. Here, the actions of the UT (i.e. control and observation of (N)-ASPs) are co-ordinated by a TMP. The UT is an implementation of the TMP. The TMP uses TM-PDUs for the co-ordination between the upper and lower testers (see figure 4).

  NOTE: No PCO at the upper service boundary of the IUT is required.

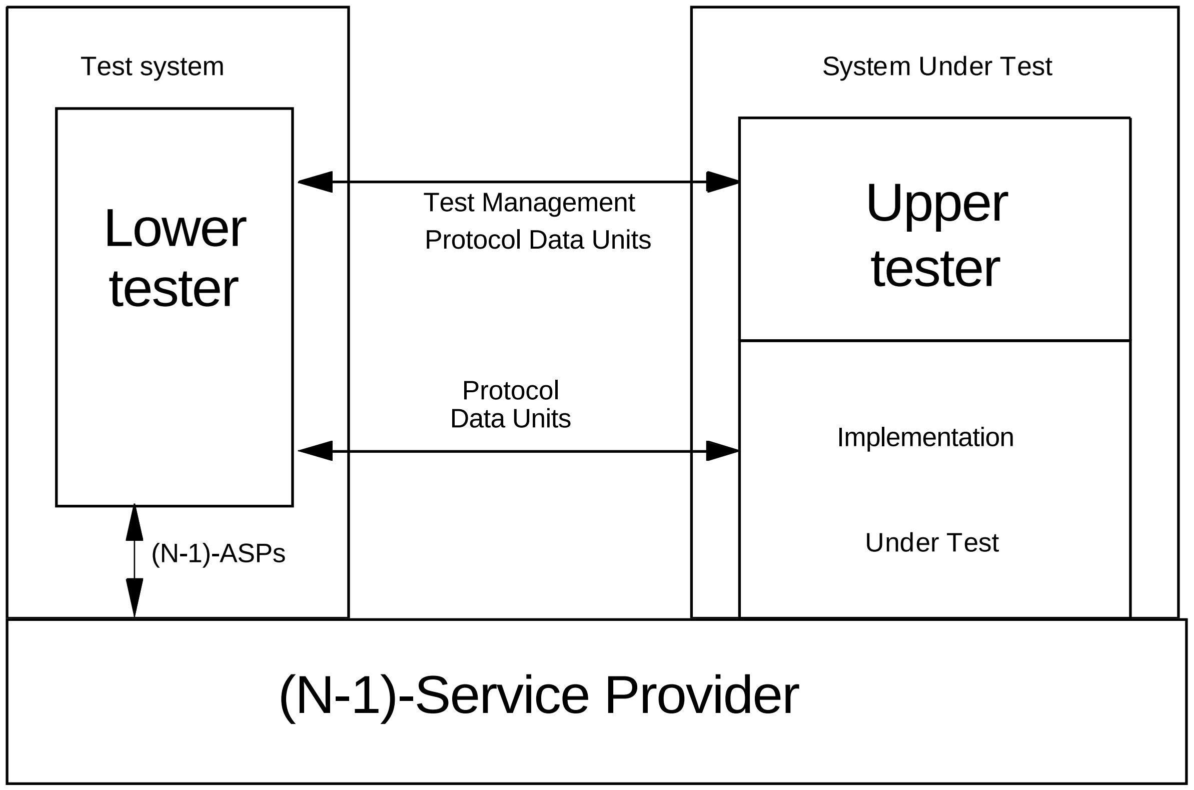

Figure 4: Co-ordinated single-layer testing method

- Remote Testing Method.

  The IUT does not have a PCO at the upper service boundary (see figure 5). Therefore, there are no test co-ordination procedures or an UT in this method. Some UT functions may be performed by the SUT.

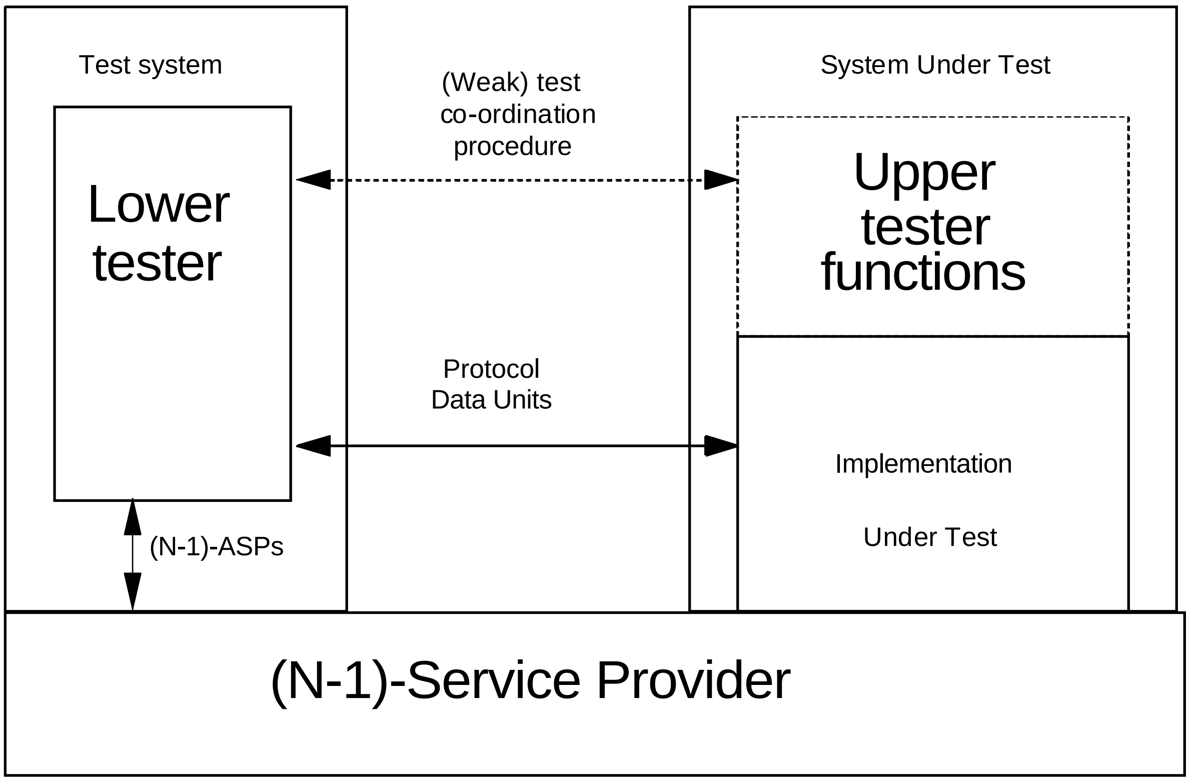

Figure 5: Remote single-layer testing method

1.2.2.4 The Ferry principle

The Ferry principle is non-standard. It has been motivated by the need to simplify and enhance the synchronisation between the UT and LT. The initial application of the Ferry principle transferred test data over a ferry channel between the UT and the IUT. This allowed the UT to be moved from the SUT into the same machine as the LT, thereby reducing the amount of test software in the SUT. Also, as the UT and the LT reside in the same machine, synchronisation between them is much simplified.

Recently, application of the Ferry principle has been extended so that it is possible to remotely access both the upper and the lower service boundaries of the IUT. This new application has been termed the "**Ferry clip**" (see "Ferry Clip Approaches in Protocol Testing" [6]). This approach is described more closely in subclause 1.2.2.5.

Testers can implement one or more of the above methods in their testbeds depending on the availability of the PCOs in an IUT. The distributed, co-ordinated, remote and ferry testing methods are also applicable to test multiple layers or an embedded layer (single layer within a multi-layer SUT) of an SUT.

Among these methods, in terms of error detection capabilities, the local testing method is the most effective one since both the upper and lower service boundaries of the IUT are assumed to be observable and controllable. This method is usually restricted to in-house testing since the service upper and lower boundaries are typically not exposed once an implementation is incorporated in a product.

The distributed and co-ordinated methods have more restrictive error detection capabilities compared to the local method since the (N-1)-ASPs are observed and controlled at the remote end of the IUT. In addition, the distributed method requires an exposed upper service boundary at the IUT. In the co-ordinated method, interchange of the TMPDUs needs additional requirements, e.g. out-of-band services, on the test suite implementation.

The ferry method is a distributed version of the local method where the synchronisation between the upper and lower testers is eliminated (they reside in the same site). However, it requires an exposed upper service boundary and out-of-band services for the ferry protocol.

The remote testing method is the most restrictive of the above methods since there is no control or observation of the (N)-ASPs at the upper service boundary of the IUT. However, it is the most simple method to implement since it does not place any restrictions on the IUT.

1.2.2.5 The "Ferry Clip" approach

The Ferry Clip (FC) approach is an extension of the Ferry approach, initiated by Zeng (see "On Ferry Clip Approaches in Protocol Testing" [6]). The purpose of the FC approach is to allow access to the upper and lower service boundaries of the IUT (if they are available), and to provide direct control and observation of each boundary. The FC architecture in general, as shown in figure 6 involves a particular intermediate service mechanism, including Active Ferry Clip (AFC) in the test system, Passive Ferry Clip (PFC) in the SUT, and the communication medium (a standard communication link).

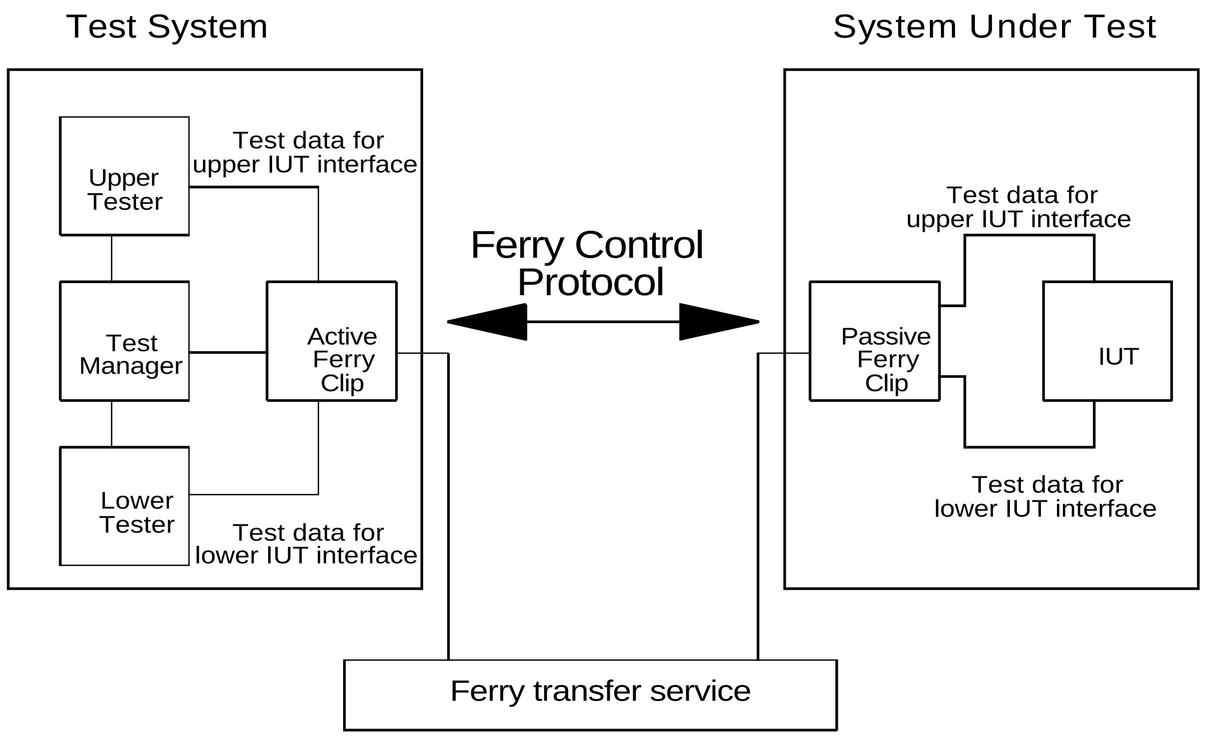

Figure 6: "Ferry Clip" test system

On an abstract level, the communication between the AFC and the PFC is through a Ferry Control Protocol (FCP). This protocol provides a transparent data transfer service which allows users in the testing context (the IUT, upper tester and lower tester), to exchange test events or test data. The advantage is that it allows the UT and LT to work as if they were local to the SUT, although they are remotely located.

The co-ordination of testing is performed by the Test Manager, which is responsible for operator event processing, test case selection and ferry channel management. The UT and LT are test case executors and test event handlers.

The creators of the approach claim that the FC approach has advantages to standard ISO test methods on the three aspects of power, flexibility and simplicity.

- The power aspect is motivated by the fact that the power of the local test method can be retained, although the test system is actually remote.

- The flexibility aspect is motivated by the fact that the FC approach can be applied to three architectural principles: local, distributed and remote testing. The approach also allows, in principle, for multi-layer testing as well as multiple connection testing. (It is unclear to what extent these situations have been actually implemented).

- The simplicity aspect is essentially motivated by two reasons. The architecture allows for a clear separation of functions, which support a modular implementation approach. There are also advantages for the distributed method in the fact that the UT is removed from the SUT. In the local test method, this is even more the case, since both the UT and LT are removed from the SUT. The FCP in itself is also very simple.

1.3 Generation of abstract test suites

According to ISO/IEC 9646 [5], test suites are constituted by test groups that are constituted by test cases. Each test case is based on a test purpose, which in turn is directly related to the specification. However, there is not always a straight one-to-one mapping from specification via test purpose to test case. The ordering of test cases into test groups and test suites is based on considerations of logical and technical efficiency. For example, the information gathered from one test case can be utilised to shortcut other test cases. Therefore, the task of test suite generation has to tackle two specific problems:

- how to create test cases from a specification and via test purposes;

- how to order test cases into test groups and test suite for maximal computational efficiency.

The latter problem is managed through knowledge about the graph that is used to represent the specification. The individual test case design is based on the concept of transition. It is assumed that the IUT performs transitions that can be modelled by a Finite State Machine (FSM) and the transitions defined for that machine.

1.3.1 Finite State Machine formalism

The behaviour of a protocol without "memory" can be specified as a FSM. A (non-extended but not deterministic) FSM can be defined as follows:

a non-deterministic FSM is a quintuple (s_0, S, A, T, τ) where:

s_0	$\in$ S is the initial state;
S	is a non-empty finite set of states;
A	is a non-empty finite set of observable actions;
T	is a non-empty finite set of transitions;
τ	is the unobservable (and non-deterministic) action.

The transition set T is a subset of the set {S x (A + τ) x S}, that is the cartesian product of states and actions. It shall be a proper subset since each transition shall lead to a distinct state. Therefore, only for a degenerated FSM with only one state in S can T be equal to {S x (A + τ) x S}, see figure 7.

In order to model machines that keep historical data through its lifetime, the model is extended with a set of variables, D, where for each variable a value set is defined. It is assumed that these variables are accessible in all states, but not accessible from outside the FSM. The local set of variables is the only source of information for the FSM that is preserved over transitions. Such a machine is called an Extended Finite State Machine (EFSM) and is defined as follows:

A non-deterministic EFSM is a sixtuple (s_0, S, D, A, T, τ) where:

s_0	$\in$ S x $\boldsymbol{\pi}$D is the initial state;
S	is a non-empty finite set of states;

D = $\{D_1, D_2, ..., D_n\}$ is a non-empty finite set of finite (value) sets D_i;

A is a non-empty finite set of observable actions;

T is a non-empty finite set of transitions;

τ is the unobservable (and non-deterministic) action.

The transition set T is a subset of the set $\{S \times \pi D \times (A \times \pi D + \tau) \times S \times \pi D\}$, that is the cartesian product of states and actions. πD is the powerset of values in D.

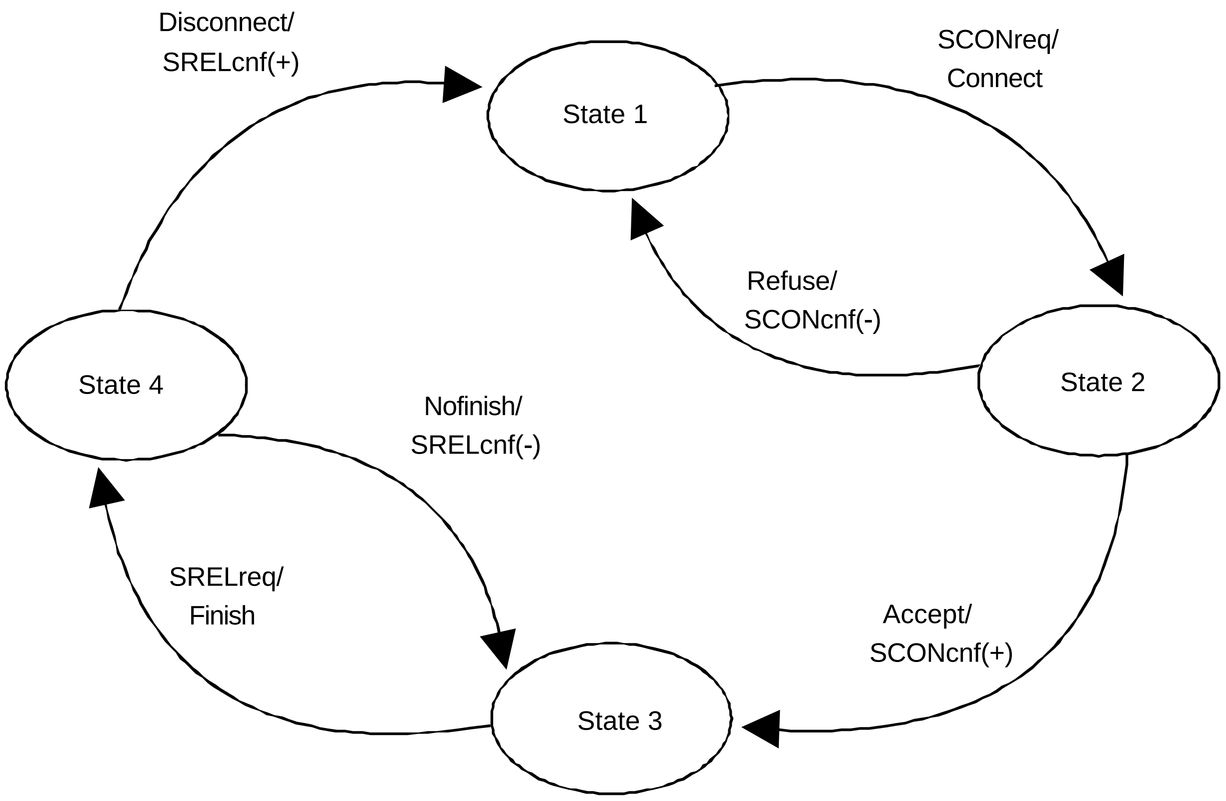

Figure 7: FSM model of a simple protocol

1.3.2 Problems with Extended Finite State Machines

The problem with an EFSM from a testing point of view, is the large number of state-value pairs caused by the powerset of values πD. (For example, a case of three variables, each ranging from 1 to 10, makes the πD to contain 1 000 distinct value triples.) The number of test cases will grow out of control, particularly if several transitions in sequence have to be considered as distinct test cases. Therefore, different techniques for complexity reduction need to be applied.

1.3.3 Problems with non-deterministic specifications

Most protocols are specified to be non-deterministic. This is due to the following fact. Assume a protocol that is performed by two co-operating processes A and B. From the view of one process, say A, process B may behave non-deterministically due to imperfections in the communication between A and B. Therefore, the specification for B, as seen by A, includes non-determinism to fully cover the total possible observable behaviour of B. The same is true for A as seen by B. So although the internal behaviour of A and B is deterministic, the fact that their communication link is not deterministic, makes it necessary to introduce mutual non-determinism. Figure 8 illustrates this situation.

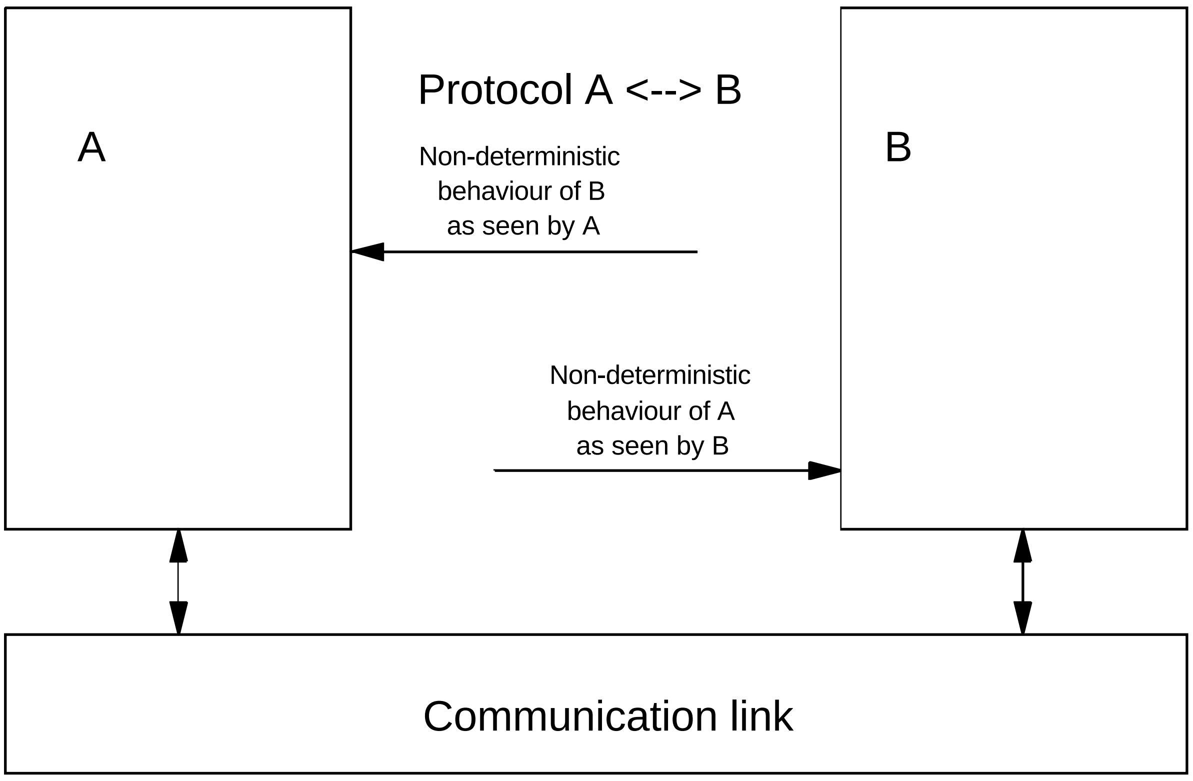
Protocol A <--> B
A
B
Non-deterministic behaviour of B as seen by A
Non-deterministic behaviour of A as seen by B
Communication link

Figure 8: A protocol situation

There are essentially three reasons for non-determinism:

1) given several senders of messages to a receiving process A, the real-time order of sending is not preserved when messages arrive at A. This is due to the fact that the communication channels are not synchronised;

2) the actual delay through a communication link. This is a non-functional property that is outside the scope of the formal specification languages. Therefore this property is open for implementation choice;

3) corruption of messages or total loss of messages in the communication link. The corruption can be considered as the loss of the correct message and silent introduction of a new message. Since there is no functional relationship between the correct message and the corrupted message, this view is justified. Therefore, we are dealing with three distinct cases:

   a) loss of a message;

   b) silent introduction of a message;

   c) a combination of "a" and "b".

One problem is to decide whether a message is delayed or lost. If the specification does not allow a certain time limit for arrival, there is no decision possible. If a time limit is allowed, and no message has arrived, it shall be considered to be lost. If it arrives later, it is considered as a silently introduced message.

The three reasons for non-determinism can be abstracted to the occurrence of silent transitions. The research in process equivalences is coping with the problem of silent transitions, regardless of the cause of their occurrence. Therefore, as long as non-determinism can be treated as a formal equivalence problem, it can be managed by existing LTS-based formalism.

1.4 Establishing a testing environment

Once a testing architecture and an abstract test suite exist, the problem of creating a testing environment is an engineering problem. The only formal problem is to preserve the abstract properties from an abstract test suite over to an executable test suite. In simple cases this can be made automatically. If it is not made automatically, the possible introduction of human errors needs to be taken into consideration.

1.5 Test execution and verdict

The test execution problem is an engineering problem. There is no reported research on the verdict problem. The verdict set {pass, fail, inconclusive} indicates a three-value logic that may cause logical problems if used in a two-value logic system. However, such problems have not been reported.

PART 2: Research approaches in conformance testing

The formal methods for generation of test data are assuming the use of formal specification languages. In particular the research is concentrated on the three standardised languages SDL, LOTOS and Estelle. Therefore a brief account of these languages.

2.1 Standardised specification languages

2.1.1 Specification and Description Language (SDL)

SDL is an acronym for Specification and Description Language. It has been developed and is maintained by CCITT WP X/3. It was first defined in the Yellow book 1976 but is now offered in the Blue book as CCITT Recommendation Z.100 [7] with the following structure:

Z.100 [7]	The SDL language itself.
Annex A	SDL Glossary.
Annex B	Abstract syntax summary.
Annex C1	Concrete graphical syntax summary.
Annex C2	SDL PR syntax summary.
Annex D	SDL user guide-lines.
Annex E	State-oriented representation and pictorial elements.
Annex F1	SDL formal definition Introduction.
Annex F2	Static semantics.
Annex F3	Dynamic semantics.

The concrete syntax of SDL is expressed by a BNF grammar as well as a flow-chart grammar. The abstract syntax is defined as Meta-IV expressions. The dynamic syntax is also expressed in Meta-IV. Meta-IV is a well-known, and mathematically solid, functional specification language. The semantics of SDL are solid since they are linked to Meta-IV.

NOTE: Meta-IV is strongly related to the mathematically based specification language Z.

On the surface level, SDL offers a graphical representation (GR) as well as a Text-string representation (PR). The PR representation is suitable for transfer between various tools. Human specifiers normally work through the GR interface, although nothing prevents experts from using a "programming" style directly into PR.

Generation of test suites from SDL has been studied by Hogrefe and others (see "Automatic generation of test cases from SDL specifications" [8], "RACE PROVE" [9] and "Exhaustive validation and test generation in Elvis SDL-89: The language at work" [10]). There are some theoretical problems involved, which calls for a simplification of the SDL specification before test cases can be automatically generated. The state of the art in this problem is represented by the RACE 1087 PROVE CATG subproject.

The problematic language concepts of SDL reported by PROVE CATG are the following:

1) Blocks

   There is no semantic meaning defined for blocks. They are provided as a method of dividing a large system into smaller component parts which can be interconnected by the use of channels.

2) Channels

   There is no semantic model for the channel queues in SDL and the unspecified delays introduced by each queue make it very difficult to predict the operation of an SDL specification. These problems have been extensively studied by Orava (see "Formal semantics of SDL specifications" [11]).

   NOTE 2: There is now a proposal by CCITT WP X/3 to allow NODELAY for channels.

3) Multiple processes

Dealing with more than one process introduces problems of non-determinism and, as a result, greatly increases the number of states to be tested. The non-determinism is caused by the independent way in which processes deal with their First-In, First-Out (FIFO) queues.

4) Problems associated with process diagrams

There are a number of elements of the process diagrams that are not easily dealt with. These include decisions, the creation of processes recursively, the save construct and timers.

The PROVE CATG project has considered Asynchronous Communication Trees (ACTs) as an intermediate representation, generated from SDL specifications. In order to generate reasonably small ACTs, the following parameters should be kept small (they will be finite!):

- the number of processes;
- the number of states in each process;
- the number of state variables associated with each state;
- the number of signal routes;
- the number of channels;
- the number of messages in the system.

Since any implementation shall choose limits of these parameters, it is, in fact, practical for both testing and implementation to offer limits either within the SDL specification, or without it, as extra requirements. A failure to do so would mean that two conforming implementations could have different sets of limits, which in practice would cause interworking problems.

2.1.2 LOTOS

LOTOS is an ISO standard 8807 [12]. LOTOS is an algebraic specification language based on a calculus of communicating systems which is used to define dynamic (observable) behaviour of processes. For the specification of data as well as interaction primitives, LOTOS is complemented with a powerful abstract data type language ACT ONE (see "Fundamentals of Algebraic Specification I" [13]). ACT ONE is used to define interaction primitives and static data. The control component is basically expressed as a process algebra in terms of a set of operations including (recursive) procedure calls. The data component is specified with the use of abstract data type operations in processes and by process functionalities, i.e. processes which accept/pass data before/after execution. There are also gate expressions for specifying parameter exchange at the interaction points. LOTOS primitive interaction type is rendezvous, i.e. synchronous. Rather than simple value passing, LOTOS processes can agree on a common value when rendezvous is achieved (see "Conformance testing Architecture and test sequences" [14] and "The formal language LOTOS" [15]).

Testing approaches based on LOTOS.

Since LOTOS is a process oriented specification language, direct use of a state transition based approach is not possible. Therefore, indirect methods shall be applied which transforms the LOTOS specification to another, logically equivalent, form. There are two approaches reported by PROVE (see RACE PROVE, "A Methodology for Computer Aided Generation or of TTCN Test Cases from LOTOS Specifications" [16]):

1) Transformation of the specification to a FSM and the use of state based test case generation technique (the "traditional" approach).

2) Transformation of the specification into a LTS and the use of parallel trace comparison (the canonical tester approach, see subclause 2.2.5).

2.1.3 Estelle

Estelle is a procedural language based on an extended finite-state machine model. It is a standard ISO 9074 [17]. Service primitives are declared as part of channel definitions using a Pascal-like syntax. PDUs need not be defined explicitly since they are sent using (N-1) service primitives. The control component is specified as an FSM: each module has a reserved major state variable and the dynamic part of the module is comprised of transitions with FROM and TO Clauses to specify state changes. The data component is specified as global types/variables, some of which can be abstract data types. Estelle primitive interaction type is queued communication which applies to communication between layer entities as well as intermodule communication.

The problematic language concepts of Estelle are reported by the PROVE CATG project:

Estelle is based on EFSM and this involves the problem of the system's internal data management. In addition the Estelle language permits some operations and includes some notions which are difficult to manage:

- non-determinism;
- dynamic management of modules;
- the DELAY Clause.

These features may not exist in the initial Estelle specification. Furthermore, when transforming a specification in standard Estelle into a simplified Estelle Normal Form, the following simplifications are made:

- modules are combined into one single module, thus eliminating internal transitions between modules;
- loops are removed;
- conditional statements are removed;
- WHEN Clauses are removed;
- calls to sub-routines are removed.

The resulting normalised specification may include unexecutable paths, which in fact makes the specification non-valid.

2.1.4 Research in common semantics

This research is important for the protocol conformance testing area for the following reason. Once there exists a solid and commonly agreed semantics for communicating processes, it is possible to regard protocol specifications as mathematical objects with well defined properties. Some of these properties are the dynamic behaviour aspects, which then can be derived from the specification. The canonical tester approach is one branch along these lines.

The languages CCS, CSP, LOTOS and ACT-ONE are taking mathematically defined properties as the base. This means that expressions in those languages can be part of a calculus.

For Estelle and SDL, which have an engineering background, the search for a common semantic is indirect. Via a mathematically sound semantic definition, the bridge to a common semantic language can be established.

The mainstream in this research is concentrated on Process Algebra, and is taking place within the RACE project SPECS. There are several related approaches documented from that research.

One approach is reported by Rob van Glabbeek and Frits Vaandrager in "Modular Specifications in Process Algebra" [18]. The paper is an effort to relate various process algebras to each other, and particularly to put, the Algebra of Communicating Processes (ACP), into a common framework. Within the PTT Research Netherlands there is a continuing sub-project within SPECS managing Common Semantics. They have developed a definition language MR and a language CRL$_k$ that is capable to express features in LOTOS and SDL. There will be developed prototype translators from both LOTOS and SDL into CRL$_k$.

In a discussion, but unpublished draft paper by J. Linn "Conformance Testing for OSI Protocols" [19], it is claimed that a Guarded Action System (GAS) offers a complete semantical model that covers Estelle, SDL and LOTOS. Linn also claims that it is always possible to map nodes and edges in GAS onto the same constructs of an LTS. This claim is unproved. It is, however, very similar in spirit to the EFSM approach described below. There are also many, here not further elaborated works, on using Guarded Horn-Clause Logic (So-called Committed-Choice Prolog) to model finite state machines extended with data and predicates for deterministic branching.

2.2 Test suite generation approaches

This text is based on "Formal Methods in Protocol Conformance Testing: From Theory to Implementation" **[20].**

2.2.1 Transition tour method

In the transition tour method, a test sequence is generated based on a FSM specification such that, when applied to the FSM implementation, every state transition, defined by the specification, is tested. The major advantage of the transition tour method is its simplicity. The important disadvantage is that the observability issue is not addressed since the tests generated by this method do not verify the new states reached. However, incorrect state transitions (although the respective input/output messages are correct) may be eventually detected during the testing.

For certain protocols, such as ISDN CCITT Recommendation Q.931 [21] ("ISDN user-network interface layer 3 specification for basic call control"), a special message, called a **status** message, is defined for testability purposes. A status message, when sent to an FSM implementation at state s_i, generates an output that uniquely identifies state s_i, for every state s_i of the FSM. For protocols that contain the status message feature, Uyar and Dahbura "Optimal test sequence generation for protocols; The Chinese postman algorithm applied to Q.931 Proc. IEEE Global Telecommunications Conference 1986" [22] showed that a minimum-cost test sequence can be generated that addresses both the controllability (minimum-cost test sequence) and the observability (by using the status messages) issues simultaneously.

Figures 9 and 10 give an example of how to generate a minimum-cost test sequence by using the Chinese Postman Algorithm. The algorithm uses linear programming (or network flow) techniques and graph theory concepts. First the FSM graph is made symmetric by duplicating the graph edges as needed such that the total cost of the duplications is minimum. Each edge (si, sj, ak/ol;cm), where cm is a non-negative integer representing the cost (i.e. the time) to traverse the edge, is assigned a variable Xij (initially set to zero) representing the number of duplicates of the edge to be used in the final symmetric graph. Then, for each vertex si of the graph, the following equation is set:

$$\sum_{\substack{\text{all incoming edges}\\ \text{of vertex si}}} (X_{ij} + 1) \; - \sum_{\substack{\text{all outgoing edges}\\ \text{of vertex si}}} (X_{ij} + 1) = 0$$

where $\sum_{\substack{\text{all incoming edges}\\ \text{of vertex si}}}$ $\sum_{\substack{\text{all outgoing edges}\\ \text{of vertex si}}}$

Represent the summation of the variables defined for each edge directed toward or away from the vertex si, respectively.

Making the FSM graph symmetric by edge duplications that are minimum-cost can be stated as a linear programming problem:

$$\text{minimize} \quad \sum_{\substack{\text{all edges} \\ \text{of graph G}}} (X_{ij} \cdot c_k)$$

with respect to $X_{ij} \geq 0$ and the equations written for each vertex as shown above. It is shown in "Matching, Euler tours and the Chinese postman" [23] that the solution for the above problem are integers and the constraint that X_{ij} be an integer need not be included.

The next step of the algorithm is to find an Euler tour for the symmetric graph. Note that the Euler tour for the symmetric graph will be a minimum-cost tour of the original graph. An Euler tour shown in figure 10 is found by using the algorithm given in "Matching, Euler tours and the Chinese Postman" [23].

In order to convert the Euler tour of figure 10 into a test sequence, the tester should add the status messages after each input/output operation in the tour (as Step 3 of the edge test procedure).

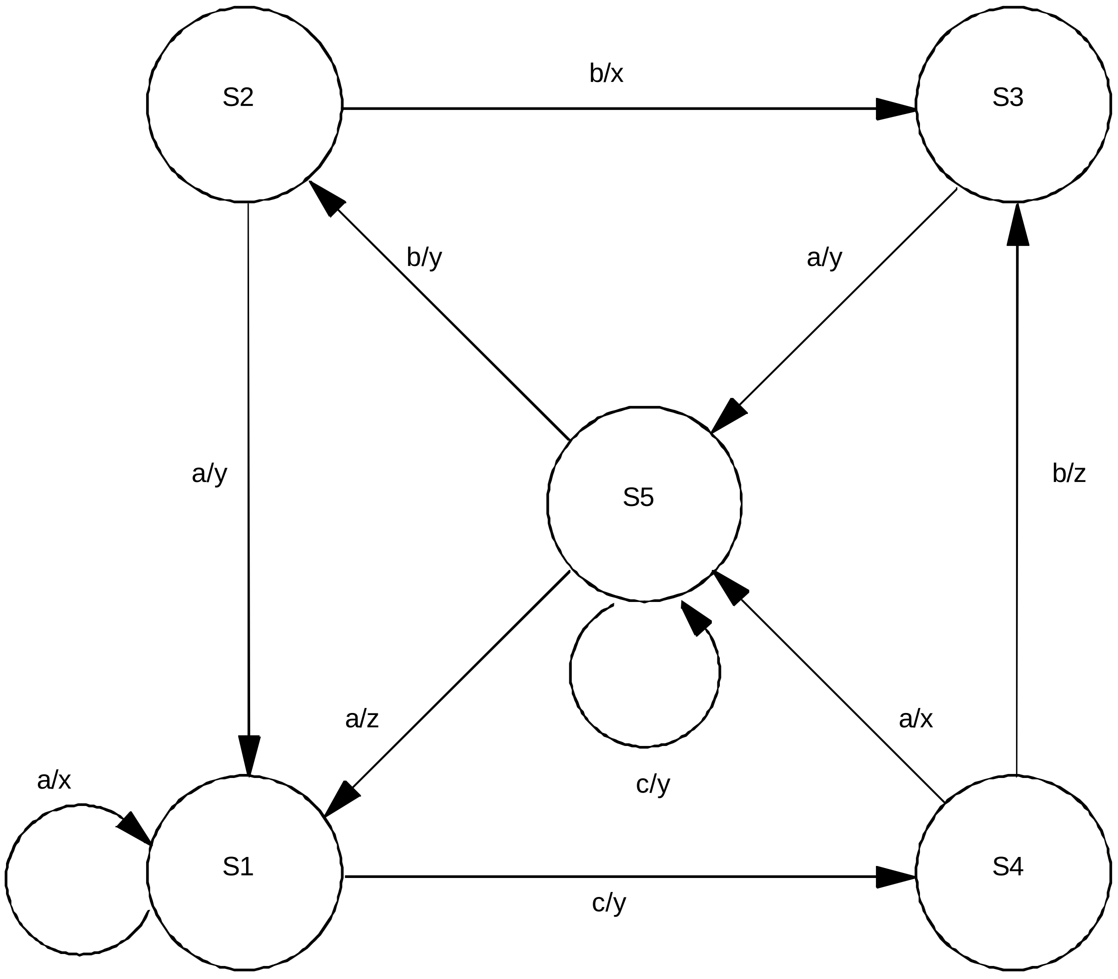

Figure 9: Transition graph

Step 1: Make the graph symmetric by using linear programming.
Solution: Add the following paths:
2(S1→S4:c/y); 1(S4→S5:a/x); 1(S5→S1:a/z); 1(S5→S2:b/y); 1(S3→S5:a/y).
The solution is in figure 10.

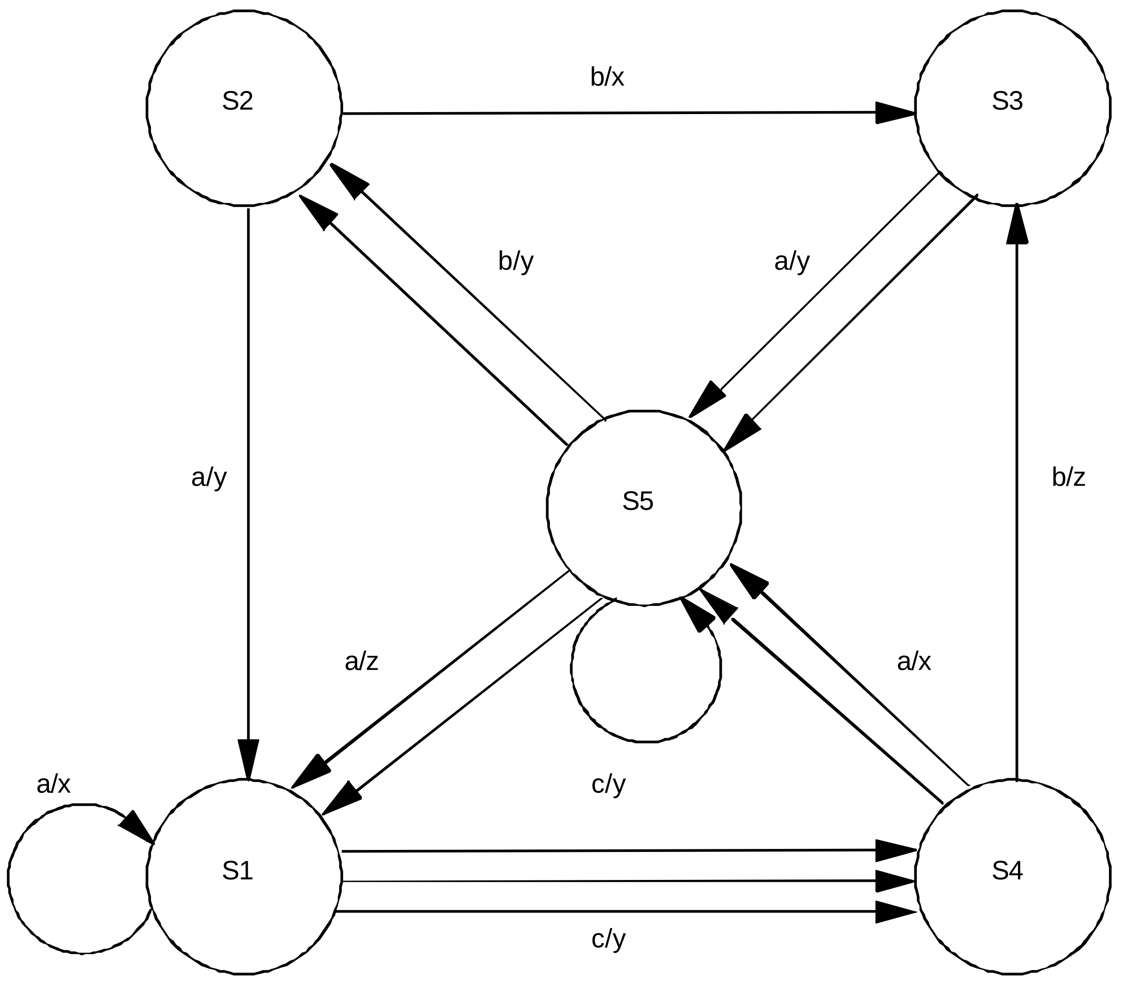

Figure 10: Minimum-cost symmetric graph

Step 2: Find an Euler tour for the minimum-cost symmetric graph:

(One solution): S1→S4:c/y, S4→S5:a/x, S5→S5:c/y, S5→S1:a/z, S1→S1:a/x, S1→S4:c/y, S4→S5:a/x, S5→S2:b/y, S2→S1:a/y, S1→S4:c/y, S4→S3:b/z, S3→S5:a/y, S5→S2:b/y, S2→S3:b/x, S3→S5:a/y, S5→S1:a/z.

2.2.2 Distinguishing sequences

A distinguishing sequence is defined as a set of inputs that generate a set of outputs different for each starting state `si` in an FSM. The distinguishing sequences developed by Gonenc "A method for the design of fault detection experiments" [24] are based on the well-studied problem of **checking experiments**, which is equivalent to the protocol conformance testing problem, developed for digital sequential circuits in finite automata theory.

For generating a test sequence by using the distinguishing sequences method, first a distinguishing sequence for the FSM to be tested is found based on the specification. For example, a distinguishing sequence for the FSM shown in figure 9 is (**a, c, a**). The outputs generated by this sequence are different for each starting state of the FSM as shown below.

The output sequences are the following for each starting state:

Starting state	Output sequence
S1	x, y, x
S2	y, y, x
S3	y, y, z
S4	x, y, z
S5	z, y, x

One can use a distinguishing sequence during Step 3 of the edge test procedure. Therefore, using this method, testing an edge (**S**i **, S**j **, a**k **/o**l) consists of placing the FSM into state **S**i, applying input **a**k, observing output **o**l and applying the distinguishing sequence to verify state **S**j. The method given by Gonenc "A method for the design of fault detection experiments" [24] additionally checks the existence of each state of an FSM before testing the state transitions.

The distinguishing sequences method brings a solution for the observability issue of testing. However, there are two major drawbacks in this method. In real-life implementations, very few FSMs have a distinguishing sequence. For example, the FSM shown in figure 11 does not possess a distinguishing sequence (see "Switching and Finite Automatic Theory" [25]). Secondly, the distinguishing sequences are typically very long, making the use of this method severely limited.

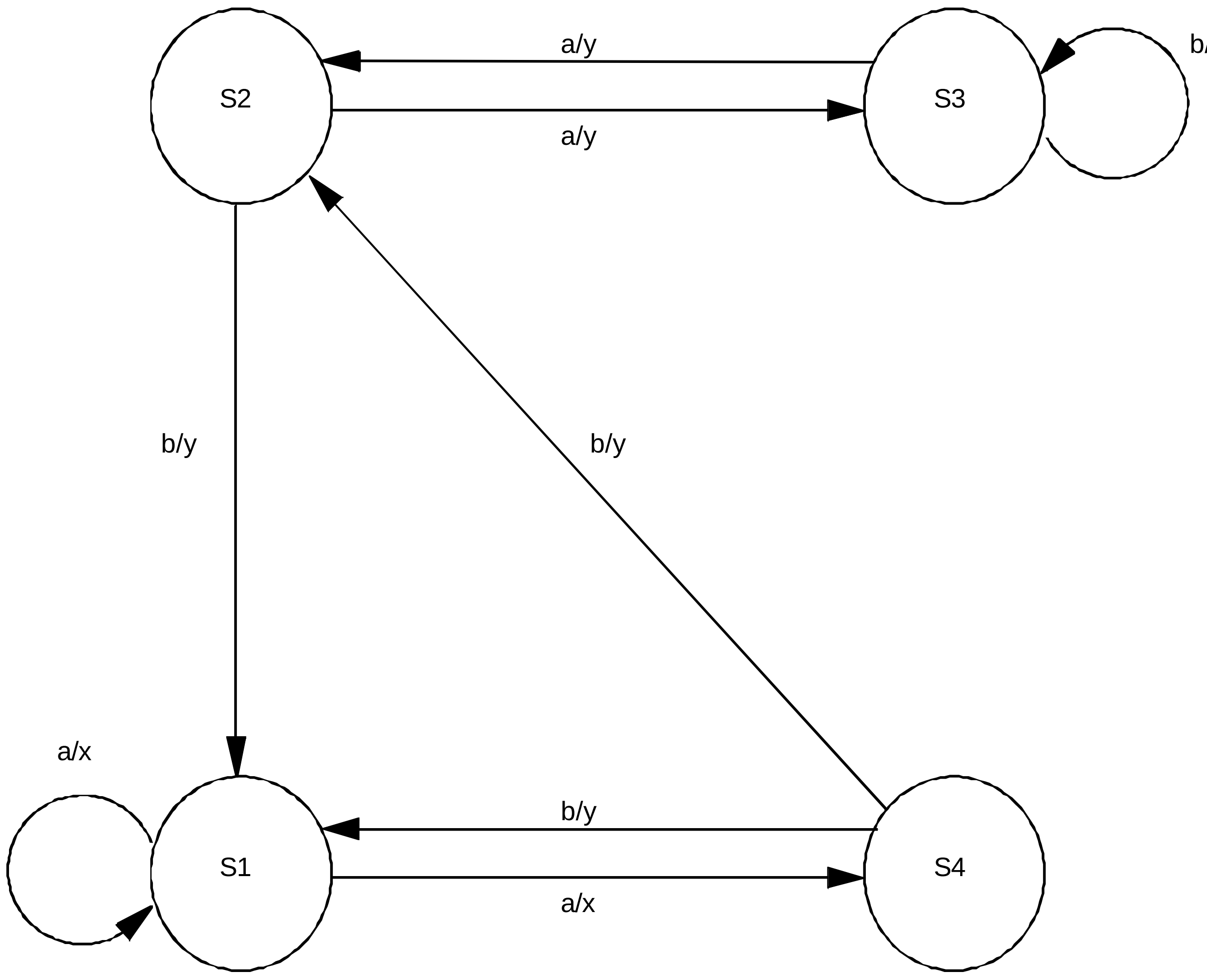

Figure 11

NOTE: This FSM does not have a distinguishing sequence.

2.2.3 Characterising sequences

For the FSMs that do not have distinguishing sequences, the characterizing sequences method defines "partial" distinguishing sequences each of which distinguishes a state s_i from a subset of the **remaining** states instead of distinguishing s_i from **every** state of the FSM. The complete set of such input sequences for an FSM is called the **characterising set** of the FSM. This method, similar to the distinguishing sequences method, requires a fully specified FSM.

The example shown in figure 11 does not possess a distinguishing sequence. However, the input sequence of (**a, b, a**) will distinguish states s_2 and s_3 from the rest of the states. In order to identify states s_1 and s_4 a second input sequence is required, which is (**b, a**) for this example. Therefore, the characterising sequences for s_1 and s_4 are the input sequences (**a, b, a**) and (**b, a**) while for s_2 and s_3 they are (**a, b, a**).

The characterising sequences method is more applicable to real-life implementations than the distinguishing sequences method. However, it does not address the controllability problem.

2.2.4 Unique input/output sequences

In this method, a minimum-cost input sequence is found for each state s_i of an FSM, such that the output generated by this sequence is unique to state s_i. This minimum-cost input sequence for state s_i is denoted UIO_i. The UIO sequence for a state can only verify that the FSM was at state s_i before the UIO_i is applied, and does not identify the state of the FSM if it is in a state other than s_i. In other words, the question of "what is the state of the FSM?" which is answered by the distinguishing and characterising sequences is changed to "is the state of the FSM s_i ?" in the UIO sequences method. If the state of the FSM is not s_i when it should be, then it is sufficient that an error is detected; the UIO sequences do not attempt to further identify the state of the FSM. In the UIO sequences method, a UIO sequence is calculated for each state of the FSM. For example, the UIO sequence for state s_1 of the FSM shown in figure 9 is UIO_1 = (**a/x , a/x**). No other state but s_1 will generate the output sequence of (**x , x**) for the input sequence of (**a , a**).

The method focuses on the observability problem similar to the distinguishing and characterising sequences methods. However, the UIO sequences method is preferable to the other methods because of the following two reasons:

- the UIO sequences methods are typically much shorter than either the distinguishing or characterising sequences since the UIO sequences are a subset of them;

- almost all FSMs have UIO sequences unless the FSM has equivalent states.

It is shown in "An optimisation technique for protocol conformance test generation based on UIO sequences and Chinese postman tours" [26] that the UIO sequences approach can be used to generate minimum-cost test sequences if combined with the Rural Chinese Postman Algorithm (RCPA). The RCP problem is NP-complete for the general case. However, a solution is guaranteed for generating a minimum-cost sequence for graphs with the following two sufficient, but not necessary, conditions:

a) the FSM specification has the reset feature;

b) each FSM state has the self-loop property.

Therefore, the UIO sequences approach brings a solution for both the controllability and the observability problems. The RCPA in addition will generate a minimum-cost test sequence under the above mentioned conditions. In this case, Step 3 of an edge test for (s_i, s_j, a_k /o_l ;**cost**) consists of applying the UIO_j to the FSM implementation. An example for minimum-cost test generation by using the UIO sequences and the RCPA is given below.

Consider the FSM graph shown in figure 9 (Note that the graph does not satisfy the sufficient conditions given above). The minimum-cost UIO sequences for each state are calculated first.

The example in figure 9 has the following minimum-cost UIO-sequence:

State	UIO sequence	Cost of UIO sequence
S1	a/x, a/x	2
S2	b/x	1
S3	a/y, a/z	2
S4	b/z	1
S5	a/z	1

Then in the next step, based on these UIO sequences, a **test graph** is constructed.

The vertices of the original graph are the vertices of the test graph. For each edge ($\mathbf{s}_i$, $\mathbf{s}_j$, $\mathbf{a}_k$/$\mathbf{o}_l$;$\mathbf{cost}_m$) of the original FSM graph, a test graph edge, called **test edge** is defined as follows: ($\mathbf{s}_i$, $\mathbf{s}_n$, $\mathbf{a}_k$ /$\mathbf{o}_l$ + $\mathbf{UIO}_j$;$\mathbf{cost}_m$ + $\mathbf{cost}$($\mathbf{UIO}_j$)) where $\mathbf{s}_n$ is the ending state of $\mathbf{UIO}_j$, and $\mathbf{cost}$($\mathbf{UIO}_j$) is the total cost of the edges in $\mathbf{UIO}_j$.

The edges of the original graph are called **ghost edges** of the test graph. Therefore, for each edge ($\mathbf{s}_i$, $\mathbf{s}_j$, $\mathbf{a}_k$ /$\mathbf{o}_l$;$\mathbf{cost}_m$) of the original FSM graph, a test graph edge, called **ghost edge** defined as ($\mathbf{s}_i$, $\mathbf{s}_j$, $\mathbf{a}_k$ /$\mathbf{o}_l$;$\mathbf{cost}_m$) is formed in the test graph. Ghost edges will be used as short-cuts between the test edges in the minimum-cost tour of the test graph when necessary.

The next step is to make the graph symmetric with respect to the test edges by duplicating the ghost edges with minimum-cost. Variables X_{ij} (initially set to 1) and Y_{ij} (initially set to 0) are assigned to the test and the ghost edges of the test graph, respectively, representing the number of duplications needed to make the graph symmetric. Then the following set of equations are set for each $\mathbf{s}_i$ in the test graph:

$$\underset{\substack{\text{all incoming edges}\\ \text{of vertex } s_i}}{\text{Sum}} (Y_{ij}) \; - \; \underset{\substack{\text{all outgoing edges}\\ \text{of vertex } s_i}}{\text{Sum}} (X_{ij}) = A_i$$

where A_i is the number of outgoing test edges minus the number of incoming test edges at s_i. The equations for the test vertices are the following:

S1: $Y_{51} + Y_{21} + Y_{11} - Y_{14} - Y_{11} = -7$
S2: $Y_{52} - Y_{21} - Y_{23} = 2$
S3: $Y_{23} + Y_{43} - Y_{35} = -1$
S4: $Y_{14} - Y_{45} - Y_{43} = 2$
S5: $Y_{45} + Y_{35} + Y_{55} - Y_{52} - Y_{51} - Y_{55} = 3$
Initially $X_{ij} = 1$, $Y_{ij} = 0$
Minimize $\Sigma\, C_{ij}\, Y_{ij}$

Similar to the case of the Chinese Postman Algorithm, the above set of equations can be solved by means of linear programming or network flow techniques to make the test graph symmetric with a minimum number of duplications.

An Euler tour found for the symmetric test graph as described using the Chinese Postman algorithm is the following:

X_{11}, X_{13}, Y_{35}, X_{52}, X_{35}, Y_{14}, X_{45}, Y_{14}, Y_{45}, X_{52}, Y_{14}, Y_{45}, Y_{52}, Y_{21}, Y_{14}, Y_{45}, Y_{52}, X_{22}, Y_{14}, Y_{45}, X_{51}, Y_{14}, X_{42}.

The resulting minimum-cost test sequence is a sequence of 38 test steps as follows:

1:S1→S1:a/x	2:S1→S1:a/x	3:S1→S1:a/x	4:S1→S4:c/y
5:S4→S3:b/z	6:S3→S5:a/y	7:S5→S2:b/y	8:S2→S3:b/x
9:S3→S5:a/y	10:S5→S1:a/z	11:S1→S4:c/y	12:S4→S5:a/x
13:S5→S1:a/z	14:S1→S4:c/y	15:S4→S5:a/x	16:S5→S5:c/y
17:S5→S1:a/z	18:S1→S4:c/y	19:S4→S5:a/x	20:S5→S2:b/y
21:S2→S3:b/x	22:S3→S5:a/y	23:S5→S1:a/z	24:S1→S4:c/y
25:S4→S5:a/x	26:S5→S2:b/y	27:S2→S1:a/y	28:S1→S1:a/x
29:S1→S1:a/x	30:S1→S4:c/y	31:S4→S5:a/x	32:S5→S1:a/z
33:S1→S1:a/x	34:S1→S1:a/x	35:S1→S4:c/y	36:S4→S3:b/z
37:S3→S5:a/y	38:S5→S1:a/z		

2.2.5 The canonical tester approach

2.2.5.1 An overview of the approach

RACE-PROVE CATG project [16] has provided much of the basis for this subclause.

The canonical tester approach was developed by Twente University (Brinksma) based on the LOTOS specification language. It is based on the conclusion that for every specification there exists a Canonical Tester (CNT). The CNT is a mirror of a specification, that calls the processes contained in the specification. The test for conformance is the way these processes are exited. For each process, there exists a predefined set of successful terminations, and if one of these is executed by the tester on leaving the process, then the test is considered as being successful (Verdict = pass).

The CNT can be broken down into test suites, each testing different processes, but together perform the same function as the original tester. A test run is complete when it has reached a deadlock state. (The proper termination may be a deadlock specified in S). An implementation I(S) can be considered to conform to the specification S, if testing the traces of I(S) by means of CNT will not lead to deadlocks that could not occur when testing S by means of CNT. The test will not detect traces of I(S) that are not specified in S. This is however, not needed for conformance.

The tests according to the canonical tester approach are so far only suitable for use with the local test method described in ISO/IEC 9646 [5], because the test is too complicated for multi-layer testing.

The approach assumes the initial transformation of the LOTOS specification into a Labelled Transition System (LTS). LTSs are generalisations of FSMs and they can be used to represent LOTOS processes.

A LTS TS is defined as a quintuple <S, L, T, $s0$> where:

S is a (countable) non-empty set of states;
L is a (countable) set of observable actions;
$T = \{-\mu\rightarrow \leq S \text{ x } S|\ \mu \in L \text{ U } \{\tau\}\}$ is a set of binary relations on S, the transitions;
$s0 \in S$ is the initial state of TS.

LTSs are a generalisation of FSMs, or automata. The special action τ represents the unobservable, internal action, which can be used to model non-determinism. In this approach a LTS is synonym to a process.

The generation of abstract test cases into standardised TTCN form is executed in six steps:

1) Transformation of Protocol Specification PS into a modified PS".

The purpose of the transformation is to eliminate properties of the PS that cause difficulties for test case generation, e.g.:

a) infinite i-chains;

b) unguarded recursion;

c) loops.

2) Transformation of PS" into a LTS (LTS).

3) Derivation of a Canonical Tester CT.

For this derivation the CO-OP method is applied.

4) Generation of Abstract Test Suites (ATS).

Input to this generation is a set of restrictions and constraints.

5) Selection of ATC from the ATS.

These test cases are expressed in LOTOS. This selection requires some intelligence and is guided by heuristics and knowledge about test purposes. The coverage will normally not be 100 %. The reason is that properties of infinity have to be removed from behaviour, data types and waiting.

The selection of ATC is a tree search problem, where a trade-off has to be made between depth and breadth. It is assumed that breadth is preferred to depth. What remains in terms of depth (e.g. not explored data values) should be explained in the assessment to what extent the successful conformance test is a contributing factor to future general interoperability.

6) Translation of ATC from LOTOS to TTCN.

This translation is more straightforward.

2.2.5.2 The CO-OP method

This subclause is based on "The CO-OP method for compositional derivation of conformance testers" [27]).

The CO-OP method derives canonical testers from LOTOS expressions that can be interpreted in terms of LTS possibly including internal transitions.

The method would work for all languages in which the expressions can be interpreted in terms of LTS. Such languages are CCS, SCCS, CSP and possibly (simplified) SDL and Estelle. However, for these languages the detailed formulas for syntactic generation of test cases have to be worked out.

The method is based on the principle that it defines two sets of elements of test cases for a LTS to be tested. These sets are (Assuming B is a LTS):

- compulsory (B);

- options (B).

Compulsory(B) is a set of events, where each element is a choice applicable at a stable state. Several choices are possible, due to the fact that there may occur internal transitions. In figure 12 the set Compulsory(B) is { {a,b}, {a,b,c} }.

Options(B) is a set of external events, where each event can be made from an unstable state without first making an internal transition. In figure 13 the set Option(B) is {a,b}.

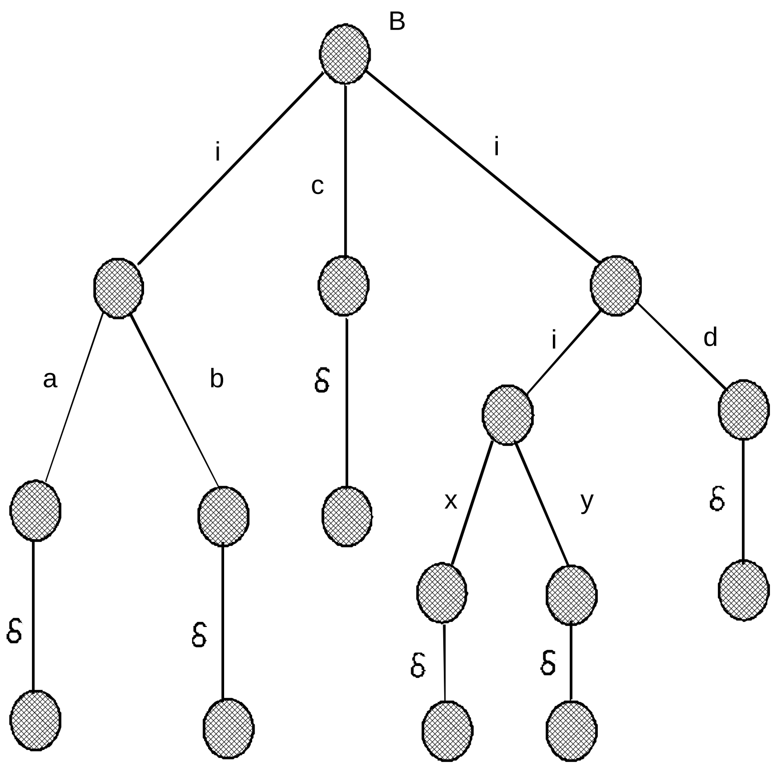

Figure 12

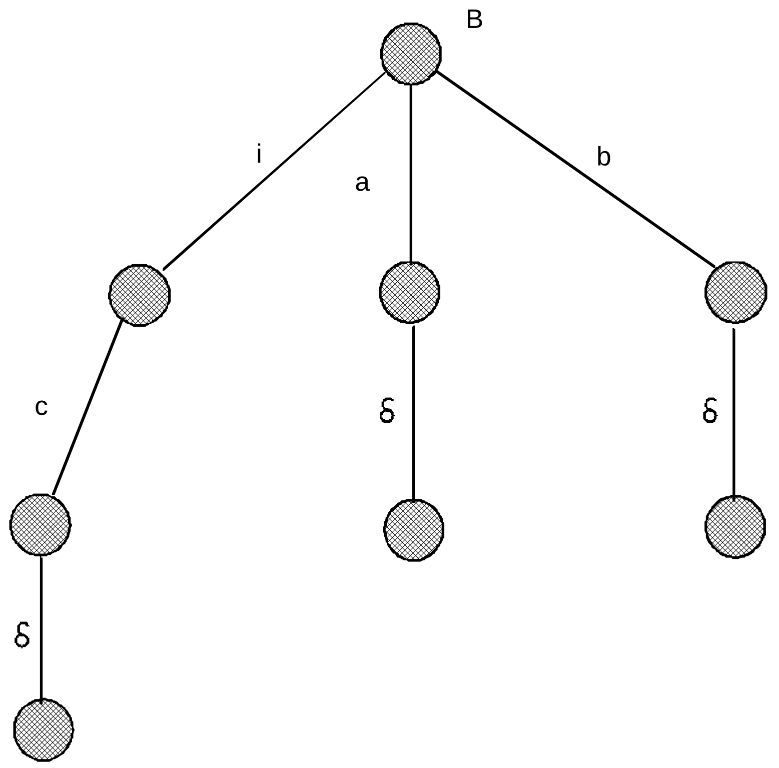

Figure 13

The set of test cases is the set of traces of external events that can be derived from a LTS. In figure 12 the set of traces is {a;δ, b;δ, c;δ, d;δ, x;δ, y;δ}.

The state of the art on constructibility is as follows:

1) From the attributes Compulsory(B), Options(B) and unstable(B) used to construct T(B1) and T(B2) it is possible to construct T(B1 * B2) where * is denoting any LOTOS operator.

2) From two testers T(B1) * T(B2) it is not possible (in general) to construct one tester T(B1 * B2) directly. The quite lengthy explanation is in "The CO-OP method for compositional derivation of conformance testers" [27].

Further research is needed on evaluation of the importance of certain test cases and the extension to value passing.

2.2.6 The extended EFSM (chart) approach

This subclause is based on a paper by P. Tripathy and Behçet Sarikaya entitled "Derivation of Test Cases from LOTOS Specifications" [28].

The main idea is to represent a concurrent system as a particular kind of transition system, called a **chart**. The concept of **chart** was introduced by Robin Milner in "A Complete Inference System for a Class of Regular Behaviours" [29]. Thus this approach is rooted in the CCS culture.

A chart is an extended EFSM such that it includes not only a set of variables, but also a set of rules for transitions. Therefore, it is an extension of the "classical" EFSM.

The formal definition of a chart is as follows:

A chart is a 8-tuple **m** = $<J, N, E, V, R, j_0, Z, h_0>$ where:

- **J** is a finite set, the control states of **m**;
- **N** is a finite set, the transitions of **m**;
- **E** is a finite subset of **J** x **I**, the extension of **m**;
- **V** is a finite set, the variables of **m**;
- **R** is a finite set, the rules of **m**;
- $\mathbf{j}_0 \in \mathbf{J}$ is the initial control state of **m**;
- Z is a finite subset of **J**, the terminal control states of **m**;
- $\mathbf{h}_0 \in \{\mathbf{v} \text{ <- } \mathbf{t} \mid \mathbf{t} \in \mathbf{T}_{\Omega D}\}$ is the initial assignment to the variables of **m**.

The chart will be manifested syntactically as a set of self-contained **rules**. The formal definition of a rule is as follows:

A rule is a 8-tuple **r** = <**a, j, j', n, p, c, f, h**> where:

- $\mathbf{a} \in \mathbf{A}$ is an **action**, the <u>when</u> clause of **r**;
- $\mathbf{j} \in \mathbf{J}$ is a **control state**, the <u>from</u> clause of **r**;
- $\mathbf{j'} \in \mathbf{J}$ is a **control state**, the <u>to</u> clause of **r**;

- **n** ∈ **N** is a **transition number**, the transition clause of **r**;
- **p** is a **predicate**, the guard clause of **r**;
- **c** is a **predicate**, the condition clause of **r**;
- **f** is a **function**, the action clause of **r**;
- **h** is a **function**, the assignment clause of **r**.

The template for a rule is as follows:

rulestart;

when: <when clause>;

from: <from clause> to: <to clause>;

transition: <transition clause>;

guard: <guard clause>;

selection-predicate: <condition clause>;

actions: <action clause>;

assignments: <assignment clause>;

end.

An SDL specification of processes can be transformed to a chart under certain well-defined restrictions. These restrictions are the following:

- the save construct must be eliminated and replaced by additional states and inputs. (This is possible since the queue is always 1-element, see below);
- macros and procedures must be eliminated by expansion. Parameter value passing is replaced by an assignment clause;
- channel name will be incorporated in the input/output construct;
- dynamic process creation may not occur;
- export/import feature may not occur.

Charts can be combined under the assumption of one input queue for each input channel and queue size one.

Silent transitions and non-deterministic choices can be represented in the chart.

It is clear that the number of rules will be very large for a SDL graph that uses macros and procedure calls in many places. One attractive feature is, however, that non-decomposable sequences of assignments are grouped into one rule. Therefore, the chart approach eliminates the overspecification in SDL that is caused by separate boxes for tasks.

LOTOS specifications can also be transformed into chart form. In order to do this, the LOTOS specification is first transformed into a semantically equivalent form. These transformations are the following:

- transformation of full synchronise composition to general composition;
- transformation of sequential composition to general composition;
- conversion of generalised choice to choice expression;
- process instantiation;
- removal of guarded internal events;
- renaming of variables.

Details are found in "Derivation of Test Cases from LOTOS Specification" [28].

In order to avoid divergence of the derivation of the chart, it is necessary to impose restrictions on the specification. The restrictions are the following:

- if μ**X.B** is a subexpression of the process, then X is guarded in B (μ is a fixpoint operator used for process declaration);
- operands of the general parallel operator are either closed or synchronous;
- operands of pure interleaving operator are closed.

Under these restrictions, there exists an algorithm that transforms any (restricted) LOTOS specification into a chart.

Thus, considering the limitations above for SDL and LOTOS, it is clear that the chart is a common semantical representation. The necessary restrictions indicate the problematic issues with the two languages.

From the chart, two graphs are generated:

1) The control graph. (Edges are transitions, Nodes are control states).

2) The data flow graph. (Edges represent the flow of information. There are four kinds of nodes: i-nodes that represent input primitives, d-nodes that represent variables, f-nodes that represent ADT operators, and o-nodes that represent output primitives.)

The general idea is expressed in figure 14.

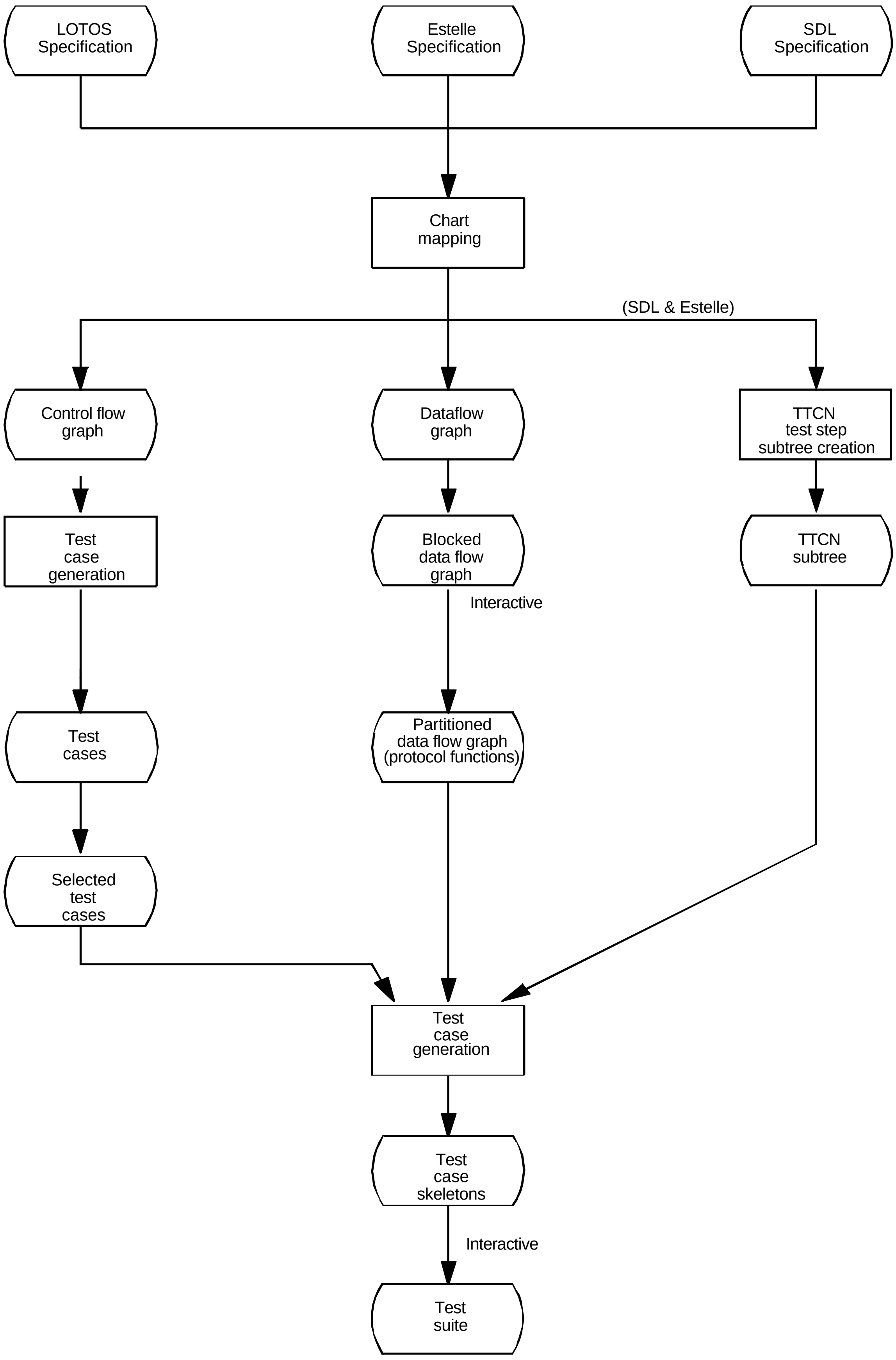

Figure 14: Chart based test generation

2.2.7 The Asynchronous Communication Tree (ACT) approach

This approach is supporting generation of test cases from an SDL specification. The approach was first reported in "Automatic generation of test cases from SDL specification" [8].

The process is performed in two steps:

1) generation of an ACT from the SDL specification;

2) derivation of test sequences (in TTCN) from an ACT.

The ACT is a commonly used representation for systems of asynchronously communicating processes. Each node in the tree represents a global state of the system, including the states of all the processes of the system and the signals en route between these processes or between the system and the environment.

In any global state (defined over the component processes), all possible events are considered. Any single event can be one of the following:

- reception of a signal by a process from a process (internal event);

- reception of a signal by the environment from a process;

- reception of a signal by a process from the environment.

Every event changes the global state of the system, e.g. by consuming a signal, changing a process state or creating an output.

In each state only a limited number of signals is expected by the system, i.e. the processes inside the system. If an unexpected signal arrives, the respective process consumes it and performs a null-transition, i.e. a transition to the same state according to the SDL semantics. Nonetheless, the global state of the system changes, because the signal is no longer pending between the sending and receiving process.

The reported approach includes an algorithm for producing an ACT from a simplified SDL specification. The introduced simplifications are:

1) no view or import;

2) no SAVE of signals;

3) only one process per block is allowed.

It is only the presented algorithm for the ACT generation from an SDL specification that demands these restrictions. It is open whether a more elaborated algorithm can overcome these restrictions, e.g. by introducing more queues and a more complex global system state. (Approaches to overcome the SAVE restriction are known but not yet formally reported.)

The second step to derive test cases from the ACT is straightforward. The internal events are eliminated from the ACT in the second step.

It should be emphasised that this approach distinguishes two signal instances which have the same signal name but different values for their parameters (data). This blows up enormously the number of possible events in a specific system state if signal parameterisation is used extensively.

Another problem is that of timers. A timer in SDL is treated almost as an ordinary signal. Since time consumption in SDL is not specified, the time associated with the setting of a timer becomes meaningless in the ACT approach. With regard to the sequences of events which are represented by the ACT, the consumption of a timer can occur almost any time after its setting.

PART 3: Dictionary

3.1 Abbreviations

For the purposes of this ETR, the following abbreviations apply:

ACT	Asynchronous Communication Tree
AFC	Active Ferry Clip
ASCII	American Standard Code for Information Interchange
ACP	Algebra of Communicating Processes
ADT	Abstract Data Type
ASN.1	Abstract Syntax Notation no. 1
ASP	Abstract Service Primitive
ATC	Abstract Test Case
ATS	Abstract Test Suite
CAD	Computer Aided Design
CATG	Computer Aided Test case Generation
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CCS	Calculus of Communication Systems (A language for communicating systems by Robin Milner)
CNT	Canonical Tester
CO-OP	Compulsory and Options
CRL	Common Representation Language (in SPECS)
CSP	Communicating Sequential Processes
CT	Conformance Testing
EFSM	Extended Finite State Machine
ETSI	European Telecommunications Standards Institute
FCP	Ferry Control Protocol
FDS	Ferry Data Service
FDT	Formal Description Technique
FIFO	First-In, First Out (Rule of a queue)
FMS	Ferry Management Service
FSM	Finite State Machine
FTS	Ferry Transfer Service

GAS	Guarded Action System
GR	SDL Graphical Representation
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardisation
ISO/IEC JTC	ISO International Electrotechnical Commission Joint Technical Committee
IUT	Implementation Under Test
LOTOS	Language Of Temporal Ordering Specification
LT	Lower Tester
LTS	Labelled Transition System
MR	Mathematical Representation (in SPECS)
NP	Non-Polynomial
OSI	Open Systems Interconnection
PCO	Point of Control and Observation
PFC	Passive Ferry Clip
PDU	Protocol Data Unit
PIXIT	Protocol Implementation eXtra Information for Testing
PR	SDL (machine) Processable Representation
PROVE	Provision for verification (A RACE project)
PS	Power Source/Power Sink
RACE	R&D in advanced communication technologies in Europe
RCP	Rural Chinese Postman
RCPA	Rural Chinese Postman Algorithm (A tour algorithm)
SDL	Specification and Description Language
SIA	Service Interface Adapter
SP	Service Primitive
SPECS	Specification and Programming Environment for Communications Software (a RACE project)
SUT	System Under Test
TC	Technical Committee (in ETSI)
TMP	Test Management Protocol

TM-PDU	Test Management Protocol Data Unit
TTCN	Tree and Tabular Combined Notation
UIO	Unique Input/Output (sequence)
UT	Upper Tester

3.2 Vocabulary of research related terms used in this paper

	Referred to in subclause:
Asynchronous Communication Tree	2.2.7
Bisimulation	1.2.1
Canonical tester	2.2.5
Characterising sequences	2.2.3
Chart (Extended EFSM)	2.2.6
CO-OP method	2.2.5.2
Distinguishing sequences	2.2.2
Equivalence relation	1.2.1
Extended finite state machine	1.3.1
Failure equivalence	1.2.1
Ferry (clip) method	1.2.2.4 - 5
Finite state machine	1.3.1
Guarded action system	2.1.4
Implementation relation	1.2.1
Labelled transition system	2.2.5
Observational equivalence	1.2.1
(Rural) Chinese Postman Algorithm	2.2.4
Testing equivalence	1.2.1
Trace equivalence	1.2.1
Unique Input/Output sequences	2.2.4

3.3 References used in this paper

For the purposes of this ETR the following references apply:

[1] CCITT Recommendation Z.110: "Criteria for the use and applicability of formal. Description Techniques".

[2] Kristoffersen, Finn
Conformance testing based on SDL specifications
SDL'89: The language at work; North-Holland 1989.

[3] Milner, Robin
Calculus of communicating systems
Lecture notes in Computer Science
(Springer, Berlin 1980).

[4] Brinksma Ed
A theory for the derivation of tests
Proceedings of the IFIP WG 6.1 VIII International
Symposium on Protocol Specification, Testing and
Verification, 1988. (Also in Ref 10).

[5] ISO/IEC 9646, Parts 1 to 5 (1991): "Information technology - Open systems Interconnections - OSI conformance testing methodology and framework - Part 1: "General concepts; - Part 2: "Abstract test suite specification (see also CCITT Recommendation X.290 (1988)); - Part 3: "The tree and tabular combined notation (TTCN); - Part 5: "Requirements on test laboratories and clients for the conformance assessment process".

[6] Zeng, H.X.; Chanson, S.T.;Smith, B.R.
On Ferry Clip Approaches in Protocol Testing
Computer Networks and ISDN Systems 17 (1989) 77-88
North-Holland, 1989.

[7] CCITT Recommendation Z.100 (1988): "Functional Specification and Description Language (SDL)".

[8] Hogrefe, Dieter
Automatic generation of test cases from SDL specifications.
SDL Newsletter No 12, June 1988.

[9] RACE PROVE
Automatic Generation of test cases from Estelle and SDL specifications via an internal representation of the protocol.
Report 89/RT61/701/PROVE/DP/21, August 1989.

[10] Bourget-Rouger, Anne; Combes, Pierre
Exhaustive validation and test generation in Elvis
SDL-89: The language at work. North-Holland 1989.

[11] Orava, Fredrik
Formal semantics of SDL specifications
Eighth IFIP WG 6.1 Int Symp on Protocol Specification, Testing and Verification
Atlantic City, 1990.

[12] ISO 8807: Information processing systems -Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviours, 1989.

[13] Ehrig, H; Mahr, B
Fundamentals of Algebraic Specification I
(Springer, Berlin, 1985).

[14] Sarikaya, Behçet
Conformance testing: Architectures and test sequences.
Computer Networks and ISDN Systems 17 (1989)
North-Holland.

[15] Eijk, P H J van; Vissers, C A, Diaz, M; Editors.
The formal language LOTOS
North-Holland 1989.
(The reference book on LOTOS).

[16] RACE PROVE
A Methodology for Computer-Aided Generation of TTCN Test Cases from LOTOS Specifications.
Report WP 602.3 August 1989.

[17] ISO 9074: Information processing systems - Open Systems Interconnection - Estelle: A formal description technique based on an extended state transition model.

[18] Glabbeek, Rob van; Vaandrager, Frits
Modular Specifications in Process Algebra
Tutorial notes from Ninth IFIP WG 6.1 International Symposium on Protocol Specification Testing and Verification, June 1989.

[19] Linn, Richard J
Conformance Testing for OSI Protocols
Computer Networks and ISDN Systems 18, 1989/90, North Holland.

[20] Bosik, Barry S; Uyar, M. Ümit
Formal Methods in Protocol Conformance Testing: From Theory to Implementation.
Tutorial notes from Ninth IFIP WG 6.1 International Symposium on Protocol Specification Testing and Verification, June 1989.

[21] CCITT Recommendation Q.931: "ISDN user-network interface layer 3 specification for basic call control".

[22] "Optiomal test sequence generation for protocols; The Chinese postman algorithm applied to Q.931 Proc. IEEE Global Telecommunications Conference, 1986.

[23] Edmonds, J; Johnson, E L
Matching, Euler tours and the Chinese postman
Mathematical Programming, vol 5, pp. 88-124, 1973.

[24] Gonenc, G.

A method for the design of fault detection experiments
IEEE Trans. on Computers, vol 19, no 7, June 1980.

[25] Kohavi, Z.
Switching and Finite Automata Theory
New York, McGrawHill, 1978.

[26] Aho, A V; Dahbura, A T; Lee, D; Uyar, M Ü
An optimization technique for protocol conformance test
generation based on UIO sequences and Chinese postman tours
Proc 8th Int Symp on Protocol Specification, Testing and Verification
North-Holland, 1988.

[27] Wezeman, Clazien D.
The CO-OP method for compositional derivation of conformance testers.
Proc. 9th Int Symp on Protocol Specification, Testing and Verification
North-Holland, 1989.

[28] Tripathy, Piuy; Sarikaya Behçet
Derivation of Test Cases from LOTOS Specifications
(IEEE Trans on Computers. To be published).

[29] Milner, Robin
A Complete Inference System for a Class of Regular Behaviors.
Journal of CSS, 28, 1984, pp 439-466.

[30] CCITT Recommendation X.400 (1988): "Message handling system and service overview".

[31] CCITT Recommendation X.500: "The Directory - Overview of Concepts, Models and Services".

3.4 Research centres, principal researchers and approaches

3.4.1 AT&T Bell Laboratories, USA

Laboratory profile

AT&T Bell Laboratories is the largest research and development laboratory in the world focusing on the state-of-the-art issues and future directions regarding computing and communication systems.

Principal researchers and their interests in protocol testing

Gerard Holzmann	Protocol verification theory
Robert Kurshan	Logical testing theory
Aleta Lapone	Protocol verification tools
D. Lee	Conformance testing theory
Joanna Patti	Protocol verification tools
Krishan Sabnani	Protocol verification theory
Kamlesh Tewani	Protocol testing standards
Ümit Uyar	Conformance testing theory
Pramode Verma	Interoperability testing tools

Key papers produced by the laboratory (in falling date order)

AT&T Technical Journal, Special Issue on Conformance Testing and Verification of Communication Protocols, Vol 69, No 1, January/February 1990 contains a set of recent papers published by the AT&T Bell Laboratories researchers. These papers are the following:

Bernstein, Lawrence	Protocols: Key to the future of computer communication.
Aho, Alfred V. Bosik, Barry S. Griesmer, Stephen J.	Protocol testing and verification within AT&T
Bertine, Herbert V. Elsner, Wolfgang B. Verma, Pramode K. Tewani, Kamlesh T.	Overview of protocol testing programs, methodologies, and standards

Uyar, M. Ümit Lapone, Aleta Sabnani, Krishan K.	Algorithmic verification of ISDN network layer protocol
Holzmann, Gerard L.	Algorithms for automated protocol verification
Har'El, Zri Kurshan, Robert	Software for analytical development of communications protocols
Sherif, Mostafa, H. Uyar, M. Ümit	Protocol modeling for conformance testing; Case study for the ISDN LAPD protocol
Bush, Matthew Rasmussen, Kris Wong, Fai	Conformance testing methodologies for OSI protocols
Dahbura, Anton T. Sabnani, Krishan K. Uyar, M. Ümit	Algorithmic generation of protocol conformance tests
Hubbard, Darrell.	Deterministic execution testing of FSM-based protocols

3.4.2 National Institute of Standards, USA

NIST-NCSL Automated Protocol Methods Group

Laboratory profile

Focus on Estelle as a specification language. Tools and applications based on Estelle. Tools and design for conformance test environments. US GOSIP Testing Program.

Principal researchers and their interests in protocol testing

J.P. Favreau	Estelle, conformance testing, FDTs in general, US GOSIP Testing Program, Test generation, ASN.1.
S. Nightingale	Conformance testing, US GOSIP Testing Program
R. Sijelmassi	Estelle (tools and use), US GOSIP Testing Program, Test generation.

Key papers produced by the laboratory (in falling date order)

Favreau, J.P. Mills, K.L. Nightingale, J.S.	"The US GOSIP Testing Program" Proc of 6th Int Conf on the Application of the Standards for OSI Washington, Oct 1991.
Sijelmassi, R.	"NIST Integrated Tool Set for Estelle" FORTE 90, Nov 1990.
Favreau, J.P. Linn, R. Nightingale, J.S.	"Formal Multi-layer Test Methodology and Its Application to OSI" FORTE 89, Dec 1989.
Favreau, J.P. Linn, R. Gaudette, P.	"Application of Formal Description Techniques to Conformance Evolution" FORTE 88, Stirling, Scotland, Sept 1988.
Sijelmassi, R. Gaudette, P	"An Object Oriented model for Estelle" FORTE 88, Stirling, Scotland, Sept 1988.
Linn, R. Favreau, J.P.	"Application of Formal Description Techniques to the Specification of Distributed Test Systems" Proc of Infocom 88, IEEE, March 1988.
Favreau, J.P. Linn, R.	"Automatic Generation of Test Scenario Skeletons from Protocol Specifications Written in Estelle" Proc 6th Int Symp on Protocol Specification, Testing and Verification North Holland, June 1986.
Nightingale, J.S. Schlicht, R.A. Linn, R.;Stewart, J. Gebase, L; Chanda, G. Favreau, J.P.	"A Test Plan for Implementation of the ISO Connectionless Network Protocol" NBS, Gaithersburg, MD. July 1985.
Nightingale, J.S.	"Application of the ISO Conformance Testing Method B to the Connectionless Network Protocol" (No Ref or Date)

3.4.3 National Physical Laboratory, United Kingdom

Laboratory profile

NPL maintains the national measurement system and does basic research to increase the effectiveness of this. The section referred here is doing research on conformance testing to protocols. It has assisted the development of ISO/IEC 9646 [5]. It has experience in developing test systems for FTAM. It is currently helping to develop the Transaction Processing standard and is writing the LOTOS specification of it. It is also investigating the automatic generation of abstract test suites from test purposes, with particular reference to Transport.

Principal researchers and their interests in protocol testing

David Rayner	Team leader. Testing methodology and conformance issues.
Michael Gill	Deputy leader. Formal methods and technical administrator.
Godfrey Cowin	Automatic test case generation and suites of test purposes.
Bronia Szczygiel	TP development and LOTOS.

Key papers produced by the laboratory in (falling date order)

Cowin, G.	Experiences in developing a Test Suite Structure and Test Purpose Document for Open Systems. DITC 179/91.
Ashford, S.J. Cowin, G.	Maintenance of Abstract Test Suites in TTCN - a comparison of three tools. DITC 174/91.
Ashford, S.J.	Abstract to Executable Test Case Translation. DITC TM 49/1.
Bunting, J. W. Rayner, D.	An optional version of TTCN for use with general purpose programming utilities. DITC TM 47/90.

3.4.4 University of British Columbia, Canada

Laboratory profile

Formal methods and tools for communication protocols, particularly protocol validation, implementation and testing. (Other areas are distributed operating systems and OSI communication networks). Emphasis is on protocol testing, especially test case generation, test coverage, trace analysis, design for testability, test case management, and test system architecture.

Principal researchers and their interests in protocol testing

Son T. Vuong	Methods and tools for test case generation, test coverage, trace analysis and design for testability.
Sam T. Chanson	Test case generation, test architecture, design for testability, trace analysis and test case management.
Gerald Neufeld.	Trace analyzer, high level protocol testing, ASN.1 tools.

Key papers produced by the laboratory in (falling date order)

Vuong, S.T. Ko K	"A novel approach for protocol test sequence generation" Proc of Globecomm'90; San Diego; December 1990.
Chanson, S. Vuong, S. Hendra, D.	"The Ferry-Clip approach to multi-party testing" Third International Workshop on Protocol Test Systems; IFIP and COS, Mc Lean, USA; October 1990.
Yang, Y. Neufeld, G.	"The design and implementation of an ASN.1 compiler" IEEE Trans on Software Engineering, October 1990.
Vuong, S. T. Chan, W.Y.L. Ito, R. M.	"The UIOv-method for protocol test sequence generation" Second International Workshop on Protocol Test Systems; Berlin, Germany; October 1989.
Chan, I; Smith, R.	"A software environment for OSI protocol test systems"

Neufeld, G.; Chanson, S.	The Ninth IFIP 6.1 Symp on Protocol Specification, Testing and
Davis, W.; Wuong, S.	Verification; Enschede, The Netherlands; June 1989
See, H.; Chan, S.	
Vuong, S.T.	"Semi-automatic implementation of protocols using an Estelle
Lau, A.C.	C compiler"; IEEE Trans on Software Engineering, pp 384-393
Chan, R.I.	March 1989.
Chanson, S.; Smith, B.	"On Ferry Clip approaches in protocol Testing"
Zeng, J.	J. of Computer Networks and ISDN Systems, Vol 17 No 2,1989.

3.4.5 University of Montreal, Canada

Laboratory profile

Teleinformatique research group; Canadian Institute for Telecommunications Research.

Research in the following areas:

- formal description techniques for OSI protocols and services;
- object-oriented specification methodology and language;
- test suite development and TTCN support;
- test result analysis and validation of test cases;
- design for simplifying testing and diagnostics;
- distributed LOTOS implementation;
- step-wise refinement and reuse of specifications.

Principal researchers and their interests in protocol testing

Gregor v. Bochmann
Behçet Sarikaya (Now left for Univ of Ankara, Turkey)
Anindya Das
Rachida Dssouli.

Key papers produced by the laboratory in (falling date order)

Forghani B.	"Semi-automatic test suite generation from Estelle"
Sarikaya, B.	Submitted for publication, 1991.
Sarikaya, B; Tripathy, P.	"LOTEST: A LOTOS test case generation tool"
Biedlingmaier, S.	Submitted for publication, 1991.
Bochmann, G. v.	"Fault models in testing"
Das, A.; Dssouli, R.	Proc. IFIP Int. Workshop on Protocol Test Systems,
Dubuc, M.	The Netherlands, Oct 1991. (Invited paper).
Ghedamsi, A.	
Fujiwara, S.	"Testing non-deterministic state machines with fault coverage"
Bochmann, G. v.	Proc. IFIP Int. Workshop on Protocol Test Systems,
	The Netherlands, Oct 1991.
Dubuc, M.; Dssouli, R.	"TESTL: An Environment for Incremental Test Suite Design Bochmann,
G.v.	Based on Finite-State Models"
	Proc. IFIP Int. Workshop on Protocol Test Systems,
	The Netherlands, Oct 1991.
Luo, G.; Das, A.	"Test selection based on SDL specifications with SAVE"
Bochmann, G. v.	Proc. 5th SDL Forum, Glasgow, UK, September 1991.
Favreau, J.P.	"Open issues in OSI protocol development and conformance Bochmann,
G. v.	testing"
Mondain-Monval, P.	Proc. Computer Networks '91, Wrocow, Poland, June 1991.

Rico, N.	"Performance description and analysis for distributed systems Bochmann,
G. v.	using a variant in LOTOS"
	Proc. IFIP Symp. on Protocol Specification, Testing and
	Verification, Stockholm, June 1991.
Fujiwara, S.	"Test selection based on finite state models"
Bochmann, G.v.	IEEE Trans. on Software Eng. Vol 17, 6 (June 1991).
Khendek, F.	pp 591-603.
Amalou, M.	
Ghedamsi	
Bochmann, G. v.	"Formal Description of Network Management Issues"
Lecomte, L.	Proc. Int. Symp. on Integrated Network Management (IFIP)
Mondain-Monval, P.	Arlington, US, April 1991, North Holland.
Tripathy, P	"Test generation from LOTOS specifications"
Sarikaya, B.	IEEE Trans on Computers, April 1991.
Eswara, S.	"Test specification in TTCN using an interactive editor"
Sarikaya, B.	J of Information and Software Technology
	Vol 32, No 9, Nov 1990.
Bochmann, G. v.	"ASN.1 and Estelle Implementation support tools"
Ouimet, D.	Proc. FORTE '90, Madrid, Spain, Nov 1990.
Neufeld, G.	
Dubuc, M.	"Translation from TTCN to LOTOS and the validation of test Bochmann,
G. v.	cases"
Bellal, O.; Saba, F.	Proc. FORTE '90, Madrid, Spain, Nov 1990.
Bochmann, G. v.	"Protocol specification for OSI"
	Computer Networks and ISDN Systems, vol 18 (April 1990).
Bochmann, G. v.	"Trace analysis for conformance and arbitration testing"
Dssouli, R; Zhao, J.R.	IEEE Tr. on Softw. Eng., Nov 1989.
Bochmann, G.v.; Barbeau, M.	"Mondel: An object-oriented specification language"
Erradi, M.; Lecomte, L.	Submitted for publication. 1990.
Mondain-Monval, P.	
Williams, N.	
Sarikaya, B; Forghani, B.	"An Estelle based test generation tool"
Eswara, S.	Computer Communications, 1989.

3.4.6 University of Ottawa, Canada

Laboratory profile

Test specification methods for distributed systems described in FSM, PetriNet, Estelle, SDL, LOTOS.

Principal researchers and their interests in protocol testing

T Yat Cheung	Formal methods for conformance testing
Luigi Logrippo	
Robert Probert	
Hasan Ural.	

Key papers produced by the laboratory in (falling date order)

Cheung, T.Y.	"Generating test sequences and their degrees of indeterminism
Wu, Y.	for distributed systems (with application to LOTOS)";
Ye, X.	Eleventh IFIP 6.1 Int Symp on Protocol Specification, Testing
	and Verification, Stockholm, June 1991.
Ural, H.	"A test sequence selection method for protocol testing"
Yang, B.	Accepted for publication in IEEE Transactions on
	Communications.
Faci, M.	"Formal specification of telephone systems in LOTOS:
Logrippo, L.	The constraint-oriented approach"

Stépien, B.	Computer Networks and ISDN Systems, April 1991.
Probert, R.L. Monkewich, O.	"TTCN - The international notation for specifying tests of communications systems"; Computer Networks and ISDN Systems, to appear.
Probert, R.L. Saleh, K.	"Synthesis of communication protocols: Survey and assessment" IEEE Transactions on Computers, Vol 40, No 4, April 1991.
Cheung, T.Y.	"An integrated GLOTOS and Petri-net based software environment for protocol specification and validation"; TR 91-14. 1991. Dept of Computer Science.
Cheung, T.Y. Ye, Y.	"An executor for graphical LOTOS"; Proc FORTE'90, Madrid, Spain, Nov 1990.
Ural, H.	"Specifications of distributed systems in Prolog" Systems and Software, Vol 11, 1990.
Probert, R.L. Saleh, K.	"A service-based method for the synthesis of communication protocols"; International Journal of Mini and Microcomputers, Vol 12, Issue 3, 1990.
Probert, R.L.;Matwin, S. Geldrez, C.;Morin, J.	"An application of explanation-based learning to protocol conformance testing"; IEEE Expert, October 1990, Vol 5, No 5.
Cheung, T.Y.; Ye, Y.C. Ye, X.; Wang, G.Q.	"UO-GLOTOS - A model/system for representing, editing and translating graphical LOTOS"; Proc FORTE'89, Vancouver, Canada, 1989.
Probert, R.L. Ural, H. Hornbeek, M.	"A comprehensive software environment for developing standardised conformance test suites" Computer Networks and ISDN Systems, Vol 18, No 1, 1989.
Cleghorn, C. Ural, H.	"ASNST: An ASN.1 support tool" Computer Communications, Vol 12, No 5, 1989.
Logrippo, L.; Obaid, A. Briand, J.P.; Fehri, M.C.	"An interpreter for LOTOS, A specification language for distributed systems"; Software Practice and Experience, Vol 18, No 4, April 1988.
Obaid, A. Logrippo, L.	"An atomic calculus of communicating systems" Seventh IFIP 6.1 Int Symp on Protocol Specification, Testing and Verification, North-Holland, 1987.

3.4.7 University of Twente, The Netherlands

Laboratory profile

The Tele-Informatics and Open Systems Group (TIOS) of the department of Computer Science of the University of Twente is a research group focusing on open and distributed systems, and computer networks. Special emphasis is put on the methodological design of open and distributed systems, in which formal methods play a key role. Much research is directed towards use of formal methods, especially the standardised Formal Description Technique (FDT) LOTOS, in protocol design, specification, verification, testing and implementation, and in FDT based tools that support these.

Within TIOS the conformance test group is involved in developing methods for conformance testing of (open) systems, based on the use of formal methods. The problem of conformance testing is studied assuming that the protocol specification is given by means of a formal description. Starting from this assumption problems are considered, like: what is conformance in a formal context, how can we test for it, how can tests be derived algorithmically from specifications, and what can be concluded from testing implementations using these tests. These questions are studied in relation with each other, and based on a common formal model.

Current research concentrates on some of these topics:

1) The development of an (FDT independent) framework for conformance testing based on formal methods, and the investigation of the relation between this framework and currently used (natural language based) methods, as in ISO/IEC 9646 [5].

2) Studying the meaning of conformance in a formal setting.

3) Investigation of the problem of test selection, both in a language independent model, and applied to the FDT LOTOS. Test selection is understood as selecting an optimal subset from the set of all (automatically) generated test cases.

4) The development of algorithms for test derivation from specifications written in the FDT LOTOS.

5) Development of tools that support the test derivation process.

The TIOS group participates in international research projects on formal methods and conformance testing, and in international standardisation organisations. Currently, TIOS is participating, among others, in ISO/IEC JTC 1/SC 21/WG1, project 54 "Formal Methods in Conformance Testing", in the ESPRIT project LOTOSPHERE, which is aimed at developing a system development strategy around the FDT LOTOS, including the conformance testing of implementations, and in the RACE project SPECS in the work package that concentrates on testing. Former participations include the standardisation of LOTOS within ISO/IEC JTC 1/SC 21, and the ESPRIT projects SEDOS and PANGLOSS.

Principal researchers and their interests in protocol testing

Rudie Alderden	Development of tools for test derivation.
Ed Brinksma	Theories of concurrency, formal models for testing.
Henk Eertink	Development of tool support for LOTOS based design and testing.
Pim Kars	Theories of concurrency, formal models for testing.
Jeroen van de Lagemaat	ISO DIS 9646 testing methodology and its relation to formal models.
Rom Langerak	Formal conformance relations, non-interleaved semantics for the specification and testing of protocols.
Giuseppe Scollo	Theories of concurrency, use of abstract data types for protocol specification.
Jan Tretmans	Formal models for conformance testing and their relation to currently used techniques, algorithmic test derivation from LOTOS specifications.
Chris Vissers	Architecture and design methodology of distributed systems, and the role of testing in it.

Key papers produced by the laboratory in (falling date order)

Verhaard, L.	"Test Selection in Conformance Testing." M.Sc. Thesis. Univ of Twente, (1991)
Brinksma, E. Tretmans, J. Verhaard, L.	"A Framework for Test Selection". to appear in Protocol Specification, Testing and Verification XI, (North-Holland, 1991)
Doornbosch, P.	"Full LOTOS Test Derivation", M.Sc. Thesis. Univ of Twente, (1991)
Tretmans, J. Kars, P. Brinksma, E.	"Protocol Conformance Testing: A Formal Perspective on ISO IS 9646", submitted to : Fourth Int Workshop on Protocol Test Systems, The Hague, The Netherlands (1991)
Brinksma, E Alderen, R. Langerak, R. van de Lagemaat, J. Tretmans, J.	"A Formal Approach to Conformance Testing" Second Int Workshop on Protocol Test systems, (North-Holland, 1990)
Alderden, R.	"COOPER, The Compositional Construction of a Canonical Tester." Proc. FORTE 90. (North-Holland, 1990)

Langerak, R.	"A Testing Theory for LOTOS using Deadlock Detection". Protocol Specification, Testing and Verification IX, (North-Holland, 1990)
Tretmans, J.	"Test Case Derivation from LOTOS Specifications." Proceedings FORTE II, (North-Holland 1990)
Tretmans, J.	"Formal Methods in Conformance Testing: ISO 9646 [5] Formally Interpreted." NNI contribution to ISO/IEC JTC 1/SC 21/WG 1 project 54 "Formal Methods in Conformance Testing", Madrid meeting (1990)
Tretmans, J. van de Lagemaat, J.	"Conformiteitsstesten." Memorandum INF-90-86, Univ of Twente, (1990). (In Dutch)
Brinksma, E.	"A theory for the derivation of tests" Protocol Specification, Testing and Verification VIII, (North-Holland, 1988)
Brinksma, E. Scollo, G. Steenbergen, C.	"LOTOS specifications, their Implementations and their Tests." Protocol Specification, Testing and Verification VI. (North-Holland, 1987)
Brinksma, E.	"On the existence of canonical testers." Memorandum INF-87-5, Univ of Twente, (1987)
Eertink, H.	"The implementation of a Test Derivation Algorithm", Memorandum INF-87-36, Univ of Twente, (1987)
Brinksma, E. Scollo, G.	"Formal Notions of implementation and conformance in LOTOS." Memorandum INF-86-13, Univ of Enschede, (1986)
Steenbergen, C.	"Conformance testing of OSI systems." Memorandum INF-86-24, Univ of Twente, (1986)

3.4.8 PTT Research, The Netherlands

Laboratory profile

PTT Research Leidschendam and Groningen.

Research in the field of formal techniques is focused on semantics and tool support for FDT. PTT Research is participating in the RACE SPECS and ESPRIT LOTOSPHERE projects. In the past PTT Research has actively participated in the development of FDTs SDL and LOTOS.

Research in the field of conformance testing is focused on automated test generation from formal protocol specifications and Application layer test suite structure. Furthermore PTT Research actively participates in ISO/IEC JTC 1/SC 21/WG 1 project 23 "Conformance Testing Methodology and Framework" ETSI TC ATM and CCITT SG X, Q.10 "A Formal Approach for Conformance Testing". PTT Research participates in EWOS EG CT and PT13.

PTT Research has extensive practical experience in test suite production, TTCN and test system realisation for ISDN protocols (CCITT Recommendation X.400 [30] "Message Handling" and CCITT Recommendation X.500 [31] "Directory Services").

Principal researchers and their interest in protocol testing

L. G. Bouma	Algebraic and process specification formalisms.
S. P. van de Burgt	Automated test generation from formal specifications, langauge theory.
H. Blik	Application layer testing.
Jan Kroon	Conformance testing methodology, automated test generation from formal specifications, test notation TTCN.
E. Kwast	Automated test generation from formal specifications, artificial intelligence.
C. A. Middelburg	Formal base and practical usage of VDM, Z, VVSL.
P. B. J. Oude Vrielink	Application layer testing.
J.J. Wester	Formal specification environments for LOTOS and ADTs.

Y. Yang	Test suites for FDT tools.
J. Zuidweg	Process algebra verification.

Key papers produced by the laboratory in (falling date order)

Wilts, H. J. Tilanus, P. A. J.	"Test design based on an SDL simulator" to appear in Proc of 5th SDL Forum, Glasgow, (September 1991)
Kwast, E.	"Towards Automatic Test Generation of Protocol Data Aspects", to appear in Protocol Specification, Testing and Verification XI, (North-Holland, 1991)
Kroon, J. Wiles, A.	"A tutorial on TTCN" To be presented at Protocol Specification, Testing and Verification XI, Stockholm, (June 1991)
Bekker, M. Balster, G.J. Blik, H. van der Haven, M.E. Oude Vrielink, P.B.J.	"Methodological aspects of OSI upper layer conformance testing". Int Workshop on Protocol Test Systems, McLean, (1990)
van de Burgt, S. P. Kroon, J. Kwast E. Wilts, H.	"The RNL Conformance Kit, tools for automated test generation", Int Workshop on Protocol Test Systems, Berlin, (1989)
van de Burgt, S. P. Kroon, J.	"Using formal language theory for generation of tests and analysis of test outcomes", Int Symp on Interoperable Information Systems, Tokyo, (November 1988)
Hengeweld, W. Kroon, J.	"Using Checking Sequences for OSI Session Layer Conformance Testing", Protocol Specification, Testing and Verification VI, (North-Holland, 1987)

History

Document history	
October 1992	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)