A Metric-based Theory of Test Selection and Coverage for Communication Protocols

by

JADRANKA ALILOVIC-CURGUS

B.Sc., Sarajevo University, 1977
M.Sc., Sarajevo University, 1987

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in the Faculty of Graduate Studies Department of Computer Science

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

JULY 1993

© JADRANKA ALILOVIC-CURGUS, 1993
In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature

Department of Computer Science
The University of British Columbia
Vancouver, Canada
Date July 29th, 1993
Abstract

A metric-based theory is developed that gives a solution to the problem of test selection and coverage evaluation for the control behaviour of network protocol implementations. The key idea is that a fast, completely automated process can uniformly cover the execution subspace of a network protocol control behaviour when characterized by appropriate metric functions, each concerned with some aspect of the protocol behaviour.

Efficient systematic approximation of complex systems behaviours is a crucial problem in software testing. This thesis gives a theoretically sound and completely automated solution to the approximation problem for the control behaviour space of network protocols generated by many concurrent and highly recursive network connections. This objective is accomplished in a series of steps. First, a metric-based theory is developed which introduces a rigorous mathematical treatment of the discipline of testing, through the definition of testing distance, test coverage metrics, and metric-based test selection method. It involves a metric characterization of infinite trace sets of protocol behaviour within complete and totally bounded metric spaces, and captures approximations of different patterns of system behaviour due to recursion and parallelism. It is shown that classes of fault coverages of well known protocol test methods form a metric hierarchy within these metric spaces. Next, a general mathematical framework is developed for reasoning about the interoperability of communicating systems. An interoperability relation is obtained which gives a theoretical upper bound for the test selection process. The two threads are drawn together in a specific test selection algorithm, showing that the generation of arbitrarily dense sets of test sequences that approximate some original test suite to some target accuracy within the theoretical upper bound, is a convergent process. Finally, the theory itself is tested on the examples of a multi-media network protocol. It is shown that very high densities of selected sets and coverage calculation can be achieved within reasonable time limits in a completely automated manner. These results indicate that there is no practical impediment to applying rigorous theoretical treatment to the discipline of testing for the case of systems that derive their complexity from highly concurrent and recursive subprocesses.
# Contents

Abstract ii

Table of Contents iii

List of Tables vii

List of Figures viii

Acknowledgement xi

1 Introduction 1
   1-1 The Problem .................................................. 2
   1-2 The Approach .................................................. 4
   1-3 Limitations and Key Assumptions ............................ 7

2 Background 11
   2-1 Testing for Reliability ....................................... 12
      2-1.1 General Software Testing .................................. 12
      2-1.2 Theoretical Investigations of Testing .................. 13
      2-1.3 Protocol Testing Theory and Practice .................. 14
      2-1.4 The Extension of the Protocol Testing Problem .......... 16
   2-2 Theoretical Background ...................................... 17
      2-2.1 Theoretical Setting ...................................... 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-2.2</td>
<td>Extensional Equivalences</td>
<td>20</td>
</tr>
<tr>
<td>2-3</td>
<td>The Metric Based Model of Testing</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>A Metric Space of Protocol Control Behaviour</td>
<td>28</td>
</tr>
<tr>
<td>3-1</td>
<td>General Issues</td>
<td>30</td>
</tr>
<tr>
<td>3-1.1</td>
<td>Testing Real Protocol Implementations</td>
<td>30</td>
</tr>
<tr>
<td>3-1.2</td>
<td>Approximating Protocol Systems by Traces</td>
<td>33</td>
</tr>
<tr>
<td>3-1.3</td>
<td>Recursion and Test Selection in Protocol Systems</td>
<td>35</td>
</tr>
<tr>
<td>3-2</td>
<td>The Metric Space of Execution Sequences of Protocols</td>
<td>38</td>
</tr>
<tr>
<td>3-2.1</td>
<td>The Definition of Testing Distance (dt)</td>
<td>40</td>
</tr>
<tr>
<td>3-2.2</td>
<td>The Interpretation of the Distance (dt)</td>
<td>43</td>
</tr>
<tr>
<td>3-2.3</td>
<td>The Finiteness of (\epsilon)-dense Covering for the Metric Space ((S, dt))</td>
<td>49</td>
</tr>
<tr>
<td>3-2.4</td>
<td>The Completeness of Space ((S, dt))</td>
<td>51</td>
</tr>
<tr>
<td>3-3</td>
<td>Control Coverage Evaluation</td>
<td>54</td>
</tr>
<tr>
<td>3-3.1</td>
<td>A Coverage Metric for Pseudo-Exhaustive Testing Strategies</td>
<td>55</td>
</tr>
<tr>
<td>3-3.2</td>
<td>A Coverage Metric for Strategies With Pre-defined Target Test Sequences</td>
<td>57</td>
</tr>
<tr>
<td>3-4</td>
<td>Metric Based Test Selection</td>
<td>59</td>
</tr>
<tr>
<td>3-4.1</td>
<td>Metric Based Test Selection Algorithm</td>
<td>59</td>
</tr>
<tr>
<td>3-4.2</td>
<td>Test Selection as a Convergent Approximating Process</td>
<td>63</td>
</tr>
<tr>
<td>3-4.3</td>
<td>Mixed Level Hierarchical Test Selection</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>A Metric Hierarchy of Fault Coverage Models</td>
<td>72</td>
</tr>
<tr>
<td>4-1</td>
<td>Some General Results</td>
<td>73</td>
</tr>
<tr>
<td>4-2</td>
<td>Analytical Comparison of Test Selection Methods</td>
<td>75</td>
</tr>
<tr>
<td>4-2.1</td>
<td>A Metric Characterization of FSM-based Methods</td>
<td>76</td>
</tr>
<tr>
<td>4-2.2</td>
<td>A Metric Characterization of Some Software Testing Covers</td>
<td>81</td>
</tr>
<tr>
<td>4-3</td>
<td>Approximating Infinite Systems</td>
<td>83</td>
</tr>
<tr>
<td>4-3.1</td>
<td>A Metric Characterization of Trace Equivalence</td>
<td>83</td>
</tr>
</tbody>
</table>
## CONTENTS

4-3.2 A Metric Hierarchy of Testing ................................................. 86

5 A Framework for Interoperability Testing ..................................... 90

5-1 Implementation relations and interoperability ............................. 92

5-1.1 Strings Equivalence ......................................................... 93

5-1.2 Implementation Relations $conf$ and $red$ ............................... 96

5-1.3 Implementation Relations and Interoperability ......................... 96

5-2 The Interoperability Relations ................................................ 102

5-2.1 The $intop$ Relation ....................................................... 102

5-2.2 Properties of the $intop$ Relation ...................................... 104

5-2.3 The $intop_{red}$ Relation ................................................ 106

5-3 The Interoperability Tester Design .......................................... 108

5-3.1 Architectural considerations ............................................. 108

5-3.2 Formal Network Protocol Specification Issues ....................... 110

5-3.3 Interoperability Tester Design .......................................... 112

5-4 Interoperability Relation and the MB method ............................. 115

6 Tests of the Theory ..................................................................... 118

6-1 Internet Stream Protocol ....................................................... 119

6-1.1 General Protocol Features ................................................. 119

6-1.2 ST-II from the Testing Point of View ................................... 120

6-2 The Implementation ............................................................. 121

6-3 Experimental Results .......................................................... 123

6-3.1 Computational Complexities and Performance Measurements ...... 123

6-3.2 Sensitivity Analysis of the Metric Based Method ..................... 124

6-4 Discussion ............................................................................. 157

7 Conclusion .................................................................................. 160

7-1 Summary of Contributions ..................................................... 160
CONTENTS

7-1.1 A Metric Characterization of Protocol Control Behaviour Space  ... 160
7-1.2 Method-independent Coverage Evaluation .......................... 161
7-1.3 The MB Test Selection Method .................................. 163
7-1.4 Interoperability Relation ........................................ 164
7-1.5 Contributions to Specific Areas of Study ...................... 165

7-2 Suggestions for Further Research ................................. 166

Bibliography ...................................................................... 168

Appendices ........................................................................ 173

A Labelled Transition Systems ........................................... 173

B LOTOS .......................................................................... 175

C Theory of Metric Spaces - Mathematical Definitions .......... 177

D A Formal Description of ISO 8072 in LOTOS ..................... 180

E A Specification of ST-II Protocol in Estelle-Y ................. 182

F Test Files and Sample Results ....................................... 194
  F-1 Seed Files ............................................................ 194
  F-2 Experiment 1.1 ....................................................... 201
  F-3 Experiment 4 ......................................................... 203
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Experiment 3 - Characteristics of randomly generated sets</td>
<td>134</td>
</tr>
<tr>
<td>6.2</td>
<td>Experiment 5 - Characteristics of test sets</td>
<td>139</td>
</tr>
<tr>
<td>6.3</td>
<td>Experiment 6 - Characteristics of randomly generated test sets</td>
<td>142</td>
</tr>
<tr>
<td>B.1</td>
<td>Axioms and Inference Rules in LOTOS</td>
<td>176</td>
</tr>
</tbody>
</table>
## List of Figures

2.1 Some illustrative examples of strings equivalence .................................................. 22
2.2 Non-determinism and testing relations ................................................................. 23
2.3 Divergence and testing relations ............................................................................. 24
2.4 Analytic Model of Testing ...................................................................................... 27

3.1 An example of faulty implementations of a simple protocol ................................. 32
3.2 Execution sequences of ISO 8072 Transport Service .............................................. 38
3.3 Increasing Similarity Information by Recognizing Recursion ................................. 44
3.4 Evaluation of the function $p_k$ ............................................................................. 46
3.5 Evaluation of the function $\delta_k$ ......................................................................... 47
3.6 Evaluation of the function $dt$ ............................................................................... 49
3.7 The Subspaces $S_n$ of Trace Truncations for the ISO 8072 Transport Service 56
3.8 Generating Cauchy sequences .............................................................................. 62

4.1 Generating a T-tour using metrics only ................................................................. 78
4.2 A Metric Hierarchy of Protocol Testing within infinite metric space $(Tr(S), dt)$ 88

5.1 Connection Establishment Phase of ST-II protocol ................................................. 95
5.2 Different implementations of a process $S$ .......................................................... 97
5.3 An interoperable implementation of $S$ ............................................................... 100
5.4 Interoperability Test Architecture ....................................................................... 109
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>A specification, its canonical tester $T(S)$, and its interoperability tester $IT(S)$</td>
<td>113</td>
</tr>
<tr>
<td>6.1</td>
<td>The Functional Structure of the Test Development System TESTGEN++</td>
<td>121</td>
</tr>
<tr>
<td>6.2</td>
<td>Timing results (in user seconds) for control sequences of length 20</td>
<td>124</td>
</tr>
<tr>
<td>6.3</td>
<td>Timing results (in user seconds) for control sequences of length 100</td>
<td>125</td>
</tr>
<tr>
<td>6.4</td>
<td>Granularity of MB test selection with event name identification</td>
<td>128</td>
</tr>
<tr>
<td>6.5</td>
<td>Granularity of MB test selection with event name, gate and data parameters identification</td>
<td>129</td>
</tr>
<tr>
<td>6.6</td>
<td>Selecting Transition Tours with moderate vertical recursion</td>
<td>132</td>
</tr>
<tr>
<td>6.7</td>
<td>Selecting Transition Tours with moderate vertical recursion</td>
<td>133</td>
</tr>
<tr>
<td>6.8</td>
<td>Pseudo-random versus random test generation</td>
<td>135</td>
</tr>
<tr>
<td>6.9</td>
<td>Selected subset size for random and backtracking algorithms</td>
<td>137</td>
</tr>
<tr>
<td>6.10</td>
<td>Sensitivity to vertical recursion</td>
<td>140</td>
</tr>
<tr>
<td>6.11</td>
<td>Mutual densities of some sets with moderate and low number of simultaneous connections</td>
<td>143</td>
</tr>
<tr>
<td>6.12</td>
<td>Mutual densities of some sets with moderate and low number of simultaneous connections</td>
<td>144</td>
</tr>
<tr>
<td>6.13</td>
<td>Mutual densities of some sets with moderate and low number of simultaneous connections</td>
<td>145</td>
</tr>
<tr>
<td>6.14</td>
<td>Sensitivity to missing events - single connection with recursion</td>
<td>147</td>
</tr>
<tr>
<td>6.15</td>
<td>Sensitivity to missing events - general metric function, multiple connections with recursion</td>
<td>149</td>
</tr>
<tr>
<td>6.16</td>
<td>Sensitivity to missing events - general metric function, moderate parallelism and recursion</td>
<td>150</td>
</tr>
<tr>
<td>6.17</td>
<td>Sensitivity to missing events - special metric, moderate parallelism and recursion</td>
<td>151</td>
</tr>
</tbody>
</table>
6.18 Sensitivity to missing events - general metric function, moderate parallelism and recursion .............................. 152
6.19 Sensitivity to missing events - special metric function, moderate parallelism and recursion ........................................ 153
6.20 Sensitivity to missing events - special metric, moderate parallelism and recursion ................................................ 154
6.21 Sensitivity to missing events - special metric, moderate parallelism and recursion ................................................ 155
6.22 Sensitivity to missing events - special metric on random sets ................................................................. 156
To my dearest Stipe

Sarajevo, - May 1993
Chapter 1

Introduction

Meeting the growing technical requirements for reliable computer systems is limited by the ability to develop, test, and maintain increasingly complex software designs. In the absence of effective tools and techniques for disciplined software engineering, the end product can be a costly source of delays and unreliability. The reliability of complex computerized systems is enhanced if they can be thoroughly tested, and released only if they satisfy stringent test coverage criteria.

In assessing the reliability, the users of computer systems are interested in the overall evaluation of systems composed of hardware and software, within their entire operational environment, and not just an isolated set of features of a particular software module. Software is now often highly concurrent, and features new complexities such as distributed computing, non-deterministic, and event-driven processes.

Extending the practice of software engineering to meet both current users’ requirements for reliable software and deal with the new set of features of modern software is a challenging problem in software testing.

The goal of this thesis is to provide a solution to the problem of modelling and au-
tomating the test selection and test coverage evaluation process with respect to both of these requirements for the control behaviour of network protocol implementations. Since overall evaluation of software implementations is our goal, the testing model presented in this thesis is geared towards a type of testing often termed coverage testing. Coverage testing strives for a thorough coverage of the entire execution space of tested systems as can be found under a wide range of operational conditions. Such are various forms of integration, pre-release, and acceptance tests, although we believe coverage testing can be a valuable component in any form of testing. The models and methods supporting coverage testing are currently exceptions, in spite of their immediate impact on the quality of released software.

1-1 The Problem

In this thesis, a protocol implementation under test is considered to be a concurrent, communicating and nondeterministic system capable of supporting any number of simultaneous network connections. Similarly, no limit is imposed by the theory on the depth of recursion that such a system may reach during its operation. In what follows, the kind of testing that we are considering is the coverage-oriented type of testing, restricted to the control protocol behaviour: the coverage target is some criterion defined on the entire control behaviour space of a protocol implementation under test. A test selection process must select some approximation of the coverage target that can be executed within a fixed amount of time. Test coverage is the degree of approximation of the coverage target. The goal of this work is to develop a mathematical theory of this process, one which precisely defines the coverage target for coverage-oriented type of network protocol implementations testing and describes a meaningful approximation of such a coverage target.

There are several reasons for this choice of problem. First, there is empirical evidence
that present partition-based and heuristic test selection techniques are an inadequate solution for testing complex software with highly unvisualizable event interleavings [36, 26]. Theoretical investigations of such techniques indicate that they are weak even with regard to developing confidence in sequential programs [25, 24, 14]. Both results suggest that there is no alternative to coverage testing for increasing confidence in the reliability of tested software. However, there is a considerable limitation in the amount and variety of test scenarios needed for coverage tests that current systematic or manual techniques can develop in a limited amount of time. Based on this research in general software testing theory and practice, we conclude that the success of such methods for testing protocol implementations with many simultaneous connections is very much in question.

Second, experimental results and software error experience with complex systems reveal a strong statistical dependency of software component failure rates on several common measures of utilization and load [38, 33]. These experimental findings underline the importance of testing for coverage of multiple simultaneous connections and high levels of recursion as major sources of increased utilization and load for protocol implementations.

Finally, of all modern complex software systems, communication protocols are perhaps the most valuable candidate for investigation of the coverage testing problem. On one hand, there is a considerable additional interest in the correctness aspect of communication protocols, in that they must be correctly implemented in order to interoperate at all. Additionally, modern communication systems depend on programmable computers and microprocessors, and exhibit complex event-driven behaviour that is composed of many concurrent and recursive processes stimulated from the layer above, from the layer below or by timers internal to protocols. Thus, if an efficient solution is found for test selection and coverage evaluation for communication protocols, it seems plausible that similar methods can be applied for a large class of complex real-time software systems.
1-2 The Approach

The problem of designing communication protocol testers in various theoretical settings has been well researched. The successful execution of specification-based testers guarantees that the protocol implementation under test is a valid implementation of the protocol specification it implements, according to the testers specific criteria of validity. All testers, however, that would attempt to satisfy the target test coverage criteria in the presence of high level of parallelism and recursion would either test exhaustively or even be infinite. The crucial issue in testing such systems is the identification of appropriate approximating subsets of test cases, including such that would be finite even if the original set of (generated) test cases is infinite. We approach this problem from the viewpoint of a mathematical characterization of all objects and activities involved in the problem, such that, when drawn together, these theoretical results will yield a practically applicable testing methodology. Whereas we will strive to illustrate the practicality of the approach by providing algorithms and tester design, these are not optimized and are meant as illustration only - this indeed is one area where much improvement can still be made.

The first step in the approach is the definition of the theoretical framework for test selection and coverage evaluation for protocol control behaviour, consisting of the following three factors: (i) the relation which describes the validity link between a protocol implementation and its specification (based on the specific validity criteria that the testing process attempts to verify), (ii) the target test coverage criteria derived from the validity relation (this becomes the theoretical upper bound for the testing process), and (iii) an efficient approximation method for such criteria (approximation mode based on some selected set of properties that hold for the implementation under test).

This work examines these three factors at the levels of theory, algorithm and implementation.

Chapter 2 provides the background material for this analysis. It begins with the sum-
mary of the major empirical and theoretical results in the general software testing theory. The emphasis is placed on recent results which suggest new directions for testing theory and practice. This is followed by an overview of important results in the testing theory for concurrent, communicating and nondeterministic processes and their application to specification-based protocol testing theory. A discussion is then presented of the ways in which these two threads can be drawn together. A testing model is derived, which is based on formal specifications of communication protocols, is mathematically sound and allows full automation of activities within it. The generality of the model is demonstrated by the fact that, in the development of the thesis, first the approximation process is characterized, and then the validity relation is defined. The validity relation is then shown to generate coverage criteria that can be successfully approximated and evaluated by the previously defined approximation process.

The analysis of the approximation process itself begins in Chapter 3. All testing activities are considered in the theoretical setting generated by the formal specifications of communication protocols. Formally defined objects are amenable to objective measurement. We exploit this relationship of formal methods to metrication to set up a metric model of the control space of process behaviour, which serves as the basis of coverage metrics and provides a guide for the selection of test cases. A minimal structure of the execution space of protocol control behaviour is assumed, that of sets of sequences of observable interactions. This enables us to use the same metric spaces for all target test relations and coverage criteria that are primarily based, or can be interpreted in terms of execution sequences. This metric aims at capturing the recursion and parallelism, which are the main sources of complexity of the protocol behaviour. A number of constraints in the metric definition are introduced, to allow a potentially infinite set of infinitely long execution sequences to be mapped into finite measures. Every Cauchy sequence of execution sequences in the metric space is also convergent with the limit in the metric completion of the same space by infinite sequences. This serves as the basis for a convergent test se-
CHAPTER 1. INTRODUCTION

lection algorithm that approximates the entire execution space of a protocol in a uniform manner with respect to the metric definition. In addition, the metric space generated by this metric is totally bounded, which guarantees the existence of finite covers for infinite sets of execution sequences. The next step is to develop a notion of test coverage measures within such metric spaces. The definition of test coverage measures for average and worst case control behaviour coverage estimates is suitable for general and special purpose testing strategies. In the conclusion of this chapter, we discuss the applicability of results to the mixed level hierarchical test selection and coverage evaluation strategies.

In the closely related Chapter 4, we use this metric system to investigate and classify the relative power of some well understood test selection methods in protocol testing. We show how the hierarchy of test covers formed by the metric based theory subsumes previously investigated theoretical coverages based on different fault models for certain coverage classes. Furthermore, the same metric based methodology naturally extends from finite to infinite domains and encompasses the full recursion and parallelism of real protocol implementations, approximating the theoretical upper bound of trace equivalence.

The question of the actual target test relation and an efficient upper bound for the testing process has been delayed to Chapter 5. In this chapter, we extend the testing theory based on formal specifications by formalizing testing for interoperability with a new relation intop. The discriminating power of some existing notions of testing relations in protocol testing theory for identifying processes which will interoperate is examined. It turns out that the intop relation is predominantly characterized by trace sets of protocol control behaviour, and that the testers built to test protocol implementations for successful satisfaction of this relation are indeed trace equivalent to the specification of protocol control behaviour. Therefore, as anticipated, this relation is a valid candidate for successful approximation in the previously defined metric spaces. It is also a more efficient theoretical upper bound to the testing process than existing conformance relations for the case of protocol implementations supporting large numbers of simultaneous
network connections.

The final step of the work is to test the theory on an actual protocol specification. In Chapter 6, process is tested on several excerpts of a multi-media communication protocol. It is shown that high target densities can be achieved in very little time for most of the traditionally considered cases. Also, the process degrades gracefully as the parallelism and recursion are allowed to increase to levels corresponding to up to 600 simultaneous network connections. The theory is shown to be capable of dealing time efficiently with a large number of situations that were not previously dealt with systematically in the testing theory or practice. This particularly includes identification of variations in vertical recursion, parallelism and missing events or event combinations, for environments supporting multiple network connections and higher levels of recursion.

The thesis results show that there is no impediment to creating a testing method that is in concert with the newest results in testing theory and empirical findings, is mathematically sound, and allows full automation of the activities in it. In Chapter 7, the contributions are summarized, and a critical look is cast back at the results achieved. We also briefly point out some improvements that the experimentation experience suggests, as well as further directions in which this theory can be developed.

1-3 Limitations and Key Assumptions

Before embarking on the development of the theory, it is important to acknowledge a number of limitations and assumptions that could potentially limit its relevance.

The key assumption of this thesis is that the description of the tested protocol system is given in terms of a formal technique that allows labelled transition systems as its semantic model. Within this model, we further assume that concurrency is represented by independent interleaving of concurrently executed events - an assumption that establishes a link
between the behaviour of a concurrent system and testing through observation. Although "concurrency via interleaving" is a commonly made assumption in protocol testing, we show how it can become a stumbling block in the development of the theoretical upper bound for interoperability testing. This assumption is relaxed to a certain extent through our definition of interoperability relation. The accompanying formal idea of testing by observation (of specified events), also precludes taking into account some common test system functions, such as run-time error indications, local timestamps, timeouts, and so on, which are commonly employed in testing activities. This limitation serves to keep the theoretical framework both simple and focused on the single issue of detecting deviations in the temporal order of specified and implemented event exchanges. This has additional impact on the practicality of the approach, in that checking for the existence of the (specified) trace will typically supply us with less knowledge about the implementation behaviour than would be the case if the former features (exception handling in particular), were to be taken into account.

The underlying idea of the research carried out in this thesis, is that a protocol system should be tested as a concurrent system, within the entire execution space which can be generated in a real operating environment. It is also assumed that all faults in this space which are observable on event interaction with the environment are equally probable. We have based our work on the work of Hamlet [25, 25], Parnas [45], and others, and theoretical results that indicate that partition testing does not inspire confidence in most realistic situations. Where this is not the case, i.e. where highly vulnerable areas of code can be determined in advance of testing, more specialized test selection methods targeting such partitions may be more efficient.\(^1\) Since this is an initial investigation of a completely new approach, we leave all additional knowledge that could potentially aid in test development (including the knowledge of operational profiles) outside of our considerations - we let the formal specification of the communicating system be the sole

\(^1\)Note that this implies that we either know the location of errors in advance, or amounts to proving the correctness of the tested system.
source of our knowledge about the implementation control space.

The possibility of divergent behaviour in the tested systems (the question of fairness in process algebras), has not been dealt with. For the development of the theory, we have chosen a theoretical setting in which all systems are assumed non-divergent - an assumption that severely limits the validity of the thesis' results in case of systems which may perform unobservable actions ad infinitum. This is an assumption that is frequently employed in the testing theory for labelled transition systems, for the ease of theoretical development. It can be removed, for instance, by introducing the notion of time into formal framework.

Another potential limitation is our choice to represent the control behaviour of the tested systems by sequences of observable interactions (formally, traces). Tree representations (e.g. trace-refusal formalism) are a more adequate representation for concurrent, nondeterministic communicating systems such as we are considering. In making this decision, we have been guided by the intuitiveness and simplicity of implementation of trace set representation. Also, it is not clear that the satisfaction of validity relations based on tree representations can be directly testable. Notice, however, that so far as the metric-based theory is concerned, a simple extension of metric function definition is needed to accomodate approximations of the tree-based representations of the protocol control space.

Finally, the mode of approximation within the metric-based method is entirely based on the similarity of behaviour evaluated through positional variations in event pattern and recursion. This choice has two implications on the possible applicability of this theory. First, the theory is only effective for systems with relatively few events and large amounts of recursion, which is the case for most protocol systems. Other methods would probably be more efficient for systems for which this is not the case. Other possible relevant features of protocol systems (such as special events in safety testing), can be treated within the theory only with substantial modifications of the metric function definition,
which is the cornerstone of the theoretical development of this thesis.

In conclusion, a limitation of a quite practical character is due. Although this thesis does not concern itself with the test generation, or tester design in particular, its industrial applicability is dependent on the existence of a theoretically valid and practically fast test generator and executable tester. One possible design that we provide (based on existing design of a canonical tester), is theoretically sound and used as an illustration of how test selection is to be viewed with respect to interoperability, both being quite practical notions. Improvements on this front in terms of scalability are much needed.
Chapter 2

Background

The purpose of this chapter is twofold - to survey the related work and establish the theoretical setting of the thesis.

First, we present a collection of results from different study areas, that lead to this thesis research. These are divided into three groups. Some reasons for the inadequacy of present testing methods for modern, complex software are presented in some detail first. They show that a fresh approach to testing may be needed in order to overcome the indicated limitations. Next, the newest advances in two rather separate testing areas: (i) theoretical investigations of testing methods for gaining overall confidence in software systems and (ii) extensional characterizations of testing and observation for concurrent, communicating and nondeterministic systems, are presented in more detail. We draw these two together for the specific case of communication protocols in the section titled "The Extension of the Protocol Testing Problem", which also gives the broadest outline of the thesis work.

Next, the theoretical setting of the thesis research is presented. Although we believe that the metric based theory of test selection and coverage is applicable to a wider range of theoretical settings, we needed to anchor our discussion for the ease of presentation. We
chose process algebras (as the specification mechanism), labelled transition systems (as their semantic model), and a rather general notion of control behaviour captured by the execution sequence space of systems, that may be derived from, but need not necessarily be tied to this theoretical setting only. The immediately following section on extensional equivalences illustrates the kinds of problems that are encountered in systems testing, and how present theory proposes to deal with them. The new testing relation derived in this thesis is entirely based on this work.

In the end of this chapter, we introduce a model for communication protocols testing, in which we draw all these results together and explicate the proposed thesis goal.

2-1 Testing for Reliability

2-1.1 General Software Testing

Current testing methods are inadequate when applied to today's complex concurrent software with highly unvisualizable event interleavings ([26]).

In practice, the majority of test design, generation, and selection is still done manually, often left to the least experienced members of the team or those who joined the project last. The manual (heuristic) approach to test selection is advocated even when the rest of the testing problem is treated mathematically [4]. Besides being time consuming and costly, manual (even if expert) test selection in the presence of complex software becomes impossible to solve systematically and with objectively measurable outcome. Not surprisingly, some recent experimental results in fault tolerance techniques [36] detect errors in implementations that evaded detection by every programmer in testing.

Recently, there has been a substantial investment by industry in test tools, standard test methods, languages and test suites, and even specialized testing corporations whose
mandate is to provide (conformance) testing services. Despite significant improvements to testing practice, the following investigations seem to point out that the expected gains may soon reach the limit due to the increasing complexity of current software systems.

A number of more recent testing methods [56, 55] extend the structural testing criteria that is traditionally applied to sequential software to concurrent programs. Taylor, Levine, and Kelly [54], point to at least two drawbacks of this type of approach: (i) structural techniques are among the weakest available with regard to developing confidence even in sequential programs (see also [24]); (ii) the scalability of methods for large systems is very much in question. Reports on hands-on experience with test tools that are based on these and other partition-based sequential software testing paradigms warn of the same problem [26, 36, 37, 43, 57].

Experimental results and software error experience with complex systems also reveal a strong statistical dependency of software component failure rates on several common measures of utilization and load [5, 33, 38]. These experimental findings are not systematically addressed by current testing methods.

Because of these problems, it does not seem justifiable to bridge admittedly weak traditional testing methods to complex concurrent software engineering.

2-1.2 Theoretical Investigations of Testing

Recent theoretical advances in the investigation of testing, and experimental results on the dependability of the released software, offer fresh insight into the perspectives for more scientific, and hopefully more successful, treatment of this problem.

In their independent investigations of testing for true reliability or confidence in released software, both Parnas [45] and Hamlet [23] reach similar conclusions with respect to the sampling basis and sampling mode for software system tests. The negative answers to
the question of the existence of a valid operational profile for today's complex software and to the validity of successful partition delineation for test methods that essentially target faults (which include all structural and functional test methods), lead to the conclusion that the tests should be chosen from the entire state space of the operating process and from a distribution in which all state-input combinations are considered equally likely. This choice of the sampling basis and the sampling mode puts the recommended test strategies into the coverage oriented class of testing. The theory does not suggest a practical testing method, recognizing among other difficulties that there are far too many tests required if the sampling basis is a very large state space. The theory has been used for theoretical comparison in [25]. The important result of this theoretical investigation (see also [14]) is that, assuming that tests are selected so that the state space is uniformly sampled, it is shown that all test methods that can broadly be categorized as partition testing are far inferior to the uniform sampling of the entire state space. In addition, the disadvantage is magnified by more partitions or by lower fault rates. Lower fault rates are expected for coverage tests, which are applied in acceptance, release and integration testing of software systems. In the conclusion of another related research [23], Hamlet states that for the true applicability of this theory a fully automated support for test development is needed to handle the much larger number of sampling points than usually used in software testing.

This thesis research is based on these theoretical advances, adopted for the specific case of specification based testing of communication protocols.

2-1.3 Protocol Testing Theory and Practice

In the conclusion of this analysis, we turn our attention to the above issues in the communication protocols domain.

For communication protocols, the major factors that influence resource utilization and
load are the prolonged exchange of data during a communication link lifetime and the number of simultaneous connections an implementation supports at a particular moment of operation. Exactly the same factors are the major source of the state space explosion for a protocol implementation operating under realistic circumstances and therefore affect the scalability of test methods.

Current protocol test methods are not well equipped to deal with this dual problem of load and space complexity. General software and Finite State Machine (FSM) based protocol test methods are not designed to systematically generate and select tests for large multiconnection environments. These selection methods include transition and transfer fault checking experiments for classes of FSMs (a summary in [52]) and combined control and data flow analysis (examples in [49, 55]). Other test selection methods reported in literature, which are based on expert opinion, valuations and probabilities ([4, 62, 51]), a priori assume that the sampling of the execution space should not be uniform but based on some specific notion of a value of an individual test case. All of the above protocol test methods involve exact evaluations of test selection criteria, and as such must impose limiting assumptions about the recursion and parallelism in protocol systems in order to be computable for very large or infinite systems([21]). Since these methods are not designed to cover the entire execution space of the operating protocol process, the relationship of their achieved coverage with respect to the entire execution space of a multiconnection protocol implementation has not been theoretically characterized.

Such protocol test generation methods exist that can be viewed as candidates for coverage oriented testing strategies. Specifically, mutation-based testing [47, 22] and all-paths testing [48, 56], are methods whose theoretical upper bound may be defined to be entire execution space of the operating process. Appropriate test selection and coverage evaluation theories and algorithms which would be provably convergent if such a space is very large or infinite, have not been developed.

The theoretical milieu of process algebraic research includes theoretical investigations
of the validity links between an implementation and its specification [39]. The process algebraic theory for concurrent systems testing, as well as the related theory for communication protocols testing imply the exhaustive test generation as the theoretical upper bound of the testing process. Test selection, however, remains an open problem in these theories of testing, if multiconnection protocol environments are the test generation domain.

2-1.4 The Extension of the Protocol Testing Problem

It is precisely this correspondence of the newest theoretical results in the theory of general software testing and process algebraic theory for communication protocols testing that lead to the research described in this thesis. We have defined the coverage testing problem for communication protocols in the Section 1-1, and presented its characteristics in Section 2-1.3. The theoretical background for exhaustive testing based on observation, which fits well into the concept of coverage testing, has been extensively researched. If a method can be found which is able to address the problem of the systematic, automated test selection and coverage evaluation based on the process algebraic theory of testing, it would yield a coverage oriented test development methodology for the case of execution space of large multiconnection protocol implementations. This thesis concerns itself with the development of a theoretical and algorithmic basis for such a method.

For the development of the subsequent theory, we will place the research work in one of the possible theoretical settings, which we hope is the most general one: that of Labelled Transition Systems (LTS) and extensional equivalences for process algebras adapted to the labelled transition systems framework. While the theoretical setting of LTS-s is used throughout the thesis, extensional equivalences serve as a basis for initial approximation of the control space and for more specific testing relations developed later.
2-2. THEORETICAL BACKGROUND

2-2 Theoretical Background

In this chapter, we present the general theoretical setting of the thesis and the theory of extensional equivalences in process algebras in sufficient detail to allow for the further development of the metric-based test selection and coverage theory for communication protocols.

2-2.1 Theoretical Setting

A. The Formal Description Technique

Formal description techniques are a fundamental step towards disciplined software engineering. The framework chosen in this thesis is Milner's Calculus of Communicating Systems (CCS), and its standardized derivative LOTOS\(^1\). CCS [42] is an algebraic language for specifying concurrent communicating processes and LOTOS [2, 31] is a Formal Description Technique (FDT) intended for formal specification of communication protocols and distributed systems. In the theory of concurrency that applies for process algebras and derived formalisms, two agents are composed via their mutual communication which determines the composite; their independent actions are treated as occurring in arbitrary order but not simultaneously [42]. The reasoning behind this is that an (external) observer can only record the effects of actions as though they took place sequentially\(^2\).

Communication protocols and distributed systems for which LOTOS is specifically designed are a typical representative of this class of systems. The above notions of concurrency and observation are particularly valuable in the formal analysis and the closely associated testing theory and practice for concurrent communicating systems. That is

\(^1\)Since only protocol control behaviour is examined, both are considered without value passing (i.e. data exchange).

\(^2\)Note that this observation applies strictly to the formal theoretical setting used in this thesis. Other formalisms may use timestamps, etc. to record concurrent effects in the test system.
because the concept of observation represents a natural interpretation of the process that takes place during the testing activities. In this thesis we use the entire set of FDT operators that generate concurrent behaviour, and examine all of its aspects through observation\(^3\).

**B. The Semantic Model**

We study the validity relation and the approximating process for communication protocol testing in the setting of *Labelled Transition Systems* (LTS). Labelled transition systems represent a natural semantic model for the formal description technique LOTOS\(^4\). LOTOS is the specification formalism that has a particular class of language expressions for the description of system behaviour, which we refer to as *processes*. The inference rules for LOTOS operators generate a labelled transition system for each operator of the language. Therefore, from now on, we will make no distinction between processes in LOTOS and a labelled transition system.

Transition systems are an abstract model based on two primitive notions, *state* and *transition*. Given any other model for which it is possible to define a notion of global state and a notion of indivisible action causing a state transition, for each object of the model a transition system can be defined. This correspondence determines an interleaving semantics for the model. We will consider a particular class of *nondeterministic* transition systems with *interleaving semantics* of concurrency, which exchange interactions with the environment and are capable of performing an internal action. This model is formally defined in the following manner.

**Definition 2.1 ([12])**

A Labelled Transition System is a quadrupole \((Q, E, -a \rightarrow, q_0)\) where \(Q\) is a countable

\(^3\)Other related research does not always consider all parallel composition operators, so care should be taken in comparing the appropriate results.

\(^4\)The results derived in this thesis for labelled transition systems can be interpreted for a wider range of formalisms for which labelled transition systems are a semantic model [2].
set of states, $E$ is a countable set of elementary actions, $-a \rightarrow$ with $a \in E \cup \{i\}$, is a set of binary relations on $Q$ and $q_0 \in Q$ is the initial state. □

In this definition, each of the relations $-a \rightarrow$, $a \in E$, describes the effect of the execution of the elementary action $a$ and, if $q, q' \in Q$, then $q - a \rightarrow q'$ indicates that, by performing action $a$, a system, when in state $q$, can reach state $q'$. In accordance to LOTOS, the special symbol $i$ is used to denote internal actions. $q - i \rightarrow q'$ indicates that a system in state $q$ can perform a silent move to state $q'$.

A transition system can obviously be unrolled into a tree. The initial state is the root and the transition relation is represented by the arcs labelled with elements from $E \cup \{i\}$; the various nodes will stand for other states. One graph model which is frequently used represents transition systems in terms of synchronization trees (see [42]).

C. Execution Sequence Space

The operational semantics of labelled transition systems and some formal specification techniques for protocols enable us to derive sequences of actions, in which a communication system can engage, from the structure of the process itself (for details see [31]).

Let $E$ be a finite set of symbols, which represents (atomic) statements of a concurrent system. For each alphabet $E$, $E^*$ denotes the set of all strings which are formed from symbols in $E$. These strings may then be interpreted as a sequence of (atomic) interactions, *execution sequence*, that may take place between the system and its environment. In this thesis we propose a theoretical framework for the test selection and coverage of protocols, on the basis of very limited assumptions on the Formal Description Techniques (FDT) used to describe a communication protocol, and very limited assumptions on the kind of observation that an observer in the environment of the tested system may record. We therefore use, as the most basic building block for the control behaviour of protocols, the notion of *execution sequence space*. 
D. The Formal Thesis Setting

For the ease of the theoretical development of the thesis, we assume that protocol systems are given a precise description by means of a formal specification in LOTOS. The formal description of a system is given in terms of behaviour expressions, which we will refer to as processes and make no distinction between them and a corresponding labelled transition system. The testing activities will be investigated in the context of observation, sometimes referred to as black-box testing in practice. In Chapters 3-4, we will carry out the investigation of the approximation process in the execution spaces of all execution sequences derived by interleaving semantics of concurrency for formal description techniques that allow specifications of concurrent communicating protocol systems and have a finite labelset of atomic actions that we call events. The full power of labelled transition systems will be used from Chapter 5 on, to investigate the validity relations and define the test selection algorithm. In most examples, we will represent labelled transition systems by synchronization trees [42].

The formal definitions of concepts and notation used in this thesis can be found in Appendix A: Labelled Transition Systems, and Appendix B: LOTOS.

2-2.2 Extensional Equivalences

The theory of extensional equivalences for processes aims at establishing whether two systems are equivalent or whether one system is a satisfactory "approximation" of the other. If some formalism is used to describe the system requirements (its specification), then it is possible to use theories based on equivalences or approximations which are defined for the formalism, to show that a particular implementation is a satisfactory approximation of the given specification.
A. Testing and Observation

The basic idea behind the extensional equivalences is that two systems are equivalent whenever no external observation can distinguish them on the basis of some crucial aspects of system’s behaviour that the particular equivalence relation takes into account. The effect of a system on the environment and its reactions to stimuli from the environment constitute its extensional or observable behaviour.

This behaviour of processes can be investigated by a series of tests. In the testing theory based on observation we think of an observable execution environment consisting of a set of processes and a set of relevant tests. Then two processes are equivalent (with respect to this set of tests) if they pass exactly the same set of tests. For example, with sequential programs we can associate a test with a pair consisting of a predicate on the input domain and a predicate on the output domain. For more general programming languages more general kinds of tests are needed.

In this thesis we are concerned with communicating, concurrent, and non-deterministic processes. For such systems, various notions of systems equivalence based on the reactions of systems to stimuli from the outside world have been presented in literature. We will here present the two that are closely related to the thesis work.

B. Strings Equivalence

The most straightforward approach for systems equivalence is to consider as equivalent two systems which can perform exactly the same sequences of observable actions.

Definition 2.2 ([12]) If $B_1$ and $B_2$ are two processes, then:

$$B_1 \sim_2 B_2 \text{ iff for all } t \in E^*: t \in Tr(B_1) \text{ if and only if } t \in Tr(B_2).$$

It is obvious that $\sim_2$ is an equivalence relation, and it is referred to as strings or trace equivalence.
The following simple example (Figure 2.1) taken from [12] illustrates the important aspects of this equivalence relation. It is obvious that strings equivalence is not suitable for testing in an environment where the possibility of nondeterministic interaction with that environment exists. In particular, this equivalence does not differentiate between systems for which certain communications with the environment will always be accepted (e.g. events $b$ and $c$ after $a$ in system $T_1$ of the example), or only at some times (e.g. always only one of the $b$ or $c$ after $a$ for system $T_2$, or possibly neither of $b$ or $c$ for system $T_3$, after event $a$).

C. Acceptance/Refusal-based Testing Equivalences

Alternative approaches to systems equivalence have been summarized and further expanded in [13] and [12]. Informally speaking, these equivalences also explore the "branching structure" of the system. When the processes involved may be nondeterministic, it is important not only to know whether given a particular test a process responds favourably or unfavourably, but also if a process responds consistently every time the test is performed. This natural notion of equivalence of two systems can be broken into two preorders on processes. The first is formulated in terms of the ability of a process to
respond positively to a test, the second in terms of its inability not to respond positively to a test [12].

Figure 2.2 and Figure 2.3 illustrate two essential considerations in defining such relations. A number of extensional equivalences and the testing equivalence [12] consider

\[ T_1 \]

\[ T_2 \]

Non-determinism and testing relations

Figure 2.2

the two systems in Fig. 2.2 as equivalent. Indeed, no external observer can distinguish them upon any experimentation that can be designed for testing through observation, unless rollback copies of the systems are available\(^5\). Similarly, acceptance/refusal based testing equivalences treat the systems \( T_1 \) and \( T_2 \) of Fig. 2.1 as non-equivalent, as would the previously outlined intuition suggest. However, the possibility of divergence is treated by different equivalences in different ways. Some of the theories treat the possibility of divergence (an example given in Fig. 2.3) as catastrophic, while others ignore it. In this thesis, we assume that the specifications of communication protocols are non-divergent,

\(^5\)Some initially defined (bisimulation) equivalences did indeed distinguish between the two systems such as those given in Fig. 2.2. It was however recognized as a drawback that the internal structure would need to be taken into account when testing such systems.
and that the same holds for their implementations. In practice, timeouts avoid this type of divergence.

As a specific example, we here quote the formal definition of the equivalence which is extensively researched in the conformance testing for communication protocols.

D. The Conformance Problem for Communication Protocols

Following is a formalization of the conformance problem.

**Definition 2.3** ([2]) Let $B_1$ and $B_2$ be two processes. Let $E$ represent a universe of labels of atomic protocol actions. $B_1 \, \text{conf} \, B_2$ if$^6$

$$
\forall \sigma \in \text{Tr}(B_2) \, \forall A \subseteq E \quad \text{if} \, \exists B'_1 \, \forall a \in A \, B_1 = \sigma \Rightarrow B'_1 \neq a \Rightarrow \\
\text{then} \, \exists B'_2 \, \forall a \in A \, B_2 = \sigma \Rightarrow B'_2 \neq a \Rightarrow. \quad \Box
$$

$B_1 \, \text{conf} \, B_2$ expresses that testing the traces of $B_2$ against the behaviour $B_1$ will not lead to deadlocks that could not occur if the same test was performed with the $B_2$ itself.

---

$^6$All subsequent notations are defined in Appendix 1.
2-2. **THEORETICAL BACKGROUND**

What is not guaranteed by the \textit{conf} relation, is that \( B_1 \) does not contain traces that \( B_2 \) does not specify. This particular problem is not possible to examine in a systematic and mathematical manner in general. All possible observers (universally quantified) would have to be examined, which clearly cannot be done by any analytic or systematic process.

\textit{Conf} is not an equivalence relation. However, it turns out that it is closely related to another relation, that can be used for the definition of an equivalence relation. To see this, consider the following definition and proposition.

**Definition 2.4** ([2]) Let \( B_1 \) and \( B_2 \) be two processes. Let \( E \) represent a universe of labels of atomic protocol actions. \( B_1 \) \( \text{red} \) \( B_2 \) if

\[
\forall \sigma \in E^* \forall A \subseteq E \quad \text{if} \quad \exists B_1' \forall a \in A \quad B_1' = \sigma \Rightarrow B_1' \neq a \Rightarrow \\
\text{then} \quad \exists B_2' \forall a \in A \quad B_2 = \sigma \Rightarrow B_2' \neq a \Rightarrow. \quad \Box
\]

This is precisely the earlier requirement related to the universal quantification of observers: by this definition, \( B_1 \) cannot have any traces that \( B_2 \) does not specify.

**Proposition 2.1** ([2])

1. \( B_1 \) \( \text{red} \) \( B_2 \) iff \( B_1 \) \( \text{conf} \) \( B_2 \) and \( \text{Tr}(B_1) \subseteq \text{Tr}(B_2) \);

2. \( \text{red} \) is a pre-order. \( \Box \)

Since \( \text{red} \) is a pre-order, it can be used for the definition of equivalence.

**Definition 2.5** Let \( B_1 \) and \( B_2 \) be two processes.

\[
B_1 \sim B_2 \text{ iff } B_1 \text{ \( \text{red} \) } B_2 \text{ and } B_2 \text{ \( \text{red} \) } B_1. \quad \Box
\]

This thesis research in validity relations and theoretical upper bounds for testing processes for multiconnection environments will be based on this type of relation.
2-3 The Metric Based Model of Testing

We use these newest results from general software testing theory and practice, and the communication protocols testing theory, to propose a new model for communication protocols testing. This model is briefly sketched in Figure 2.4. Fundamental to the model is the idea that if the theoretical upper bound of the testing process is derived from an extensional validity relation between a specification and an implementation, and it is systematically approximated over the entire execution space, it will yield a coverage oriented type of testing and address load conditions for large multiconnection protocol systems. The rest of this section details the essential requirements that we impose on this testing model.

Theoretical Soundness
The necessity of the theoretical soundness of test methods that target the reliability of released systems is the fundamental requirement in the model. The reliability of tested software cannot be addressed through testing with ad-hoc methods whose precise characteristics are not known. The method is therefore based on formal specifications of protocols and theoretical results in process algebras. This provides a strict mathematical foundation for subsequent analysis. All testing activities in the model are subject to rigorous mathematical treatment and evaluation.

Automated Treatment
Little incentive exists to use formal description techniques and formal methods based on these techniques if they do not increase the speed of development of the end product and reduce cost. This model is entirely based on the generic information about the processes, which is contained in the formal specification of the system. The model is therefore geared towards full automation. Full automation of testing activities is a requirement for coverage testing in order to handle the large number of test points needed, many more than are usually selected in classical testing methods.
2.3. THE METRIC BASED MODEL OF TESTING

Analytic Model of Testing
Figure 2.4
Chapter 3

A Metric Space of Protocol Control Behaviour

Efficient analysis of the behaviour of communication protocol systems is very difficult, since these systems involve large amounts of parallelism and recursion. If the behaviour of such systems is given in terms of all global states that can be reached on the interaction of the system and its environment, then this space is so large that it can be considered infinite for purposes of practical analysis.

For the successful test development and evaluation of such systems, a method is needed which can deal with the complexity of approximation of the global state space of such systems in a simple and intuitive manner, without sacrificing the mathematical soundness of the results.

Two problems are central to this approach. The first has to do with the formal representation of the global state space of a system which is to be approximated, and the second one is the choice of the mode of approximation of such a space. The choice of trace characterization of execution space of a system and its implications with respect to coverage testing, the generality of the approach, and finally its relationship to the
testing theory for process algebras is elaborated on following this section. The choice of the mode of approximation in this space, which singles out the temporal ordering of clusters of events generated by recursion as the primary approximation characteristics, is discussed next.

Based on this analysis, we develop a metric model of execution sequence space for analysing the control behaviour of protocol systems. Two specific simultaneous goals were to be achieved by the definition of the metric function. On one hand, the metric incorporating the notion of approximation of different temporal ordering of externally observable events must be well defined with respect to the operational semantics of the specification language. On the other hand, this metric is to be not only mathematically tractable, but also applicable in practical setting involving finite (and infinite) behaviours with recursion and parallelism.

This chapter includes a collection of results that show how these goals have been met. The theoretical development is organized in the following manner. We introduce a class of metric $dt$ which captures both the event pattern and the recursive characteristics of sequences of interactions in protocol control spaces. The metric model is based on the underlying notion of concurrency via interleaving and trace equivalence within the testing theory for labelled transition systems. The method involves a metric characterization of (infinite) trace sets $S$ of a protocol control behaviour. The metric $dt$ allows us to characterize all infinite execution sequences as limits of sequences of finite traces. The metric completion of the trace set of finite traces yields the space of all (finite and infinite) execution sequences of a process. This metric generates totally bounded metric spaces of execution sequences of protocol control behaviour providing the theoretical foundation for test selection and coverage for infinite systems. The metric interpretation of objects in this theory is used to derive metric-based definitions of average and maximum coverage of an execution sequence space by a selected set of tests. The existence of finite covers within complete metric spaces offers theoretical foundation for test selection techniques.
where the selected test cases approximate the (infinite) set from which the selection is
done to an arbitrary degree of precision.

3-1 General Issues

Several general issues are involved in the formulation of the metric characterization of
protocol control behaviour space. This section discusses three most important ones: (i)
a fault model applicable for coverage testing of real protocol implementations, and its
relationship to the formal characterization of the protocol control behaviour space by
traces (ii) the implications of the selection of trace sets as the test selection and coverage
evaluation domain, and (iii) the role of recursion in the protocol control behaviour.

3-1.1 Testing Real Protocol Implementations

In this section we summarize some observations on testing complex concurrent systems
in general and communication protocols in particular. This discussion explicates the
link between coverage testing, the fault model it subsumes, and the proposed formal
characterization and approximation mode of the execution sequence space.

We first turn our attention to some aspects of testing protocol implementations in
the setting where bugs really are: that is the entire (infinite) execution space of protocol
implementation behaviour.

Protocols are real-time systems that run continuously. During a (long) run, an imple-
mentation accumulates state and resource information. Although an implementation is
in a common state (for formal specifications whose semantics is representable by labelled
transition systems), or traverses the same node in the design graph (in general software
graph representation techniques), differences in accumulated information can cause im-
implementation to behave differently. Consequently, no state can really ever be considered
the same as another previous state unless we can guarantee the identical accumulation of
knowledge. For this reason, testing only with short test sequences is not representative of
the accumulated protocol behaviour, and may be misleading ([45]).

Second, software system implementations are really representable by discontinuous
functions. This has several implications on how we test. First, a valid testing strategy
should not \textit{a priori} exclude any point in the protocol space as nonrelevant for testing.
Second, we should choose many more points than is usual [25], to expose as many faulty
situations as possible. Of course, singular points may be hard to identify, but many, in
fact, will be exposed by some sufficiently similar situation in the protocol software system.

The final aspect we address in such systems has to do with our understanding of test
coverage. In systems of this complexity, ”subsumes” relationship between test strategies
cannot hold; indeed, the same test suite can be less powerful than itself if the subsumes
relationship is understood as exposing more faulty implementations [24]. This is because a
test suite may expose faults on one execution but not on another. This behaviour appears
as a nondeterministic protocol implementation, but is the consequence of the practical
impossibility to model and test \textit{all} aspects of a complex implementation. Unmodelled
aspects then surprise us as nondeterministic test outcomes.

As an illustration, consider the possible implementations of a generic protocol data
exchange phase and a connection establishment phase of a multiconnection protocol,
depicted in Figure 3.1.

This simple example illustrates how valid input events (data packets or incoming
connection requests) may easily introduce additional states (in this case labelled as crash)
in faulty protocol implementations. The behaviour (which differs from the one formally
specified), is both discontinuous and testable only by test sequences longer than a certain
minimum. The possible non-determinism of a faulty implementation indicates that any
An example of faulty implementations of a simple protocol

Figure 3.1

ordering of test suites by "subsume" relationship would be flawed\(^1\).

If a test selection and coverage evaluation method is to be applicable to a broad range of protocols, then it must address the issues mentioned above in some manner. The techniques which assume a finite fault model can apply exact selection techniques, fault targeting and subsume relationships for coverage. Exact selection techniques and finite fault models become impractical in infinite execution spaces with above characteristics, so we adopt trace equivalence to characterize an infinite fault model: every string is a possible fault. The choice of the trace equivalence as the fault model has several immediate implications on the resulting test development process.

First, trace preorders and equivalences in process algebras target all faults, since all

\(^1\)Note, however, that these errors could be detected by data flow analysis, using the knowledge of the internal details of the implementation under test (i.e. "white box" testing).
3-1. GENERAL ISSUES

strings are potential faults. This eliminates the drawbacks of the fault targeting based on non-uniform value weighting of possible tests which is characteristic of partition based testing techniques.

Secondly, the fault model does not exhibit preference for test cases based on test’s length, or puts any restrictions on the type or frequency of event occurrences. For protocols, this means that any number of network connections or prolonged data exchanges are possible candidates for test cases: these are exactly the main sources of load generating conditions in communication systems environments.

Finally, we observe that starting with the initial theoretical setting of labelled transition systems, the trace characterization of execution space yields traces that lead to all possible global system states in the control space of protocol behaviour.

Therefore, the type of testing subsumed by this fault model is exactly coverage testing as defined earlier.

3-1.2 Approximating Protocol Systems by Traces

There are two additional important aspects involved in the choice of the trace characterization of the execution space and trace equivalence as the applicable fault model.

The first is the obvious simplicity of using trace sets as test selection and coverage evaluation domain. The notion of traces is also well known to the practitioners in industry, and is considerably simpler for automated manipulation than more elaborate characterizations given in traces and refusal sets, their synchronization trees model, and other related formalisms. Trace refusal formalisms are, however, more accurate characterization of execution space of nondeterministic systems (communication protocols are almost always such). It is worth noting that it is possible to extend the metric theories of concurrent behaviour to tree equivalences, using the full power of extensional equivalences for
concurrent non-deterministic systems for metric characterization of execution space (e.g. direct application of the metric characterization of synchronization trees given in [18]). It is not clear that such theoretical results would be directly applicable in practice. In order to use the full power of such techniques in testing, it must be possible to establish the existence of the exact branching structure of the corresponding tree, after a trace has occurred. This means, that either (i) multiple copies of a system are available after a trace has been observed, or (ii) rollback copies of a system are available after a trace and one subsequent event have been observed. Neither of these are readily available in practice. In this thesis we restrict ourselves to deriving the strongest theoretical results which are also practically feasible, leaving the aspects of the system behaviour that are not theoretically covered, to an essentially engineering approach - designing testers such that the validity of the behaviour (which is possibly given in trace-refusal formalism), is indirectly determined through repetitive execution of test cases specified as traces [12, 2].

The second aspect is the generality of the approach. Trace equivalence, as a fault model, provides a link between test selection and testing theories [12, 2] which are applicable for specification techniques that admit labelled transition systems as their semantic model. All extensional equivalences proposed in the theoretical setting of testing protocol implementations for conformance, as well as the preorders defined in Chapter 6 of this thesis for testing protocol implementations for interoperability, are essentially based on trace preorders. This purely theoretical characteristic has two implications on practical testing and protocol tester design: (i) in order to test for any extensional preorder or equivalence, the suggested testing methodology [12] is to check for the existence of a trace first, and subsequently its possible further characteristics (i.e. stability of the occurrence of a trace, or the refusal sets associated with the trace) are examined and (ii) all exhaustive (e.g. the canonical) testers are trace equivalent to the original protocol specification. By deciding on the trace characterization of execution space, we are able to guarantee a broad applicability of metric based selection and evaluation techniques for all testing the-
ories that are essentially based on trace preorders, although their more advanced features may involve some other formalism.

3-1.3 Recursion and Test Selection in Protocol Systems

Based on the foregoing analysis, the investigation of the approximation process will be carried out in the spaces of all traces derived by interleaving semantics of concurrency for formal description techniques that allow specifications of concurrent nondeterministic protocol systems and have a finite labelset of actions that we call events. With the trace equivalence as the fault model in this space, there is a very large and possibly infinite number of potential fault strings from which a finite number of test cases must be selected. For the general case, there is obviously no exact method to deal with this fault model and terminate in finite time. Test selection must therefore viewed as an approximating process within infinite control behaviour spaces. The approximation must be based on a function which provides a measure of how well one execution sequence approximates another, and consequently how well one set of execution sequences approximates some other reference set. Such functions are metric functions, which evaluate the similarity of execution sequences based on some pre-defined criteria.

As mentioned earlier, the problem in defining the metric function is twofold. First, the criteria which will characterize the mode of approximation must be defined. Second, the notion of approximation resulting from these criteria must be well defined with respect to the operational semantics of the specification formalism - in particular, since the specification formalism allows for infinite sets of infinite traces, the mode of approximation must allow for approximating such infinite elements by finite ones. The second requirement of the metric function definition, i.e. the well-definedness of the metric function with respect to operational semantics of the formal model of labelled transition systems, is proven in a number of theorems following the definition of the testing distance. We here give some
observations which are relevant for defining the first requirement.

Many different properties of nondeterministic concurrent systems may be of interest, allowing for different notions of approximation based on intuitive perceptions of value of an execution sequence. In this thesis, we concentrate on the criteria which can be computed on the basis of the formal specification itself: in the theoretical setting of execution sequence spaces, these are individual events, their temporal ordering and their level of recursion.

The general notion of behaviour in LOTOS, CCS or other labelled transition system based specification formalisms is closely related to the operational semantics of step-wise evolution of a system. This allows the simple definitions of distance $da$ (Appendix C) to be directly applicable to such specifications. Informally, this distance is the well known distance on strings which measures the longest common prefix of two strings. In the context of these distances, as soon as two execution sequences differ in one step, their distance is evaluated, without regard to the future behaviour of these processes or the characteristics of that step. These distances are based on the individual events and their temporal ordering. They can be used to describe depth bound algorithms for test selection and coverage evaluation, and are directly applicable to testing protocols without multiconnection capabilities or prolonged repetitive event exchanges.

For the specifications of complex distributed systems and protocols, which typically involve multiple concurrent connections and other recursively specified processes, the metric characterization of execution space through individual events and their temporal ordering is not sufficient\(^2\). Steps generated by recursive instantiations of processes constitute major

\(^2\)We here note that in the specification formalism LOTOS both of these critical features: simultaneous network connections and prolonged repetitive event exchanges, are specified using recursion. Simultaneous multiple connections are specified by process instantiations within parallel composition operator, and repetitive event exchanges by process instantiations within sequential operator. If the distinction is needed, recursion derived through multiple parallel instantiations of a process will be termed "horizontal" recursion, and recursion derived from sequential consecutive instantiations of a process will be termed "vertical" recursion.
part of protocol behaviour. It is generally recognized that testing all levels of consecutive exchanges of the same event should be carefully balanced against testing new events or event combinations. Additionally, the main problem in comparing and selecting tests is due to time and space limitations, caused by the execution sequence explosion, which is exactly due to recursive calls to processes. For these two reasons, recursion has traditionally been a major factor influencing test selection. By capturing its characteristics it is possible to design more efficient and intuitive test selection and evaluation algorithms.

We illustrate this point on an excerpt from a typical communication protocol specification: ISO 8072 Transport Service [50] (excerpt from a LOTOS specification given in Appendix D, and a simplified version in Section 3-4). Figures 3.2 (b) and (c) each show three execution sequences s, t and u. Suppose that s, t and u in each of Figures 3.2 (b) and (c) are all valid test sequences generated by some test generation system. Applying the test selection techniques which do not recognize recursion, or metric models which do not allow for recursion in the definition of distance, one could be lead to give preference to test suites of the type \{s, t\}, which is intuitively a poor selection from the testing point of view. Since the test sequences s and t are very similar, the selection of \{s, u\} or \{t, u\} instead of \{s, t\} would be a preferred choice according to most test selection criteria (e.g. distinct-paths criterion, branch testing, as well as switch-cover paradigm).

A testing-oriented metric must therefore be defined in such a way to take into account recursive characteristics of test sequences, in order to allow the possibility of tradeoff between selecting a new event combination and the finer distinction in the level of recursion.

Based on these considerations, the following criteria are used to characterize the approximation mode of execution sequence space of communication protocols in this thesis: the occurrence of individual events in the execution sequence, their temporal ordering, and their level of recursion.
3-2 The Metric Space of Execution Sequences of Protocols

Protocol systems are highly recursive in nature. Consecutive actions may be identical and may originate from a single source (data packets arriving over a network) or from multiple sources (connection requests from several users.) We define a concise notation for the control component of an execution sequence by condensing consecutive identical actions into a single tuple: the action $a$ and its recursion level $\alpha$ yield the pair $(a, \alpha)$. 
The recursion level is the number of actions which have been condensed. For example, the execution sequence \( bbbacbaaa \) has the concise form \( (b, 3)(a, 1)(c, 2)(b, 1)(a, 3) \). This mapping of execution sequences of a process onto the space of the sequences of pairs \( (a_k, \alpha_k) \) we will denote as \( \xi \). We give the following formal definition of the mapping \( \xi \).

**Definition 3.1** (of the mapping \( \xi \)) Let \( s \) be a sequence of observable events of a process, \( s = \{m_i\}_{i=1}^{L}, L \in \mathbb{N} \cup \{\infty\} \), ( \( \mathbb{N} \) denotes the set of natural numbers). Denote by \( s(k) = m_k \in E \), the \( k \)-th element of \( s \) (\( E \) is the set of elementary protocol actions). With this sequence we shall associate a sequence of pairs \( \xi(s) = \{(a_k, \alpha_k)\}_{k=1}^{K}, K \in \mathbb{N} \cup \{\infty\} \) in the following way:

\[
\begin{align*}
a_1 &= s(1) = \ldots = s(\alpha_1) \\
&\neq s(\alpha_1 + 1) \\
a_2 &= s(\alpha_1 + 1) = \ldots = s(\alpha_1 + \alpha_2) \\
&\neq s(\alpha_1 + \alpha_2 + 1) \\
&\vdots \\
a_k &= s(\sum_{j=1}^{k-1} \alpha_j + 1) = \ldots = s(\sum_{j=1}^{k} \alpha_j) \\
&\neq s(\sum_{j=1}^{k} \alpha_j + 1), \ k = 3, 4, \ldots
\end{align*}
\]

where

\( a_k \in E, \ \alpha_k \in \mathbb{N}. \)

If for some \( k \),

\[
s(\sum_{j=1}^{k-1} \alpha_j + 1) = s(i) \text{ for all } i \geq \sum_{j=1}^{k-1} \alpha_j + 1,
\]

then we set
$a_k = s(\sum_{j=1}^{k-1} \alpha_j + 1)$, and $\alpha_k = \infty$.

In this mapping, a pair $(a_k, \alpha_k)$ represents a recursive cluster of the event $a_k$, whose level of recursion in the initial sequence $s$ is $\alpha_k$. The recursive clusters are temporally ordered in the sequence $\xi(s)$ by the relative position, in the sequence $s$, of the corresponding events $a_k$ which generate each recursive cluster. A single occurrence of an event has a level of recursion equal to 1; $\infty$ is used to characterize any infinite execution sequence.

### 3-2.1 The Definition of Testing Distance $dt$

From this point on, we assume that all execution sequences are represented in concise notation unless otherwise noted.

We will let $a, b, c, \ldots$ denote elements of the event labelset $E$, and use Greek letters $\alpha, \beta, \gamma, \ldots$ to denote the recursion level of an event cluster. Likewise, we will use $S$ to denote the set of all execution sequences of a protocol system specification, $T, U, \ldots$ to denote its subsets. By $s, t, u, \sigma, \ldots$ we denote execution sequences, and $\varepsilon$ an empty sequence. A dot is used as a concatenation symbol. Further details of the notation can be found in Appendix A.

Definitions of concepts in metric spaces that are used in this chapter can be found in Appendix C.

Let $p$, and $r$ be functions satisfying the following properties:

**P1** $\{p_k\}_{k=1}^{\infty}$ is a sequence of positive numbers such that $\sum_{k=1}^{\infty} p_k = p < \infty$.

**P2** $\{r_k\}_{k=0}^{\infty}$ is an increasing sequence in $[0, 1]$ such that $\lim_{k \to \infty} r_k = 1$. Put $r_\infty = 1$.

**Definition 3.2** (distance $dt$) Let $s, t$ be two (finite or infinite) execution sequences in $S$ and let $s = \{(a_k, \alpha_k)\}_{k=1}^{K}$, and $t = \{(b_k, \beta_k)\}_{k=1}^{L}$, $K, L \in \mathbb{N} \cup \{\infty\}$. The testing distance
between two execution sequences $s$ and $t$ is defined as

$$ dt(s, t) = \sum_{k=1}^{\max(K,L)} p_k \delta_k(s, t) $$

where

$$ \delta_k(s, t) = \begin{cases} |r_{a_k} - r_{b_k}| & \text{if } a_k = b_k \\ 1 & \text{if } a_k \neq b_k \end{cases} $$

If $s$ and $t$ are of different lengths then the shorter sequence is padded to match the length of the longer sequence, so that $\delta_k = 1$ for all $k$ in the padded tail.

We can now prove the following theorem.

**Theorem 3.1** $dt$ is a distance in $S$.

**Proof.**

1. $dt(s, t)$ is a non-negative real number for each $s, t$. Indeed, since

$$ 0 \leq \delta_k(s, t) \leq 1 \quad \text{for all } k = 1, 2, \ldots, \max\{K, L\} $$

we have

$$ 0 \leq \sum_{k=1}^{n} p_k \delta_k(s, t) \leq \sum_{k=1}^{n} p_k \leq \sum_{k=1}^{\infty} p_k = p < \infty \quad \text{for all } n \in \mathbb{N}. $$

Thus, $\sum_{k=1}^{\infty} p_k \delta_k(s, t)$ converges and consequently

$$ 0 \leq dt(s, t) = \sum_{k=1}^{\max\{K,L\}} p_k \delta_k(s, t) < \infty. $$

2. $dt(s, t) = 0$ if and only if $s = t$.

Indeed, $dt(s, t) = 0$ iff $\delta_k(s, t) = 0$ for all $k = 1, 2, \ldots, \max\{K, L\}$. Clearly, $\delta_k(s, t) = 0$ for all $k = 1, 2, \ldots, \max\{K, L\}$ is equivalent to $K = L$ and $a_k = b_k$ and $\alpha_k = \beta_k$ for all $k = 1, 2, \ldots, K = L$, what gives $s = t$. 
3. \( dt(s, t) = dt(t, s) \)

It follows from the definition of \( \delta_k(s, t), \ k = 1, 2, \ldots, \max\{K, L\} \), that

\[
\delta_k(s, t) = \delta_k(t, s), \ k = 1, 2, \ldots, \max\{K, L\}.
\]

Consequently

\[
dt(s, t) = \sum_{k=1}^{\max\{K, L\}} p_k \delta_k(s, t) = \sum_{k=1}^{\max\{K, L\}} p_k \delta_k(t, s) = dt(t, s)
\]

4. Now we will show that the triangular inequality holds for \( dt \).

Let \( s, t, u \) be three (finite or infinite) execution sequences in \( S \). The following holds:

\[
\delta_k(s, u) + \delta_k(u, t) \geq \delta_k(s, t), \ k = 1, 2, \ldots, \max\{K, L, M\}
\]

Consequently,

\[
dt(s, u) + dt(u, t) = \sum_{k=1}^{\max\{K, M\}} p_k \delta_k(s, u) + \sum_{k=1}^{\max\{L, M\}} p_k \delta_k(u, t)
\]

\[
\geq \sum_{k=1}^{\max\{K, L\}} p_k \delta_k(s, t)
\]

\[
= dt(s, t)
\]

Hence

\[
dt(s, u) + dt(u, t) \geq dt(s, t)
\]

and the triangular inequality indeed holds. \( \square \)
3-2. THE METRIC SPACE OF EXECUTION SEQUENCES OF PROTOCOLS

3-2.2 The Interpretation of the Distance $dt$

A. An Intuitive Comparison of Distance Functions

Perhaps the most straightforward way to illustrate the distance $dt$, is to compare it to the other well known distance, denoted as $da$, used in the theoretical investigations of the theory of concurrency. This distance was previously mentioned, (Ch. 3-1.3), and it is evaluated on the basis of individual events and their temporal ordering only. We show what has been gained by introducing the notion of recursion into the definition of distance $dt$.

In Figure 3.3 we give a simple example to motivate the definition of the distance $dt$. We will let $da(s, t) = 2^{-k}$, where $k$ is the length of the longest common prefix of strings $s$ and $t$. (Note that strings in this metric are not represented concisely.) We will let $dt$ be defined as in Definition 3.2, and $p_k = 1$, $k = 1, 2, 3$, otherwise $p_k$ and $r_k$ satisfy the constraints $P1$ and $P2$.

Then the following evaluations of the distances between representative execution sequences hold.

1. $da(s, u) < da(s, t)$, $dt(s, t) < dt(s, u)$

2. $da(t, s) = da(t, w) = da(t, u)$

3. $dt(t, s) < dt(t, w) < dt(t, u)$

By failing to recognize recursion, the distance $da$, if applied in testing, would generate:

1. counterintuitive preference for test cases, where the occurrence of a completely new event (abort) and new test scenario (connect - data - abort) is treated as "closer" or more similar, therefore less valuable, than a sequence involving exactly the same
events (connect, data and release) and exactly the same event ordering (connect -
data - release), except for one extra occurrence of one event (case 1.)

2. ambiguity in test case preference, where an entirely new event combination (data.abort)
and additional event occurrence (abort) is treated the same as all other previously
encountered event orderings and event occurrences, and ambiguity with respect to
the extent of differing in the level of recursion (case 2., vs. case 3.)

One can therefore view the distance $d_a$ as approximating similar behaviors by basing
its decision on the granularity of a single event, and the first occurrence of the difference
at that level of granularity. The distance $d_t$ does the approximation on a more coarse
level, that of clusters of events generated by consecutive occurrences of the same event.
Therefore, a new event cluster \((\text{abort}, 1)\) will be recognized by the metric, and consequently the sequence \(u\) will differ by at least \(p_3\) from any other sequence in the example. In addition, the decision is not reached based on the first occurrence of the difference at that level of granularity. Rather, the cumulative effect of differences in individual clusters and their respective levels of recursion is scanned along compared sequences and assigned an appropriate measure of mutual approximation.

**B. The Parameterization of \(d_t\)**

The distance \(d_t\) therefore requires two parameters. The similarity of a temporal ordering of clusters of events is measured by the sequence \(\{p_k\}_{k=1}^{\infty}\), where each individual \(p_k\) represents the weight related to differing with respect to that position of the cluster in an execution sequence. Several options are allowed, according to the Definition 3.2. In particular, the weight of the clusters of events related to their absolute position within the execution sequence may be defined such that

1. the weights converge in a monotonous decreasing sequence (Figure 3.4\(^3\), function \(p_k^1\))

2. the weights are equal up to any finite \(n \in \mathbb{N}\), after which they converge in a decreasing sequence (Figure 3.4, function \(p_k^2\))

3. the weights are different but finite, up to any finite \(n \in \mathbb{N}\), after which they converge in a decreasing sequence (Figure 3.4, function \(p_k^3\))

The similarity of recursive depth for the same position of event clusters and identical events \(a_k\), along an execution sequence is measured by the function \(\delta_k\). Notice that in the definition of \(d_t\), this measure is not constant for a constant difference in recursive depth.

---

\(^3\)Although we use the functions \(p_k\) and \(r_k\) only at discrete intervals, we plot them as continuous functions to better observe trends in their behaviour.
The values of $r_k$, by property \textbf{P2}, may be computed from a function that fluctuates between concave/convex on a finite subdomain, (i.e. for (initial) $r_k$ values), but is convex on its subdomain starting at some large enough $k$. For simplicity of discussion, from now on we will assume that $r_k$ is convex on its entire domain. Under such assumption, if two sequences differ at two event clusters in recursion only, more weight will be given if the event clusters are carrying recursion depths of 2 and 3 (for the two sequences respectively), than if the event clusters carry recursion depths of one thousand and 2 and one thousand.
3-2. THE METRIC SPACE OF EXECUTION SEQUENCES OF PROTOCOLS

and 3 (again respectively to the two sequences). To see this, consider the evaluation of the function $\delta_k$ on an example of the function $r_k$, given in Figure 3.5, where $\alpha_3 - \alpha_1 = \alpha_7 - \alpha_5$, $\delta_{k_1} = |r_{\alpha_5} - r_{\alpha_7}|$, $\delta_{k_2} = |r_{\alpha_1} - r_{\alpha_3}|$.

![Diagram of function $\delta_k$](image)

Evaluation of the function $\delta_k$

Figure 3.5

Under the convexity assumption, the following result holds.

**Proposition 3.1** Let $s$ be a sequence $s = \{(a_k, \alpha_k)\}_{k=1}^K$, and let $t$ be a sequence $t = \{(b_k, \beta_k)\}_{k=1}^L$ such that $K = L$ and $a_k = b_k$ for all $k = 1, 2, \ldots, K = L$. Let $\alpha_{k_0} \neq \beta_{k_0}$ for one specific $k_0 \in \{1, 2, \ldots, K\}$, and let $\alpha_k = \beta_k$ for all $k \in \{1, 2, \ldots, K\} \setminus \{k_0\}$.

1. for a fixed $\alpha_{k_0}$, and for $\beta_{k_0} > \alpha_{k_0}$, the distance $d_t(s, t)$ increases as $\beta_{k_0}$ increases
2. for a fixed $\alpha_{k_0}$, and for $\beta_{k_0} < \alpha_{k_0}$, the distance $d_t(s, t)$ increases as $\beta_{k_0}$ decreases

---

4This characteristic of the function $\delta_k$ enables the generation of totally bounded metric spaces. For the completeness result of the metric spaces, it would be sufficient to consider a simpler function $\delta_k$ which is constant with respect to the constant difference in recursion depth [59].
for a fixed difference $\alpha_{k_0} - \beta_{k_0}$, the distance $d_t(s,t)$ decreases as $\alpha_{k_0}$ increases.

**Proof.** The proof follows directly from the Definition 3.2. and the convexity assumption. □

Notice that, if the convexity assumption for function $r_k$ does not hold, then Proposition 3.1 holds for large enough $\alpha_{k_0}$ and $\beta_{k_0}$.

The practical significance of the distance $d_t$ being parametrized by two functions, weighing the difference in (i) event clusters at a position $k$ within two sequences, and (ii) recursion levels within respective event clusters, is in that it enables test designers to specify their individual criterion of the relative importance of testing new event pattern versus a finer distinction in the level of recursion. Figure 3.6 illustrates several simple cases. The specific functions used are $p^1_k$ (Figure 3.6 (a)) and $p^2_k$ (Figure 3.6 (b)) (both functions are taken from Figure 3.4), and constant values for $\delta_k = \delta_1 = 0.5$ and $\delta_k = \delta_2 = 0.25$ for all $k \in \mathbb{N} \cup \{\infty\}$. The area under each curve $p^i_k \delta_1$ ($p^i_k \delta_2$), for $i = 1, 2$, is the distance between any two sequences that (i) have $a_k$ the same for all $k$, and (ii) all event clusters constantly differ in recursion level by the value $\delta_1$ ($\delta_2$). The area under the curve $p^1_k$ ($p^2_k$) is the distance between any two sequences that constantly differ in $a_k$ for each $k$ (i.e. each event cluster $(a_k, \alpha_k)$). The difference area between the lines $p^1_k$ and $p^1_k \delta_1$ (the unshaded area in Figure 3.6), is the value of the difference between testing new event clusters at all positions $k$, versus refining the recursion to the value $\delta_1$ within each event cluster of some existing test sequence. Similar holds for the difference areas of other $p^i_k$ and $p^i_k \delta_j$ combinations, for $i, j = 1, 2$. The difference area between the curves, between the vertical lines $i$ and $i + 1$ is the value by which a single cluster at the position $i$ contributes to the distance between two execution sequences, in cases where these differ in that cluster by recursion $\delta_1$ ($\delta_2$), or in the event $a_i$ itself.
3-2.3 The Finiteness of $\epsilon$-dense Covering for the Metric Space $(S, dt)$

The choice of trace equivalence as the applicable fault model for coverage testing of communication protocol implementations, leaves us with one outstanding problem: selecting a finite number of test cases from the infinite number of strings in the trace set of a protocol process. The following theoretical result resolves that problem.
Theorem 3.2 The space \((S, dt)\) is totally bounded.

Proof. Let \(\epsilon > 0\) be arbitrary. Let \(k_\epsilon\) be such that \(\sum_{k=k_\epsilon+1}^{\infty} p_k < \epsilon/2\). Put \(p = \sum_{k=1}^{\infty} p_k\).

Let \(n_\epsilon \in \mathbb{N}\), be such that \(r_k > 1 - \eta, \eta = \epsilon/2p\) for all \(k \geq n_\epsilon\). The set

\[ F = \{ \{ (a_j, \alpha_j) \}_{j=1}^{k_\epsilon}, \ a_j \in E, \ \alpha_j \in \{1, \ldots, n_\epsilon\} \} \]

is finite. The set of all spheres centered at elements in \(F\) and of radius \(\epsilon\) covers the entire space.

Indeed, let \(s = \{(b_j, \beta_j)\}_{j=1}^{L}\) be arbitrary. Then there exists \(t = \{(a_j, \alpha_j)\}_{j=1}^{k_\epsilon}\) in \(F\), such that \(a_j = b_j\), for \(j = 1, \ldots, k_\epsilon\). Put \(\alpha_j = \beta_j\) if \(\beta_j \leq n_\epsilon\) or \(\alpha_j = n_\epsilon\) if \(\beta_j > n_\epsilon\). It follows that

\[
\begin{align*}
\text{}dt(s, t) &= \max\{k_\epsilon, L\} \\
&= \sum_{k=1}^{\max\{k_\epsilon, L\}} p_k \delta_k(s, t) \\
&\leq \sum_{k=1}^{k_\epsilon} p_k |r_{\alpha_k} - r_{\beta_k}| + \sum_{k=k_\epsilon+1}^{\infty} p_k \\
&\leq \sum_{k=1}^{k_\epsilon} p_k |r_{n_\epsilon} - 1| + \sum_{k=k_\epsilon+1}^{\infty} p_k \\
&< \eta \sum_{k=1}^{k_\epsilon} p_k + \epsilon/2 < \eta p + \epsilon/2 = \epsilon
\end{align*}
\]

which completes the proof of this theorem. \(\Box\)

The total boundness result which implies the existence of finite covers for infinite metric spaces \((S, dt)\), offers theoretical foundation for test selection techniques where the selected test cases approximate the (infinite) set from which the selection is done to an arbitrary degree of precision.
3-2. **THE METRIC SPACE OF EXECUTION SEQUENCES OF PROTOCOLS**

### 3-2.4 The Completeness of Space \((S, dt)\)

Recall that \(S\) is the space of all sequences \(\{(a_k, \alpha_k)\}_{k=1}^{K}\) where \(K \in \mathbb{N} \cup \{\infty\}\). If \(K = \infty\) then \(a_k \in E\), \(\alpha_k \in \mathbb{N}\) for all \(1 \leq k < \infty\). If \(K\) is finite, then \(a_k \in E\), for all \(1 \leq k \leq K\), and \(\alpha_k \in \mathbb{N}\) for all \(1 \leq k < K - 1\), and \(\alpha_K \in \mathbb{N} \cup \{\infty\}\).

Define the space \(\hat{S}\) to consist of all sequences \(\{(a_k, \alpha_k)\}_{k=1}^{K}\) where \(a_k \in E\), \(\alpha_k \in \mathbb{N} \cup \{\infty\}\) for all \(1 \leq k < \infty\) if \(K = \infty\), and \(1 \leq k \leq K\), if \(K\) is finite.

In a natural way the metric \(dt\) extends to the space \(\hat{S}\).

As a consequence of the following theorem we will obtain that an extension \(\hat{S}\) of the space \(S\) is complete.

**Theorem 3.3** Let \(\{s_n\}_{n=1}^{\infty}\), where \(s_n = \{(a^n_j, \alpha^n_j)\}_{j=1}^{K_n}\), be a sequence in \(\hat{S}\). The following statements are equivalent:

(i) The sequence \(\{s_n\}_{n=1}^{\infty}\) is a Cauchy sequence.

(ii) The sequence \(\{K_n\}_{n=1}^{\infty}\) is eventually constant (Definition C.9) or \(K_n \rightarrow \infty\) \((n \rightarrow \infty)\), the sequence \(\{a^n_j\}_{n=1}^{\infty}\) is eventually constant, and the sequence \(\{\alpha^n_j\}_{n=1}^{\infty}\) is eventually constant or \(\alpha^n_j \rightarrow \infty\) \((n \rightarrow \infty)\).

(iii) The sequence \(\{s_n\}_{n=1}^{\infty}\), converges in the space \((\hat{S}, dt)\).

**Proof.** The implication (iii) \(\Rightarrow\) (i) is well known. We now prove the implication (i) \(\Rightarrow\) (ii). Assume that \(\{s_n\}_{n=1}^{\infty}\) is a Cauchy sequence.

We first prove that the sequence \(\{K_n\}_{n=1}^{\infty}\) is eventually constant or \(K_n \rightarrow \infty\) \((n \rightarrow \infty)\). For arbitrary \(m \in \mathbb{N}\) consider the sets

\[
U_m = \{n \in \mathbb{N} : K_n \geq m\} \quad \text{and} \quad L_m = \{n \in \mathbb{N} : K_n < m\}.
\]
Clearly $U_m$ is a nonincreasing and $L_m$ is nondecreasing sequence of subsets of natural numbers. If $j \in U_m$ and $k \in L_m$, then $K_j \geq m > K_k$. Therefore by the definition of $\delta_m$ we have that $\delta_m(s_j, s_k) = 1$. Consequently $dt(s_j, s_k) \geq p_m > 0$. Since $\{s_n\}_{n=1}^{\infty}$ is a Cauchy sequence not both sets $U_m$ and $L_m$ can be infinite. We distinguish the following two cases:

(a) All $U_m$, $m \in \mathbb{N}$, are infinite (i.e. all $L_m$, $m \in \mathbb{N}$, are finite). This implies that $K_n \to \infty$ ($n \to \infty$).

(b) At least one $U_m$ is finite. Let $m_0$ be a maximum natural number for which $U_{m_0}$ is infinite. Then $L_{m_0}$ is finite. If $n_0$ is an upper bound for $L_{m_0}$ then $K_n = m_0$ for all $n \geq n_0$, i.e. $\{K_n\}$ is eventually constant.

Now we prove that the sequence $\{a_j^n\}_{n=1}^{\infty}$ is eventually constant. For every $\varepsilon > 0$ there exist $N_\varepsilon$ such that $dt(s_n, s_m) < \varepsilon$ for all $n, m > N_\varepsilon$. Choose $\varepsilon$ such that $\varepsilon < p_k$. Then

$$p_k \delta_k(s_n, s_m) \leq dt(s_n, s_m) < \varepsilon < p_k, \text{ for all } n, m > N_\varepsilon. \quad (3.1)$$

Consequently $\delta_k(s_n, s_m) < 1$ for all $n, m > N_\varepsilon$. From the definition of $\delta_k$ we conclude that $a_k^n = a_k^m$ for all $n, m > N_\varepsilon$. Therefore the sequence $\{a_j^n\}_{n=1}^{\infty}$ is eventually constant.

Now we prove that the sequence $\{a_j^n\}_{n=1}^{\infty}$ is eventually constant or $a_j^n \to \infty$ ($n \to \infty$). The proof is similar to the first part of this proof. For arbitrary $m \in \mathbb{N}$ consider the sets

$$U_m = \{n \in \mathbb{N} : a_k^n \geq m\} \quad \text{and} \quad L_m = \{n \in \mathbb{N} : a_k^n < m\}.$$

Clearly $U_m$ is a nonincreasing and $L_m$ is nondecreasing sequence of subsets of natural numbers. If $j \in U_m$ and $l \in L_m$, then $\alpha_j^k \geq m > \alpha_k^l$. Since the function $r$ is increasing by the definition of $dt$ we have that

$$dt(s_j, s_l) \geq p_k(r_{\alpha_j^k} - r_{\alpha_j^k}) \geq p_k(r_m - r_{m-1}) > 0.$$
Since \( \{s_n\}_{n=1}^\infty \) is a Cauchy sequence not both sets \( U_m \) and \( L_m \) can be infinite. We distinguish two cases:

(a) All \( U_m, \ m \in \mathbb{N}, \) are infinite (i.e. all \( L_m, \ m \in \mathbb{N}, \) are finite). This implies that \( \alpha_k^n \to \infty \ (n \to \infty). \)

(b) At least one \( U_m \) is finite. Let \( m_0 \) be a maximum natural number for which \( U_{m_0} \) is infinite. Then \( L_{m_0} \) is finite. If \( n_0 \) is an upper bound for \( L_{m_0} \) then \( \alpha_k^n = m_0 \) for all \( n \geq n_0, \) i.e. \( \{\alpha_k^n\}_{n=1}^\infty \) is eventually constant. This proves the implication (i) \( \Rightarrow \) (ii).

Assume (ii). The first part of (ii) implies that we can define \( K_\infty = \lim_{n \to \infty} K_n, \) where \( K_\infty \in \mathbb{N} \cup \{\infty\}. \) For \( 1 \leq k \leq K_\infty \) put \( a_k = \lim_{n \to \infty} a_k^n \) and \( \alpha_k = \lim_{n \to \infty} \alpha_k^n, \) where \( \alpha_k \in \mathbb{N} \cup \{\infty\}. \) Put \( s = \{(a_k, \alpha_k)\}_{k=1}^{K_\infty}. \) It is clear from this definition that if \( K_\infty \) is finite than the sequence \( \{s_n\}_{n=1}^\infty \) is eventually constant. In fact it eventually equals \( s. \) Therefore it converges. If \( K_\infty = \infty \) the sequence \( \{s_n\}_{n=1}^\infty \) converges to \( s. \) Indeed, let \( \epsilon > 0 \) be arbitrary. Let \( k_\epsilon \in \mathbb{N} \) be such that \( \sum_{k=1}^{k_\epsilon} p_k < \epsilon/2. \) Since each of the sequences \( \{a_k^n\}_{n=1}^\infty, \ 1 \leq k \leq k_\epsilon \) are eventually constant there exists \( n_\epsilon \in \mathbb{N} \) such that \( a_k^n = a_k \) for all \( 1 \leq k \leq k_\epsilon \) and for all \( n \geq n_\epsilon. \) Since \( \lim_{n \to \infty} \alpha_k^n = \alpha_k \) for all \( 1 \leq k \leq k_\epsilon \) we can choose \( n_\epsilon \) such that \( r_{\alpha_k} - r_{\alpha_k^n} = \delta_k(s_n, s) < \epsilon/2p \) (here \( p = \sum_{k=1}^\infty p_k \)) for all \( 1 \leq k \leq k_\epsilon \) and for all \( n \geq n_\epsilon. \) Therefore for all \( n > n_\epsilon \) we have:

\[
\begin{align*}
dt(s_n, s) &= \sum_{k=1}^{\infty} p_k \delta_k(s_n, s) \\
&\leq \sum_{k=1}^{k_\epsilon} p_k \delta_k(s_n, s) + \sum_{k=k_\epsilon+1}^{\infty} p_k \\
&< \frac{\epsilon}{2p} \sum_{k=1}^{k_\epsilon} p_k + \frac{\epsilon}{2} \\
&\leq \frac{\epsilon}{2p} p + \frac{\epsilon}{2} = \epsilon.
\end{align*}
\]
This proves that the sequence \( \{s_n\}_{n=1}^{\infty} \) converges and the implication (ii) \( \Rightarrow \) (iii) is proved.
\( \square \)

An immediate consequence of this theorem is:

**Corollary 3.4** The metric space \((\hat{S}, dt)\) is complete.

It is a general result in the theory of metric spaces, that a metric space which is totally bounded and complete, is also compact. Therefore, we state the following theorem as a simple consequence of the previous results.

**Theorem 3.5** The metric space \((\hat{S}, dt)\) is compact.

### 3-3 Control Coverage Evaluation

By defining a metric which measures how far a test sequence is from its closest potential neighbor (another test sequence) in some other set of execution sequences of the specification or in a given test suite, the intuitive notion of similarity of test cases is described in analytic terms. Intuitively, the smaller the distance, the closer or more similar a test sequence is to some specified or already selected test sequence. Such a notion of distance between two sequences can be extended to sets of sequences. With this extension, the metric \( dt \) allows us to view the selected test suite not as an arbitrary or subjective collection of test cases, but rather a metric space for which the distance from the original specification or oracle test suite can be calculated to facilitate its analysis and comparison with other test suites.

The definition of the control coverage is based on the distance between each element of the set of execution sequences (i.e. the superset (of tests) derived from the specification), and its closest approximant in the set of selected sequences (or selected test suite). The
formal definition of a distance of a point from a set in metric spaces, is given in Appendix C, Definition C.6.

Also, the comparisons which are based on averages, in general have to be evaluated for all elements of the infinite execution space. The problem is resolved by basing the evaluation on the metric comparison of truncations of execution sequences. The following definition formally specifies the concept of truncations of execution sequences.

**Definition 3.3** Let \( \{a_k\}_{k=1}^{L} \) be a sequence, of events \( a_k \), in the space \( S \) of sequences of observable interactions (i.e. not in concise representation), and let \( n \) be a natural number. A truncation of length \( n \) of this sequence is defined as follows: If \( L > n \), then the truncation is the sequence \( \{a_k\}_{k=1}^{n} \), and if \( L \leq n \), then the truncation is the sequence itself, \( \{a_k\}_{k=1}^{L} \). □

Denote the set of all truncations of length \( n \) by \( S_n \). Note that it is not necessarily true that \( S_n \subseteq S_{n+1} \). Figure 3.7 illustrates the example ISO 8072 specification trace space, where the stratification of the space is based on the length of truncations.

**3-3.1 A Coverage Metric for Pseudo-Exhaustive Testing Strategies**

General acceptance testing, final system integration testing, and some availability testing strategies, will produce a final verdict on the quality of the implementation system in terms of how well it performs (and conforms) according to the entire specification. This is of course impossible to judge precisely since the domain of execution sequences is infinite in general. In terms of an objective measure, however, the following estimate can be determined through the metric \( dt \).

To measure the closeness of approximation of a (super) set of execution sequences of
a specification $S$ by its subset, a test suite $T$, we calculate the supremum of the average distances from elements of $S$, to the test suite $T$.

**Definition 3.4** Let $S$ be the set of all execution sequences of a protocol specification and $T$ its selected test suite. We assume that for any $T$ there exists a natural number $n_T$ such that $T \subset S_{n_T}$ (i.e. we assume that $T$ is finite and that there always exists the longest test sequence).

Let $k_n$ be the number of different elements of $S_n$. For a natural number $n$, $n \geq n_T$, we define $v_n$ to be the average of distances\(^5\) from elements of $S_n$ to $T$. Therefore, for infinite

\(^5\)The distance $dt$ is, of course, measured on concise representations of execution sequences.
3-3. CONTROL COVERAGE EVALUATION

\[ v_n = \sum_{k_n} \{dt(x, T), x \in S_n\} \]

As a measure of how well \( T \) approximates \( S \), we will take the normalized supremum of \( v_n \)

\[ a(T) = \sup \{v_n : n \geq n_T\} \]

\[ \sum_{k=1}^{\infty} p_k \]

**Definition 3.5** The average coverage of a set \( S \) by a selected set \( T \) is defined as

\[ CovAve(T) = 1 - a(T) \]

Notice that, removing a point from a set \( T \) cannot decrease the infimum in the calculation of \( v_n \), and conversely, adding a point to the set \( S \) cannot decrease the supremum of the distances calculated from the points of that set. Therefore, we can formulate the following remark.

**Remark 3.1** (Properties of \( CovAve \))

1. If \( T_1 \subseteq T_2 \) then \( CovAve(T_1) \leq CovAve(T_2) \).
2. \( 0 \leq CovAve(T) \leq 1 \)
3. \( CovAve(S_n) \to 1, n \to \infty \)
4. Coverage of an empty suite is 0 by definition.

3-3.2 A Coverage Metric for Strategies With Pre-defined Target Test Sequences

If a certain subset of a specification is pre-selected as a focal point of the testing campaign, as in safety-critical testing, test-purpose driven testing, estimated-usage profile testing,

\(^6\)\( v_n \) are positive real numbers that lie in an interval bounded from above, so supremum always exists.
the knowledge of specific execution sequences that need to be well approximated in the original specification may be available. In these cases, we are interested in knowing how well a given test suite performs in the worst case.

For the measure of the quality of coverage of a test suite $T$ with respect to sets of execution sequences $S$, we will take the supremum of distances from the elements of the complement of $T$ with respect to $S$, to the set $T$.

**Definition 3.6** Let $S$ be the set of all execution sequences of a protocol specification and $T$ its proposed test suite. (Note that the set $S$ may be infinite.) Let $S \setminus T$ be the complement of the set $T$ with respect to $S$. We define $m(T)$ to be the normalized supremum of distances from the elements of the complement $S \setminus T$ to the set $T$:

$$m(T) = \frac{\sup \{ d_l(x, T) : x \in S \setminus T \} }{\sum_{k=1}^{\infty} p_k}$$

**Definition 3.7** The worst-case coverage of a set $S$ by a selected set $T$ is defined as

$$\text{CovMax}(T) = 1 - m(T)$$

Using the same arguments as in formulating the Remark 3.1., we state the following.

**Remark 3.2** (Properties of CovMax)

1. If $T_1 \subseteq T_2$ then $\text{CovMax}(T_1) \leq \text{CovMax}(T_2)$.
2. $0 \leq \text{CovMax}(T) \leq 1$
3. Coverage of an empty suite is 0 by definition.

In both definitions of coverage, it is implicit that sets $S$ are infinite. If sets $S$ are finite, the calculation of average distance $a(T)$ can be done without intermediate $S_n$ sets, and supremums in both definitions are maximum distances.
It is important to note that, in metric based theory, test coverage is expressed in independent objective terms rather than by "subsumes" relationships. The meaning of coverage figures is therefore purely analytic. We expect that poor coverage figures in this theory will help point out undertested areas and prompt examination of test suites that differ widely in their coverage of each other. Similarly, we expect that good coverage figures will increase our confidence in the tested implementations. This is in keeping with the theoretical results already considered in the introductory part of this chapter (Section 3-1.1), and the intention of the theory to provide theoretical basis for testing systems which are possibly non-deterministic in their implementation.

### 3-4 Metric Based Test Selection

The ability of the theory to support building effective tools and techniques for communication protocols engineering is essential in the development of reliable communication software. In this section, the testing distance metric is incorporated into a test selection algorithm. A simple procedure using a stratified selection approach for the control space of system behaviour is presented. The essence of the selection method is the generation of $\epsilon$-dense sets of test sequences (the definition of $\epsilon$-density is given in Appendix C, Definition C.5), that approximate some original test suite to some target accuracy of distance $\epsilon$ in a convergent manner, ultimately establishing the string equivalence between the two sets involved.

#### 3-4.1 Metric Based Test Selection Algorithm

The selection algorithm for the metric based method (MB selection method for short), is a multi-pass process in which each pass is realized by the selection function $\text{SELECT}(T_0, G, \epsilon, C)$, which returns a selected set $T$, being an $\epsilon$-dense (Definition C.5) subset of the original set.
CHAPTER 3. A METRIC SPACE OF PROTOCOL CONTROL BEHAVIOUR

Let $G$ be a set of test cases generated by some test generator, such that the cost of $T$ (which includes the initially selected set $T_0$) is less than some given threshold cost $C$.

The cost function of a test case can be defined to represent the resources (time and space) required to execute that test case, e.g. its length. The cost of a set of test cases can be defined simply as the sum of the cost of the individual test cases in the set.

**Metric-Based Test Selection Algorithm**

1. **Step 1.** Initially, the selected set $T$ is empty, $G$ is the given (generated) set of test cases, $\epsilon$ is the initial target distance, and $C$ is the given cost threshold for the selected set.

2. **Step 2.** While $\text{Cost}(T) < C$ do $T = \text{SELECT}(T, G, \epsilon = \epsilon/k, C)$ for some scaling factor $k > 1$, applied at each iteration (that is, each pass).

3. **Step 3.** Stop. No further pass is possible because any other test case added to the set $T$ of test cases selected so far violates the cost constraint.

The function $\text{SELECT}(T, G, \epsilon, C)$ (that is, each pass in Step 2 of the above general algo-
rithm) can be specified as follows:

Step 2.1 Let \( X = G \setminus T \), i.e. \( G \) excluding \( T \).

Step 2.2 If \( X \) is empty return \( T \) and exit.

Else remove (randomly) some test case \( t \) from \( X \).

Step 2.3 If \( dt(t, T) < \epsilon \) goto Step 2.2.

Step 2.4 If \( \text{Cost}(T \cup \{t\}) < C \) then add \( t \) to the selected set \( T \).

Step 2.5 Goto Step 2.2.

In this algorithm we use a greedy heuristic in selecting the next test case \( t \) (Step 2.2). In fact, we simply select the first randomly encountered test case which is sufficiently far (greater than \( \epsilon \)) from the already selected test cases (Step 2.3), and which satisfies the cost constraint (Step 2.4).

The scaling factor \( k \) is such that the target distance \( \epsilon \) decreases with each pass of the algorithm. The multipass algorithm therefore generates Cauchy sequences over the successive passes. This process of Cauchy sequence generation is illustrated in Figure 3.8, where test sequences are selected in the following order:

Pass 1: \( t_{s1} \), density \( \epsilon_1 \).

Pass 2: \( t_{s2}, t_{s3} \), density \( \epsilon_2 \).

Pass 3: \( t_{s4}, \ldots, t_{s6} \), density \( \epsilon_3 \).
Then, the initial segments of generated Cauchy sequences are, e.g., $t_{s_1}, t_{s_2}, t_{s_4}$, also $t_{s_1}, t_{s_3}, t_{s_6}$ and $t_{s_1}, t_{s_5}$.

![Generating Cauchy sequences](image)

The multipass algorithm presented above eventually terminates if and only if the initial set of test sequences $G$ is finite. One possibility is that $G$ can also be generated from a random test generator, and the MB test selection algorithm run simultaneously with it. Assuming unlimited resources, set $G$ would be infinite. In this case, the termination of the loop of Step 2 of the algorithm must be insured by an additional variable bounding from above the number of iterations ("tries") for a particular pass. Notice that terminating the loop prematurely (before the target density for the pass is provably achieved), does not affect the convergence properties of the algorithm in the limit. To see this, recall that
the algorithm always generates Cauchy sequences which are convergent. In the limit (i.e. given unlimited resources), every Cauchy sequence will be identified and its latest selected element approximated to an arbitrarily small $\epsilon > 0$, even if some intermediate members in some sequences are skipped.

The MB selection algorithm presented above can also be a basis for another view on test selection from infinite sets. Recall that the metric spaces $(S, dt)$ have a finite diameter. The proof of Theorem 3.2 gives a method for computing the elements of the set $S$, that cannot be further to already selected test cases than the current target value of $\epsilon$. Given a finite representation of possibly infinite test sets $G$, this result may be used in the Step 2.1. of the algorithm, to discard all but finitely many test cases that are relevant to the current pass of the algorithm.

### 3-4.2 Test Selection as a Convergent Approximating Process

In this section, we present and prove some interesting properties of the MB selection method.

**Proposition 3.2** Let $\epsilon$ denote the target density (distance) of the test selection algorithm, and assume the cost threshold is unlimited, i.e. cost is not a constraint. If the input set, $G$, of test cases for this algorithm is $\epsilon_G$-dense in the original specification, then the algorithm will produce an $\epsilon_{\text{sel}}$-dense cover of the original specification, where

$$
\epsilon_{\text{sel}} = \epsilon_G + \epsilon .
$$

**Proof.** The proof follows immediately from the specification of the MB selection algorithm and the triangular inequality property of the distance $dt$. 
Consequence 3.1 The algorithm is able to select a test suite $T_S$ whose density ($\epsilon_{sel}$) in the original specification can be made arbitrarily close to the density ($\epsilon_G$) of the input (generated) set $G$ of test cases with respect to the original specification. (For example, if $\epsilon = 1/n\epsilon_G$, then $\epsilon_{sel} = \epsilon_G + 1/n\epsilon_G$, and so, $\epsilon_{sel} \to \epsilon_G$ as $n \to \infty$).

Theorem 3.6 Every (infinite) sequence in the space of execution sequences of a protocol is a limiting sequence of some Cauchy sequence in the same space, when this space is viewed as a metric space $(S, dt)$.

Proof. Let $t = \{(a_k, \alpha_k)\}_{k=1}^{\infty}$ be an infinite sequence. For $n \in \mathbb{N}$, by $t_n^*$ we denote the finite sequence $\{(a_k, \alpha_k)\}_{k=1}^{n}$. This sequence is a Cauchy sequence (by the $dt$ distance definition), and is convergent (by the Completeness Theorem). Moreover, we prove that the sequence $\{t_n^*\}_{n=1}^{\infty}$ converges to $t$. In fact, we have $dt(t_n, t) = \sum_{k=n+1}^{\infty} p_k$. Since the series $\sum_{k=1}^{\infty} p_k$ converges, for every $\epsilon > 0$, $\exists N_\epsilon$ such that $\sum_{k=n+1}^{\infty} p_k < \epsilon$ for all $n > N_\epsilon$. Thus $\forall \epsilon > 0$, $\exists N_\epsilon$ such that $dt(t_n, t) < \epsilon$ for all $n > N_\epsilon$. This proves that $\{t_n^*\}$ converges to $t$.

These results imply that the MB test selection algorithm is a convergent approximation process such that the more test cases selected (through successive passes), the closer the selected set comes to the original set, with no relevant test cases or regions of test cases being missed out. The last theorem implies that in the approximation process, the selected set may contain only finite test sequences even though the original set may contain infinite sequences, and still, the previously established approximation properties will hold. It is interesting that we have arrived to the same result in the metric based theory, as is observed in [27]. There, Hennessy observes (without proving) that one can show that, although infinite tests are allowed (by the theory of testing based on observation), finite tests suffice: if two processes are distinguishable, they are distinguishable using a finite test.
As observed in [39], an exhaustive tester like a canonical tester ([2]), can be viewed as a theoretical upper bound on the testing process for communication protocols. The trace set of the canonical tester is equal to the trace set of the protocol specification it tests. The selection algorithm based on the metric $dt$, defined on sequences of atomic protocol events, can be seen as a convergent approximating process for this theoretical upper bound. The degree of approximation of such a theoretical upper bound is only limited by the amount of testing resources available. Given adequate resources, it will approximate every (finite and infinite) sequence in the starting set to an arbitrarily small $\epsilon > 0$. The exact mode of approximation depends on the characteristics of the metric parameters selected by a test designer, but the convergence always exists if the constraints of the metric definition are satisfied.

### 3-4.3 Mixed Level Hierarchical Test Selection

The results in the previous section show that the MB test selection method is suitable for approximations of theoretical upper bounds of testing processes derived from formal test relations based on atomic event granularity. In this section we show that the MB method can also be used for a more general case of mixed level hierarchical test selection.

Large (and infinite) systems are specified using hierarchical descriptions of their behaviour. LOTOS, for instance, describes distributed systems in terms of processes as follows:

- a system is always described by a process
- a process can be defined in terms of (interacting) subprocesses.

A recursion mechanism allows for the definition of large (and infinite) processes.

Given such a hierarchical system description, corresponding fault models may be established using these different levels of abstraction. As pointed out in [56], it may be that the consideration of hierarchical system description and corresponding hierarchical fault
models is essential for dealing with fault diagnosis for more complex systems.

Since the metric $d_t$ is defined in terms of behaviour graph only, specifically the sequences of interactions derived from it, recursion is not limited to repetition of a single event, but of a subsequence, or even a set of subsequences. The following set of results gives the theoretical foundation for mixed level hierarchical test selection using the MB method.

Let $S$ be a set of execution sequences of atomic events. Let $S_s = \{s_1, \ldots, s_n\}$, $s_i = \{(c_i, \gamma_i)\}_{i=1}^{L}$, $(n, L, \gamma_i \in \mathbb{N})$, be a finite set of finite subsequences of elements of $S$. Then each $t = \{(t_i, \beta_i)\}_{i=1}^{l} \in S$ can be written, in the obvious way, as a concatenation of substrings of $t$ and elements of $S_s$ (substrings of $t$ may be empty):

$$t = t' = s_{i_0} \ldots s_{i_{k_0}} t_{\text{length}(s_{i_0})} + \ldots + t_{\text{length}(s_{i_{k_0}})+m_0} \ldots t_{\text{length}(s_{i_{k_0}})+\ldots+m_l} s_{i_{k_0}+1} \ldots$$

$$\ldots t_{m_1} s_{i_1} \ldots s_{i_{k_0}}$$

for $s_{i_k} \in S_s$ or $s_{i_k} = \varepsilon$ (empty sequence), for all $k = k_0, \ldots, k_m$.

The length of a sequence is taken, as usual, to be the number of its elements (in concise notation, the number of pairs $(c_i, \gamma_i)$). Denote by $t_{(S_s)}$ the sequence $\xi(t')$ obtained by substituting, in the sequence $t'$, all occurrences of nonempty sequences $s_{i_k}$ by their identifier in $S_s$\(^7\), in a concise representation.

**Example 3.1** Consider the following example.

$A := b; B$

$B := c; C$

$C := a; A[d; D$

$D := \text{stop}$

and $S_s = \{s_1\}, s_1 = \{(b, 1)(c, 1)(a, 1)\}$

Then for some sequence $t = \{(b, 1)(c, 1)(a, 1)(b, 1)(c, 1)(a, 1)(b, 1)(c, 1)(d, 1)\}$

---

\(^7\)The identification of substrings is suboptimal in case of sequential composition of an event $a$ and a process $P$, where $P$ can also evolve first by the same event $a$. 
Both sequences \( t \) and \( t_{(S_t)} \) obviously describe the same behaviour given in terms of temporal ordering of atomic events in an execution sequence. From now on, we shall refer to them as the \( t \) and the \( t_{(S_t)} \) representations of that behaviour.

**Theorem 3.7** Let \( (S, dt) \) be a metric space. Let \( S \) be a finite set of finite subsequences of execution sequences \( t = \{(b_i, \beta_i)\}_{i=1}^{n} \in S \). Let \( S_{(S_t)} = \{t(S_t)|t \in S\}, t_{(S_t)} = \{(a_i, \alpha_i)\}_{i=1}^{n} \). Then

1. \( (S_{(S_t)}, dt) \) is a totally bounded metric space

2. the metric space \( (\hat{S}(S_t), dt) \), obtained by completing the set \( S_{(S_t)} \) with ideal elements in the manner described in Section 3-2.4, is complete

3. the metric spaces \( (S, dt) \) and \( (S_{(S_t)}, dt) \) have the following characteristic: if a sequence \( \{t_k\}_{k=1}^{\infty} \) is a Cauchy sequence in the space \( (S, dt) \), then the sequence \( \{t_{(S_t)}_k\}_{k=1}^{\infty} \), where \( t_{(S_t)}_k = t_k(S_t), \forall k \), is a Cauchy sequence in the space \( (S_{(S_t)}, dt) \) and vice versa.

**Proof.** The proof of 1. and 2. is straightforward after observing that the properties of metric spaces of strings of identifiers with the metric \( dt \) are independent of the granularity of events in execution sequences.

The proof of 3. follows from the following observations.

Case 1: Let each sequence \( t_{(S_t)}_k \) be obtained by substituting at most an initially determined finite number \( r \) of occurrences of strings from \( S \) in the corresponding sequences \( t_k \).

Differing in, at most, a fixed finite number of occurrences of finite substrings within their individual elements, does not affect the convergence properties of either Cauchy sequence in the limit. Therefore, if \( t \)-representations of execution sequences form a Cauchy sequence
in the space \((S, dt)\), their \(t_{(S_s)}\)-representations also form a Cauchy sequence in the space \((S_{(S_s)}, dt)\), and vice versa.

Case 2: Let the sequences \(t_{(S_s)} = ((a_i^k, \alpha_i^k))_{i=1}^{K_k}\) be obtained by substituting an arbitrary number of occurrences of strings from \(S_s\) in sequences \(t_k = ((b_i^k, \beta_i^k))_{i=1}^{L_k}\).

Case (i). In this case, sequences \(t_{(S_s)}\) may also contain elements of the form \((a_i^k, \alpha_i^k)\), where each \(a_i^k = s_j \in S_s\), and such that \(\alpha_i^k = m\), for some arbitrary \(m \in \mathbb{N}\). Let one such element be \((a_{i_0}^k, \alpha_{i_0}^k)\). It follows that in sequences \(t_k\) there exist subsequences with arbitrary numbers of repetitions of the sequence \(a_{i_0}^k = s_i \in S_s\). Let \(\{t_k\}_{k=1}^{\infty}\) form a Cauchy sequence. From the property of Cauchy sequences in the space \((S, dt)\) (Theorem 3.3. (ii), second part), and the fact that sequences \(t_k\) allow arbitrarily large number of substitutions of (finite) substrings, it follows that the sequences \(t_k\) must be such that their length tends to infinity. Also, by the same theorem, each of the sequences \(\{b_i^k\}_{k=1}^{\infty}\) is eventually constant, and \(\{\beta_i^k\}_{k=1}^{\infty}\) is eventually constant or \(\beta_i^k \to \infty (k \to \infty)\). Infinite number of substitutions cannot be derived from cases where \(\beta_i^k \to \infty\), since the set \(S_s\) is finite.

It follows that, the arbitrary number of substitutions of the string \(s_i\) is obtained, from eventually constant strings \(\{b_i^k\}_{k=1}^{\infty}\), with eventually constant \(\{a_i^k\}_{k=1}^{\infty}\). Therefore, for \(\{a_i^k\}_{k=1}^{\infty}\), in the \(t_{(S_s)}\) representations of \(t_k\) strings, the sequence \(\{\alpha_i^k\}_{k=1}^{\infty}\) is such that \(\alpha_{i_0}^k (= m) \to \infty\), as \(k \to \infty\). By the second property in Theorem 3.3, (ii), the sequence \(\{t_{(S_s)}\}_{k=1}^{\infty}\), is Cauchy.

The reasoning in the opposite direction (i.e. if \(\{t_{(S_s)}\}_{k=1}^{\infty}\) is Cauchy, then \(\{t_k\}_{k=1}^{\infty}\) is also Cauchy), is carried out using the same arguments.

Case (ii). Alternatively, the arbitrary number of substitutions of finite strings \(s_j \in S_s\) in \(t_k\), does not generate elements in \(t_{(S_s)}\) with arbitrary recursion levels. (For instance, every other substring is substituted, in an infinite string.) It is easy to see that, in this case, if either representation is Cauchy in its space, by Theorem 3.3., sequences \(\{b_i^k\}_{k=1}^{\infty}\) and \(\{a_i^k\}_{k=1}^{\infty}\), as well as \(\{\beta_i^k\}_{k=1}^{\infty}\) and \(\{\alpha_i^k\}_{k=1}^{\infty}\), are eventually constant, and lengths of sequences \(t_k\) and \(t_{(S_s)}\) tend to infinity, as \(k \to \infty\). Therefore, by Theorem 3.3., the other one of the sequences (precisely, the other representation) is also Cauchy, in its metric
From the proof of the previous theorem it is obvious that, irrespective of the representation of execution sequences in $S$ (i.e. in $t$ or $t_{(S_s)}$-representations), if one set $T_1$ of execution sequences approximates (in terms of the maximum distance $m(T_1)$) some set $G \subset S$ to an arbitrary $\epsilon_1 > 0$ in its metric space, then there exists such $\epsilon_2 > 0$, and the set $T_2$ (the alternative representation of $T_1$) in another metric space representation, such that it approximates the same set $G$ with $\epsilon_2$. The details of the proof reveal that, this is in particular true of sets $T_1$ and $T_2$ of Cauchy sequences, which are $t$ and $t_{(S_s)}$ representations of each other. The consequence of this observation is that, the MB selection method, which generates Cauchy sequences over its consecutive iterations, really does the approximations in both metric space representations simultaneously: the speed of the approximations of a particular granularity level differs, however, depending on the metric space representation (and metric parameters, as before).

In the introduction to this section, we mentioned that the results of the MB method may also be applied to metric spaces that contain sets of subsequences inside an execution sequence representation. We now formalize this result, which is essentially an extension of the previous theorem.

Let $S$ be a set of execution sequences of atomic events, and $(S,dt)$ a metric space. Let $S = \{S_1, \ldots, S_n\}$, where $S_j = \{s_i|i \in \mathbb{N}\}$, and $s_i = \{(e, i, \gamma_i)\}_{i=1}^{L}$, $(n, L, \gamma_i \in \mathbb{N})$, be a family of a finite number of sets of subsequences of elements of $S$, such that all sets and subsequences are finite. Then each $t = \{(t_i, \beta_i)\}_{i=1}^{L} \in S$ can be written, in the way previously detailed for the set $S_s$, as a concatenation of substrings of $t$ and elements of $S_i \in S$.

Denote by $t_{(S)}$ the sequence obtained by substituting, in a sequence $t$, all occurrences of nonempty sequences $s_{i_k} \in S_j \in S$ by the identifier of the set $S_j$ (i.e. the set they belong to in the family of sets), in a concise representation.
Theorem 3.8 Let \((S, dt)\) be a metric space. Let \(S\) be a finite family of sets of subsequences of execution sequences of \(S\), and all sequences and sets in the family are finite. Let \(S(S) = \{t(s) | t \in S\}\). Then

1. \((S(S), dt)\) is a totally bounded metric space

2. the metric space \((\hat{S}(S), dt)\), obtained by completing the set \(S(S)\) with ideal elements in the manner described in Section 3-2.4, is complete

3. if all \(S_i \in S\) are disjunct, then every sequence that is Cauchy in \((S, dt)\) is also a Cauchy sequence in the space \((S(S), dt)\).

Proof. The proof of 1. and 2. is the same as in the previous theorem. Also, the proof for 3. follows directly from the proof for 3. in the Theorem 3.7, the only difference being purely syntactic with respect to identifiers (of strings in Theorem 3.7, and sets in this theorem).

Notice, however, that the opposite implication in Theorem 3.8, 3., i.e. if a sequence is Cauchy in \((S(S), dt)\), then it is Cauchy in \((S, dt)\), does not necessarily hold. (It only holds in cases when this statement reduces to Theorem 3.7, 3., i.e. only one string in a set is used for substitution). To see this consider the following example.

Example 3.2 Let \(S = \{S_1\}, S_1 = \{s_1, s_2\}, s_1 = \{(a,1), (b,1)\}\) and \(s_2 = \{(b,1), (a,1)\}\). Let \(s\) be the identifier for set \(S_1\).

Let a sequence \(\{t_k\}_{k=1}^\infty\) be an infinite sequence in \((S, dt)\), where

\[ t_k = \{(a,1)(b,1)(a,1)(b,1)\ldots\}, k = 1, 3, 5, \ldots \]
\[ t_k = \{(b,1)(a,1)(b,1)(a,1)\ldots\}, k = 2, 4, 6, \ldots \]

Then \(t_k(s) = \{(s, \alpha)\}, \alpha = \infty, \forall k = 1, 2, \ldots \).
3-4. **METRIC BASED TEST SELECTION**

The sequence \( \{t(s)\}_{k=1}^{\infty} \), is obviously Cauchy (with the distance between any two sequences equal to 0), whereas

\[ dt(t_k, t_{k+1}) = p, \forall k = 1, 2, \ldots \text{ (p is the sum of the series } p_k). \]

The last theorem implies that the convergence properties of strings in the \((S(s), dt)\) space are strictly tied to their new syntactic representation and consequently, nothing can be said about their convergence properties with respect to the space \((S, dt)\).

The final results in this chapter enable us to use the same MB-method and selection algorithm for mixed level hierarchical test selection and coverage evaluation. If event subsequences of execution sequences, derived from behaviour graphs, relevant to testing can be identified, then the selection of \(\varepsilon\)-dense sets can be done relative to "direct" recursion, "mutual" recursion, or even at the level of individual processes or groups of processes\(^8\). Additionally, Theorem 3.8 and the characteristics of the MB method are essential for treatment of theoretical upper bounds on the testing process, which are essentially, but not entirely based on the atomic event temporal orderings. Whereas the relevance of the first observation is evident in testing, the latter point will be further detailed in Chapter 5.

---

\(^8\)For conciseness, we here use the terminology mainly inherited from process algebras.
Chapter 4

A Metric Hierarchy of Fault Coverage Models

In this chapter we apply the metric theory of the protocol execution space to investigate and classify the relative power of some well understood test selection methods in protocol testing. These selection methods include transition and transfer fault checking experiments for classes of Finite State Machines [52], and control flow analysis [49, 55]. These methods often involve exact evaluations of test selection criteria, and may need to impose limiting assumptions about the recursion and parallelism in protocol systems in order to be computable for very large or infinite systems. We show how these test selection methods can be metrically characterized in their (finite) domains. The same metric based methodology naturally extends from finite to infinite domains to encompass the full recursion and parallelism of real protocol implementations. This extension then characterizes all-paths infinite software test coverage criteria and trace equivalence for labelled transition systems. This theoretically well-founded extension is supported by the proof of the existence of finite covers for infinite systems as well as the systematic convergence of the metric based cover hierarchy for infinite systems. Finally, we put all the results in
a common theoretical setting of trace sets of protocols, to show how the hierarchy of test covers formed by metric based theory can be related to previously investigated fault coverage models for certain coverage classes, and approximates the theoretical upper bound of trace equivalence.

4-1. SOME GENERAL RESULTS

The characterization of coverage of protocol test methods in metric terms is investigated using two related concepts from the general theory of metric spaces: the concept of $\varepsilon$-dense sets in metric spaces (Definition C.5), and Hausdorff distance (Definition C.8).

Let $(M, dt)$ be a metric space and $T, S$ be two subsets of that metric space. Informally, $T$ is $\varepsilon$-dense in $S$ if the family of all open spheres with centers at the points in $T$ and with radius $\varepsilon$ forms a covering for $S$.

**Remark 4.1** Let the selected set of test sequences, $T$, be $\varepsilon$-dense in $S$. Then the coverage $CovMax(T)$ can be related to the achieved density $\varepsilon$, by observing that

$$\sup\{dt(s, T) : s \in S \setminus T\} \leq \varepsilon$$

Therefore, according to the definition of $m(T)$, (Definition 3.6),

$$m(T) = \frac{\sup\{dt(s, T) : s \in S \setminus T\}}{\sum_{k=1}^{\infty} p_k} \leq \frac{\varepsilon}{\sum_{k=1}^{\infty} p_k}$$

Thus, (definition 3.7),

$$CovMax(T) = 1 - m(T) \geq 1 - \frac{\varepsilon}{\sum_{k=1}^{\infty} p_k}$$

where $p_k$ is the appropriate sequence for the distance $dt$ used in the selection process.
Therefore, we will often use $\epsilon$-density to characterize the coverage of a selected test set, whenever the sequence $\{p_k\}_{k=1}^{\infty}$ is fixed or normalization is not relevant.

We also remark that the Hausdorff distance, takes on a particular meaning in the environment of sets of tests and their selected subsets.

**Remark 4.2** Let $T$ represent a selected set of test sequences and $S$ represent the set from which the selection is done. Recall that the Hausdorff distance is assigned the greater of the two distances,

$$dt_H(S, T) = \max\{\sup\{dt(s, T) : s \in S\}, \sup\{dt(t, S) : t \in T\}\}$$

Since $T$ is a subset of $S$, then the second supremum in the definition of $dt_H$ need not be evaluated (it is always 0). Therefore if $dt_H(S, T) = \epsilon$, it follows that $T$ is $\epsilon$ dense in $S$. Notice also that if the Hausdorff distance between a test set $T$ and its superset $S$ is equal to zero (or $\epsilon$-density of $T$ in $S$ approaches zero), the $CovMax(T)$ becomes 1 (approaches 1).

We quote some general observations which hold in metric spaces of sequences of observable actions characterized by the metric function $dt$.

In the following propositions, let $T$ and $S$ denote subsets of a metric space $(M, dt)$, where $M$ is a (finite or infinite) set of execution sequences of a protocol behaviour. Furthermore, let $t = (a_k, \alpha_k)$ be an element in $T$. Likewise, let $s = (b_k, \beta_k)$ be an element of the set $S$. We will also take that the sequence $p_k$ is a monotonously decreasing sequence, $p_{m+1} < p_m$, for all $m$. Similarly, let the sequence $r_j$ in the definition of $dt$ be such that $r_j - r_{j-1}$ is decreasing for $j = 2, 3, \ldots$. These assumptions greatly simplify the subsequent analysis.

**Proposition 4.1** If $dt_H(T, S) < p_k$ for some $0 \leq k < \infty$, then for each $s \in S$ there exists
4-2. ANALYTICAL COMPARISON OF TEST SELECTION METHODS

$t \in T$ such that $a_j = b_j$, $j = 1, 2, \ldots, k$.

**Proof.** From $dt_H(S, T) < p_k$ it follows that $dt(s, T) < p_k$. Consequently, there exists $t \in T$ such that $dt(s, t) < p_k$. If for some $j \in \{1, 2, \ldots, k\}$, $a_j \neq b_j$ then $dt(s, t) > p_j$. Therefore we must have $a_j = b_j$ for all $j = 1, 2, \ldots, k$.

**Proposition 4.2** If $dt_H(T, S) < \rho_k p_k$ for some $0 < k < \infty$, $0 \leq \rho_k < 1$, then for each $s \in S$ there exists $t \in T$ such that $a_j = b_j$ and $|r_{\alpha_j} - r_{\beta_j}| < \rho_k$, $j = 1, 2, \ldots, k$.

**Proof.** From $dt_H(T, S) < \rho_k p_k$ it follows that $dt(s, T) < \rho_k p_k$. Consequently, there exists $t \in T$ such that $dt(s, t) < \rho_k p_k$. If for some $j \in \{1, 2, \ldots, k\}$, $a_j \neq b_j$ or $|r_{\alpha_j} - r_{\beta_j}| \geq \rho_k$, then $dt(s, t) > \rho_k p_j > \rho_k p_k$. Therefore, $a_j = b_j$ and $|r_{\alpha_j} - r_{\beta_j}| < \rho_k$, for all $j = 1, 2, \ldots, k$.

4-2 Analytical Comparison of Test Selection Methods

In this section, we provide an analytical characterization of the error classes that can be detected by a hierarchy of $\epsilon$-dense test covers in metric spaces of execution sequences of the protocol control behaviour. These covers do not involve automata-theoretic or other software fault-model concepts and are not proposed as an alternative to such methods. Our purpose is to investigate the effect of the increasing coverage $\text{CovMax}(T)$ of a selected test set on the error detection capability in software designs for protocols within a unified framework of infinite metric systems.

The environment of execution sequences enables us to investigate the behaviour of protocol systems in a unified framework for all (formal) description techniques that allow derivation of execution sequences in which the systems they specify may engage,
and without regard to the atomicity of their interactions (see also Section 3-4.3). The non-exhaustive list includes traces [17] for LOTOS and other LTS-based specification techniques, with the labelset of atomic actions or events, paths of state transition diagrams corresponding to Finite State Machines (FSM) labelled with transitions [29], derivations and computations [7] for process algebras with labelset of (action,state) pairs, etc. We use this fact to our advantage, so that in each subsequent section, for each test method and corresponding formal description model that the method is based on, we redefine the execution sequence to consist of (not necessarily atomic) actions that are derivable within the model.

4-2.1 A Metric Characterization of FSM-based Methods

Finite State Machine based methods for detecting errors in protocol implementations are a major research area in protocol testing. We first turn our attention to the metric characterization of some basic fault models for finite state machines descriptions of protocols. This characterization is given in terms of the increasing coverage $CovMax(T)$ of selected test set $T$ with respect to $S$, in the metric space of paths generated by the state transition diagram that corresponds to the specification FSM (see [29] for details).

A FSM is defined as a 5-tuple $(S, \Sigma, s_0, \delta, F)$ where $S$ is a finite set of states, $\Sigma$ is a finite set of input/output operations, $s_0 \in S$ is the initial state, and $\delta : S \times \Sigma \rightarrow S$ is a state transition function, and $F$ is a set of final states. In the simplest case, one assumes a completely specified FSM. The type of basic faults considered in this model are the following [56].

*Transition errors:* A machine is said to have a transition error, if for a given state and input pair it produces a different output than the specification machine it is compared to.
Transfer errors: A machine is said to have a transfer error, if, for the corresponding state and input received, the machine enters a different state than the specification machine it is compared to.

We assume a minimal, completely specified, deterministic FSM considered in protocol checking experiments [52]. As noted in [52], interaction paths recorded by the state space exploration procedure can be used to generate conformance test sequences. We use a similar methodology here for analysis. To every transition we assign a distinct label \( t \in L \), where \( L \) is a finite labelset of transitions. We build a reachability tree, labelling the edges with transitions. This reachability tree obviously decomposes into distinct paths originating at the root of the tree (state \( s_0 \)), and traversing over the nodes (states in \( S \)) and edges (transitions in \( L \)) to the leaves of the tree. For uniformity of discussion, we call these paths execution sequences. These paths are uniquely labelled by their edges. We use the simple technique outlined in the previous chapter, to represent these execution sequences concisely (Definition 3.1).

We shall designate the metric space built on top of the set of all execution sequences of a protocol FSM, supplied with metric \( dt \), as \( (\text{ExecSeqFsm}, dt) \).

Let \( N_T, N_S \) be the minimum levels of the reachability tree corresponding to a FSM specification, by which all transitions, respectively states, have been traversed by an interaction path. (We take that the level of the root node \( s_0 \) within the reachability tree is zero.)

Theorem 4.1 (Metric characterization of transition faults)

Let \( (\text{ExecSeqFsm}, dt) \) be a metric space. If

\[ \epsilon_{TF} < p_{NT} \]

then every \( \epsilon_{TF} \)-dense test suite in \( (\text{ExecSeqFsm}, dt) \) has the same fault detection power.
CHAPTER 4. A METRIC HIERARCHY OF FAULT COVERAGE MODELS

with respect to transition faults as transition checking experiments on FSM-s (i.e. T-method [52]).

Proof. The proof of this theorem follows directly from Proposition 4.1.

We illustrate the generality of this approach in Figure 4.1 where a T-tour is generated using the knowledge of $N_T$ and metric only. (The metric based method is of course not as efficient for selection T-tours as other methods specially designed to deal with it.) Note also that from now on we always assume that generated test sequences will be appropriately mapped to input events only and supplied with other intermediate subsequences (such as reset sequence) in order the form the exact sequence required by a particular method.

One $\varepsilon$-dense set is \( \{ t_0, t_2, t_3, t_0 \} \)

\( \varepsilon < 1/4 \quad p_k = 1/2^{k-1} \quad r_k = 1 - 1/2^k \)

Generating a T-tour using metrics only

Figure 4.1

Let the length of the longest state signature (UIO or distinguishing sequence [29]) for a particular FSM be \( uiol \).
Theorem 4.2 (Metric characterization of transfer faults)

Let \((\text{ExecSeq}_{FSM}, dt)\) be a metric space. If

\[
\epsilon_{SF} < (r_{N_S+1+uiol} - r_{N_S+uiol})PN_{S+1+uiol}
\]

then every \(\epsilon_{SF}\) dense test suite in \((\text{ExecSeq}_{FSM}, dt)\) has the same fault detection power with respect to transfer faults as transfer checking experiments on FSM-s using state signatures (e.g. U-method, D-method [52]).

Proof. The proof of this theorem is straightforward after observing that \(r_{N_S+1+uiol} - r_{N_S+uiol}\) is the smallest value that can appear as the factor \(\delta_k\) in the evaluation of the distance between any two sequences, whose length is bounded from above by the value \(N_S+1+uiol\). Proof for this theorem then follows directly from the definitions of methods and Propositions 4.1 and 4.2.

Consider an application of metric \(dt\) in the above setting.

Consequence 4.1 (A metric characterization of UIO sequences)

Consider a protocol finite state machine with \(n+1\) states, and designate with \(sp_0, sp_1, \ldots, sp_n\) the shortest paths to those states [52]. We construct a reachability tree for such a FSM so that the root of the tree is \(s_0 \in S\), and edges originating at \(s_0\) leading to \(s_0, s_1, \ldots, s_n\) are labelled \(sp_0, sp_1, \ldots, sp_n\) respectively. The reachability tree then can further be constructed as previously described, and the generated set of all execution sequences originating at \(s_0\) we denote as \(\text{ExecSeq}_{FSM_{sp}}\). Consider the metric space \((\text{ExecSeq}_{FSM_{sp}}, dt)\).

Then any suffix \(s_{uiol}\) of an execution sequence \(s = sp_i + s_{uiol} \in \text{ExecSeq}_{FSM_{sp}},\) (where \(sp_i + s_{uiol}\) denotes the concatenation of sequences \(sp_i\) and \(s_{uiol}\)), is a UIO sequence for state \(s_i\) if and only if
CHAPTER 4. A METRIC HIERARCHY OF FAULT COVERAGE MODELS

\[ dt(s, \text{ExecSeq}_{FSM}, \{t \mid t = sp_t_{rest}) > p_1. \]

**Proof.** The proof of this consequence follows directly from the observation that the tail \( s_{uiol} \) of any sequence starting at a node \( s_i \) will differ from any sequence starting at any other node \( s_j \) if and only if the distance between such two sequences is greater than \( p_1 \).

Similar to [6, 52], we formulate a result relating the fault detection power of transition and transfer checking experiments for a FSM satisfying the basic assumptions as stated earlier.

**Consequence 4.2** Every \( \epsilon_{SF} \)-dense test suite in the space \((\text{ExecSeq}_{FSM}, dt)\), is also \( \epsilon_{TF} \)-dense in it. The opposite is not true, i.e. there always exist \( \epsilon_{TF} \)-dense test suites which are not \( \epsilon_{SF} \)-dense in the same space.

**Proof.** The proof is based on the Theorems 4.1 and 4.2, as well as the observation that \( N_S + 1 \geq N_T \) and \( uiol > 1 \), under the assumptions about FSMs that we adopted.

In other words, transfer cover is always a transition cover. Notice that the opposite does not hold.

Finally, we formulate a general theorem relating the fault detection power of a test suite selected by the MB method with respect to transition and transfer faults of specification FSMs.

**Theorem 4.3** *(Minimal fault detection power of an \( \epsilon \)-dense test suite)*

Let \( N_S \) and \( uiol \) be defined as before.

Then every \( \epsilon \)-dense test suite in \((\text{ExecSeq}_{FSM}, dt)\), where

\[
N_{T'} = \max\{k \mid \epsilon < p_k\} \\
N_{S'} = \max\{k \mid \epsilon < (r_{N_S+1+uiol} - r_{N_S+uiol})p_k\}
\]
4-2. ANALYTICAL COMPARISON OF TEST SELECTION METHODS

can uncover all transition faults of transitions which appear up to level $N_T$ in the reachability tree, and all transfer faults for states that appear up to level $N_s - 1 - \text{viol}$, for a completely specified, minimal and deterministic FSM.

Proof. The proof for this theorem follows directly from Theorems 4.1, 4.2 and Proposition 4.2.

4-2.2 A Metric Characterization of Some Software Testing Covers

In this subsection we examine some general software testing covers and selection strategies which have been applied in protocol testing for identifying sequencing errors [56]. As is customary in general software testing, we take that the protocol designs are represented by graphs. We label the edges of such graphs with observable protocol actions. For further details and general applicability of the method see [49, 55].

We first present the brief definitions of basic software test covers [6] and then discuss the appropriate metric coverage hierarchy which would result in those covers.

Branch Cover: The number of executable paths in a design with loops may be infinite. A common practice is to require that the test sequences form a "branch cover", i.e. every branch in the design is traversed by at least one test sequence.

Path Cover: A path cover consists of test sequences such that every path in the design graph is traversed by some test sequence.

In the following propositions, let $T$ and $S$ denote subsets of a metric space $(M, dt)$ generated by the paths of the protocol design graph and metric $dt$. We omit most of the proofs as they are simple extensions of the above results.
Chaprer 4. A Metric Hierarchy of Fault Coverage Models

Proposition 4.3 If

\[ dt_H(T, S) < \epsilon_{BF} < p_{k_t} \text{ for some } 0 \leq k_t < \infty, T \subset S \]

then the set \( T \) is a branch cover for the branches up to the level \( k = k_t \) for the design graph of \( S \).

Consider (as in [6]) that the design graph is generated by a FSM, and label the edges of the graph with transitions. Then, if \( k_t = N_T \), all branches in the design graph have been traversed at least once, by sequences in any \( \epsilon_{BF} \)-dense sets \( T \) in \( S \).

We can therefore formulate the following result.

Consequence 4.3 For branches in the design graph labelled with transitions of a minimal, deterministic and fully specified FSM, for every \( \epsilon_{TF} \)-dense set \( T \) in \( S \), there is an \( \epsilon_{BF} \)-dense set \( T' \) in \( S \) that is arbitrarily dense in it. The opposite is also true.

We immediately obtain the same result as in [6], where automata theoretic methods were used to derive this result.

Consequence 4.4 A branch cover can detect all transition errors and not all transfer errors.

Proof. The proof follows directly from Consequence 4.2 and Consequence 4.3.

If the generation of the set \( S \) is constrained by the number of times a loop in the protocol design graph is entered, (which is the usual technique for dealing with design graphs with loops), then the set \( S \) is closed. In that case the following proposition holds.

Proposition 4.4 Let the set \( S \) be closed. If
4-3. APPROXIMATING INFINITE SYSTEMS

\[ dt_H(T, S) = 0 \]

then \( T \) forms a path cover for \( S \).

Various sequencing coverages of the types characterized above have been used in protocol testing, improved by combining them with data flow analysis [49, 55].

4-3 Approximating Infinite Systems

In this section we extend previous results by comparing the methods of test selection and coverage for real protocol implementations which are specified as infinite systems. The techniques described in Section 4.2 assume a finite fault model which permits them to use exact selection techniques, fault targeting and subsume relationships for coverage. Exact selection techniques and finite fault models become impractical in infinite execution spaces generated by discontinuous and nondeterministic behaviour. In this section we adopt trace equivalence to characterize an infinite fault model: every string is a possible fault.

It turns out that the effect of increasing coverage (decreasing \( \epsilon \)), on the fault detection power of selected tests carries over for from finite to infinite systems, but taking on a slightly different meaning. Exploiting the link provided by the metric based framework, we summarize the main theoretical results within the framework provided by trace equivalences of labelled transition systems.

4-3.1 A Metric Characterization of Trace Equivalence

In the analysis that follows, we will change the theoretical setting to the most general one, in order to compare techniques that differ in control space representation and test characterization. First, we assume that the specification formalism that is used has a
particular class of language expressions for the description of system behaviour. The operational semantics of labelled transition systems and some formal specification techniques for protocols enable us to derive sequences of actions, in which a communication system can engage, from the structure of the expression describing the system behaviour itself (for details see [31]). Sequences of actions are also the fundamental control behaviour description element, that can be derived from a number of other specification formalisms.

From this point on, we will carry investigation in metric spaces of all execution sequences for any specification formalism that allows their derivation. For the particular case of formal description techniques that allow specifications of concurrent systems and have a finite labelset of actions, we will carry the investigation for trace sets derived by interleaving semantics of concurrency.

We view test selection process as an approximating process within infinite control behaviour spaces and base the approximation on the $dt$ metric function.

The $dt$ metric function provides a measure of how well one execution sequence approximates another. An initial approximation of this infinite metric space can be achieved through an $\epsilon$-dense topological cover as demonstrated in Section 4.2. Such an approximation would have several benefits: similar behaviour patterns would be grouped and tested as a unit, the degree of approximation is easily controlled through the $\epsilon$ parameter and parameters of the metric $dt$, and, intuitively, classes of similar faults will expose themselves in similar situations of event pattern and recursion.

The main concern of the proposed approximation technique must be the selection of a finite set of test cases according to the selection criteria. Theorem 3.2 proves that the metric space is totally bounded which implies a finite $\epsilon$-dense cover for all values of $\epsilon$. We are therefore justified to turn our attention to characterizing any (finite or infinite) theoretical upper bound of approximations by $\epsilon$-dense covers, as $\epsilon$ tends to 0.

Let $T$ and $S$ be two subsets of a metric space $(E^*, dt)$, where $E$ is the labelset of
observable protocol actions. The following propositions are reformulations of results in the general theory of metric spaces, given an interpretation in the setting of control execution spaces of protocols and their tests.

**Proposition 4.5** If $S$ is closed then the following relation holds:

$$\sup \{dt(t, S) : t \in T\} = 0 \Rightarrow T \subseteq S$$

Let $B_1$ and $B_2$ be two protocol processes and $T = Tr(B_1)$ and $S = Tr(B_2)$ the sets of their traces. Then the above proposition gives metric characterization of the trace preorder between processes $B_1$ and $B_2$.

**Proposition 4.6** Let $T$ and $S$ be any subsets of the same metric space. If

$$dt_H(T, S) = 0$$

then the sets of all truncations of sequences in $T$ and $S$ are the same.

Again, let $B_1$ and $B_2$ be two protocol processes and $T = Tr(B_1)$ and $S = Tr(B_2)$ the sets of their traces. Then the above proposition characterizes the concept of trace equivalence between processes $B_1$ and $B_2$.

Note that, in either case, the set $T$ can be viewed as a set of selected tests for a specification of a protocol system whose trace set is $S$. Then the above two propositions characterize the inclusion relation and the level of approximation of a specification by a set of tests with respect to trace equivalence.

The above theorems characterize trace preorder and trace equivalence in terms of the metric function $dt$. As $\epsilon$ tends to 0 in $\epsilon$-covers (and so test coverage $CovMax(T)$
tends to 1), these characterizations imply that the covers approximate precisely the trace equivalence, thus providing the theoretical foundation for metric based theory of test selection and coverage. This last property is then used in a later section to build a metric hierarchy of protocol testing.

4-3.2 A Metric Hierarchy of Testing

In Section 4.2, we examined different protocol test selection methods in the theoretical settings they are designed for. We now show that similar results hold if these methods are viewed in infinite metric spaces of traces of protocol specifications.

Let \( Tr(S) \) be a trace set of a protocol specification. Let the set of observable protocol actions \( E \) be the union of two sets which we call \( I \) (input) and \( O \) (output) events, \( E = I \cup O \). For the results of Section 4.2 to hold, we require that the sets \( I \) and \( O \) of input and output actions of a protocol be disjoint. (This should not be too restrictive in case of protocols). For \( NT, NS \) and \( uiol \) defined as in Section 4.2, define

\[
NS_{tr} = 2NS \\
NT_{tr} = 2NT \\
uiol_{tr} = 2uiol
\]

Consider the metric space \((Tr(S), dt)\) and let

\[
\epsilon_T < pNT_{tr} \\
\epsilon_S < (rNS_{tr} + 2uiol_{tr} - rNS_{tr} + uiol_{tr})pNS_{tr} + 2uiol_{tr}
\]

**Proposition 4.7** Let \( TST \) be any \( \epsilon_T \)-dense test suite in \((Tr(S), dt)\), and let \( TS_S \) be any \( \epsilon_S \)-dense test suite in \((Tr(S), dt)\). Then \( TST \) (\( TS_S \)), when applied to a minimal, completely specified and strongly connected FSM, have the same fault detection power with
respect to transition faults (resp. transfer faults) as $\epsilon_{TF}$ (resp. $\epsilon_{SF}$) dense test covers in $(\text{ExecSeq}_{\text{FSM}}, dt)$.

**Sketch of the proof:** The proof of the above proposition is essentially a direct consequence of Theorems 4.1 and 4.2. However, one needs to notice that, by writing the sequences from the set $Tr(S)$ in the concise notation as is customary with metric $dt$, we run the risk of collapsing together actions which may represent distinct actions in other models. We eliminate such danger in case of overlapping identifiers for the sets $U$ and $I$ by imposing the above artificial restriction. We also implicitly assume that, (and this is the case for models considered), if a FSM performs a transition with null output event, it will remain in the same state. This ensures that considering traces with observable events broken down to atomic actions, and then collapsing them through their recursion, will not result in information loss. If (the maximum of) twice the length of subsequences labelled with transitions is examined in the same manner as before, the equivalent results will be obtained.

A similar result holds for the branch cover (let the reformulation of $\epsilon_{BF}$ be called $\epsilon_{B}$-dense cover) of a design graph which corresponds to a specification FSM in the sense of Proposition 4.3, when it is viewed in the space $(Tr(S), dt)$.

Based on the analytical investigation carried out in previous sections, we are able to establish the metric hierarchy of fault coverage models in protocol testing as illustrated in Figure 4.2. The fault coverage models are placed lower in the hierarchy, if the $\epsilon$-dense cover needed to identify the class of faults identifiable in that model is coarser (i.e. $\epsilon$ is larger). Conversely, the finer distinction in terms of $\epsilon$-dense covers needed, places the models higher in the metric based hierarchy. Arrows, then, point towards fault models for which the detectability of errors is granted by $\epsilon$-dense covers higher in the hierarchy. Note that, for finite models, fault detection capability is strictly related to the appropriate fault model definition and assumptions about the implementations within such models.
In other words, transition errors indicate the fault detection capability for deterministic implementations of finite state machine descriptions of protocols with assumptions as in Section 4.2. Nothing is said about the fault detection capability of such covers for general, infinite and possibly non-deterministic implementations.

The above results show that the metric based theory of test selection and coverage can be viewed as a general theoretical framework for some seemingly unrelated testing theories.

From the practical point of view, we would like to point out that approaches based on pattern identification and pattern similarity evaluation have often been used in areas of physical sciences for problems that defy exact evaluation for reasons of complexity. Similar
problem is found in test selection and coverage evaluation for communication protocols. In the light of above results, the metric based theory of test selection and coverage can be used to reformulate protocol test selection problem as an approximation technique in abstract terms of various measures, for different fault models. These measures are easier for automated manipulation in very large systems than models based on detailed analysis of particular atomic events, states or other specific test selection criteria.
The theory of test selection and coverage requires that two factors be defined: the target theoretical upper bound and the approximation mode. In the development of the metric based theory for test selection and coverage, we have defined the metric approximation that will tend to the theoretical upper bound of strings equivalence between the execution space of the protocol control behaviour and selected test sequences. This approach gives us the flexibility, to define more specific target test criteria if needed, provided they are based on strings equivalence.

In this chapter we extend the formal theory of testing protocol implementations along these lines. In investigating the control space of the protocol behaviour, we have concluded that a more efficient test selection and execution process may be obtained for implementations supporting many network connections, if the target coverage criteria is not strings equivalence, but a conveniently related notion. We propose a new relation, from which the new target test criteria is defined. The corresponding test architecture, formal protocol specification style and tester design are also specified.
First, existing notions of implementation preorders and equivalences in testing theory are examined with respect to their discriminating power for identifying processes which will interoperate. We show that all of the proposed relations are too strict if the interoperability of implementations is desired. Moreover, the practical testers designed to test with these relations as target criteria may be difficult to design and run, typically generating many inconclusive test runs for test cases where these relations impose too strict requirements (see also [2]). In particular, we show that the question of test selection turns out to be impossible to solve adequately within the proposed theories themselves.

Next, we introduce a formal framework for interoperability testing of (two or more) communicating systems. We extend the testing theory based on formal specifications by defining a new \textit{intop} relation which is defined to hold between an implementation and a specification of a system. It insures that implementations which are valid in \textit{intop} sense can interoperate with other such implementations of the same specification. This relation improves the efficacy of the test selection process by distinguishing the traces which are either unexecutable or irrelevant, from those that are essential for basic interoperability, without the need to step outside the theory for an adequate solution. In particular, it is a tighter upper bound to the testing process than the previously defined relations, since the number of inconclusive test runs is significantly reduced.

This result is particularly significant for designing testers for systems which are capable of running independent concurrent processes (i.e., for testing simultaneous network connections). When it comes to testing modern network protocols such as ST-II (multimedia) communication protocol [15], some features emerge (shown on the example ST-II specification throughout the chapter), that were not critical in testing classical protocols usually considered in the protocol test theory. For instance, even some basic functionality of an ST-II agent cannot be tested without testing at least three simultaneous connections. This then poses a problem in protocol test design (which typically deals with one connection or linear test sequences), which we feel is best solved by redefining the ba-
sic test relations to better suit the design of modern protocols. At the same time, the new testing framework greatly facilitates test development and tester design for multiple simultaneous network connections, paving the way to meeting stricter reliability requirements for tested communication systems.

5-1 Implementation relations and interoperability

The goal of this chapter is to develop a general mathematical framework for reasoning about the interoperability of communicating systems. Interoperability is a pivotal notion within Open Distributed Processing (ODP) concept in general, and network protocols in particular. If we assume that a (formal) abstract specification of a communicating system is available, then the interoperability of its different implementations is dependent on the design decisions taken during the implementation process leading from the specification to an executable implementation. Viewed this way, the question of interoperability of communicating systems is closely related to the formalization of the notion of validity, i.e., the nature of the relation which should hold between the implementation of a system and its (formal) specification.

This formal relation has been a subject of extensive research (especially within process algebraic techniques), and the results can broadly be summarized in two categories.

Equivalence Relations

Extensional equivalences play a central role in reasoning about the external behaviour of systems described by process algebraic languages (see, for example [42, 31]). Two processes are considered equivalent in this context if they cannot be distinguished by external observation, i.e. short of taking them apart. Examples of these equivalences are observation equivalence [42] or various testing equivalences [12].
Certain testing theories are naturally based on the notion of indistinguishability by observation. Testing based on equivalence relations of this kind ensures that the implementation will behave exactly as prescribed by specification. For testing communication protocols, equivalence relations \textit{conf-eq} [39] and \textit{te} [2] have been proposed as a criterion for establishing that a certain protocol implementation is a valid implementation of a given specification.

\textit{Implementation Relations}

Implementation relations [39] allow the notion of validity between an implementation and its specification to be extended to those implementations which are not externally equivalent to their specification. These relations have been less studied within the process algebraic techniques than equivalences.

Some implementation relations based on the idea of reducing nondeterminism as a valid implementation choice have been proposed so far for the use in testing. Examples are \textit{conf}, \textit{conf*}, \textit{red}, \textit{ext} [2, 39] defined for communication protocols and distributed systems in the context of conformance testing.

We proceed by first defining and briefly examining the strings equivalence and the implementation relations \textit{conf} and \textit{red}. We then evaluate these relations informally with respect to the level of interoperability achievable between implementations which are valid in the sense of these relations. Our results apply to other relations mentioned in the introduction in much the same way.

\section*{5-1.1 Strings Equivalence}

The most straightforward proposal for systems equivalence is to consider as equivalent two systems which can perform exactly the same sequences of visible actions. Let $P_1$ and $P_2$ be processes. It is clear that $P_1 \sim P_2$ iff $Tr(P_1) = Tr(P_2)$ and it is obvious that it is
an equivalence relation, which is termed *strings equivalence* ([12] and Definition 2.2).

Consider the following example.

**Example 5.1** \(S := EstStreamReq_{T1,T2};\)

\[
( S\text{Connect}_{T1}; S\text{Connect}_{T2}; S')
\]

\[
[]
\]

\[
( S\text{Connect}_{T2}; S\text{Connect}_{T1}; S' )
\]

\(S' := ( S\text{Accept}_{T1}; S\text{Refuse}_{T2}; S'' )\)

\[
[]
\]

\[
( S\text{Refuse}_{T2}; S\text{Accept}_{T1}; S'' )
\]

\(S'' := ( \text{ConfInd}_{T1}; \text{ConRejInd}_{T2}; D )\)

\[
[]
\]

\[
( \text{ConRejInd}_{T2}; \text{ConfInd}_{T1}; D )
\]

The above is an example of a process (an excerpt from the connection establishment phase of the ST-II communication protocol [15], for the origin agent only). The ST-II protocol at origin may receive a stimuli in the form of a (multimedia) stream establishment request from an application above, towards a number of targets (\(T1\) and \(T2\) in the example). For simplicity, we have instantiated one possible situation (where one target accepts and the other target rejects the connection), resolved the parallel composition of two simultaneous connections into a simpler choice (\(\square\) ) structure, and shown some of the events and interleavings only. The synchronization tree of this process is given in Figure 5.1 (a). (Figure 5.1 (b) represents an ST-II agent as a target agent, again showing a simple instantiated case). An implementation \(I_S\) which is strings equivalent to the specification \(S\) will have the following trace set, for traces of length \(\leq 3\), (\(\varepsilon\) is the empty trace)

\[
Tr(I_S) = \{\varepsilon, EstStreamReq_{T1,T2}, EstStreamReq_{T1,T2}.S\text{Connect}_{T1}\}
\]
5-1. IMPLEMENTATION RELATIONS AND INTEROPERABILITY

Connection Establishment Phase of ST-II protocol

Figure 5.1

\[ \text{EstStreamReq}_{T1,T2}, \text{SConnect}_{T2}, \]
\[ \text{EstStreamReq}_{T1,T2}, \text{SConnect}_{T1}, \text{SConnect}_{T2}, \]
\[ \text{EstStreamReq}_{T1,T2}, \text{SConnect}_{T2}, \text{SConnect}_{T1}, \]

(*) This branch evolves in a manner similar to the subtree starting at TConnect \( T1 \)
5-1.2 Implementation Relations \textit{conf} and \textit{red}

More involved concepts of equivalence between a specification and an implementation include the consideration of the traces that can be observed at the interface of a system with its environment as well as the sets of actions that may be refused after a certain trace. Such relations are \textit{conf} and \textit{red}. We here briefly recall their definitions in trace-refusal formalism (notation given in Appendix A). A more detailed discussion of these relations can be found in Chapter 2.

\textbf{Definition 5.1} ([39])

\[ P_1 \text{conf} P_2 \text{ iff } \forall \sigma \in \text{Tr}(P_2) \text{ we have } \text{Ref}(P_1, \sigma) \subseteq \text{Ref}(P_2, \sigma) \]

or equivalently

\[ P_1 \text{conf} P_2 \text{ iff } \forall \sigma \in \text{Tr}(P_2) \cap \text{Tr}(P_1) \text{ we have } \text{Ref}(P_1, \sigma) \subseteq \text{Ref}(P_2, \sigma). \]

Informally, \( P_1 \text{conf} P_2 \) iff, placed in any environment whose traces are limited to those in \( P_2 \), \( P_1 \) cannot deadlock when \( P_2 \) cannot deadlock. This relation is known as \textit{conformance} relation in protocol testing theory and is used for conformance testing of protocol implementations [2, 39].

\textit{Red} is the reduction relation as defined in Chapter 2. It limits the traces of a "\textit{red}"-valid process \( P_1 \) to those of \( P_2 \), but the essential deadlock property remains the same as in Def. 2.3. These two relations are the basis for the equivalence relations \textit{conf-eq} and \textit{te} [39, 2] proposed in protocol testing.

5-1.3 Implementation Relations and Interoperability

Consider the implementation relations defined above on the example ST-II origin specification \( S \) and the sample implementations \( I_1 \) and \( I_2 \) of this specification, depicted in
5-1. IMPLEMENTATION RELATIONS AND INTEROPERABILITY

Figure 5.2 (a) and (b).

![Diagrams with nodes and edges representing process interactions]

Different implementations of a process $S$  

Figure 5.2

**Example 5.2** The process $I_1$ (Figure 5.2(a))

$I_1 := EstStreamReq(T_1, T_2);$

$((SConnect_{T_1}; SConnect_{T_2}; I_1'));$

$[]$

$(SConnect_{T_2}; SConnect_{T_1}; I_1'));$

$I_1' := (SAccept_{T_1}; SRefuse_{T_2}; S'n)$
is neither in relation \textit{conf} nor \textit{red} with the process (specification) \( S \). This is because the refusal set \( \text{Ref}(I_1, \sigma_1) \), where \( \sigma_1 = \text{EstStreamReq}(T_1, T_2). \text{SConnect}_{T_1}. \text{SConnect}_{T_2} \) (or \( \sigma'_1 = \text{EstStreamReq}(T_1, T_2). \text{SConnect}_{T_2}. \text{SConnect}_{T_1} \)) includes, among other sets, also the set \( \{ \text{SRefuse}_{T_2} \} \), which is not in the refusal set for the same trace in the process (specification) \( S \).

\textbf{Example 5.3} The process \( I_2 \) (Figure 5.2 (b))

\( I_2 := \text{EstStreamReq}(T_1, T_2); \\
\quad (\text{SConnect}_{T_1}; \text{SConnect}_{T_2}; I'_2) \\
\quad [] \\
\quad (\text{SConnect}_{T_2}; \text{SConnect}_{T_1}; I'_2)) \\
I'_2 := (\text{SAccept}_{T_1}; \text{SRefuse}_{T_2}; I''_2) \\
\quad [] \\
\quad (\text{SRefuse}_{T_2}; \text{SAccept}_{T_1}; I''_2) \\
I''_2 := \text{ConRejInd}_{T_2}; \text{ConConfInd}_{T_1}; D \\
\)

is neither in relation \textit{conf} nor \textit{red} with the process (specification) \( S \). This is because the refusal set \( \text{Ref}(I_2, \sigma_2) \), where \( \sigma_2 = \text{EstStreamReq}(T_1, T_2). \text{SConnect}_{T_1}. \text{SConnect}_{T_2}. \text{SAccept}_{T_1}. \text{SRefuse}_{T_2} \), contains, among other sets, also the set \( \{ \text{ConConfInd}_{T_1} \} \), which is not in the refusal set for the same trace in the specification of the process \( S \). (It is easy to see that there are three more traces which share the same characteristic with the trace \( \sigma_2 \).)

Notice also, that neither of the equivalences generated by these implementation relations holds.

This is because an observer (in the environment of these implementations) will be unable to observe some traces at the interface of these implementations, although they should be present in all implementations of \( S \) that are valid in the sense of \textit{conf} and \textit{red}. But are the implementations \( I_1 \) and \( I_2 \) equal with respect to their ability to interoperate,
as origin agents, with other ST-II implementations? A brief informal investigation (we defer the formal interoperability analysis for a later section) and some knowledge of the semantics of the ST-II protocol specification shows that the implementation I1 may fail to interoperate with a full implementation of ST-II, if it is presented with the event $SRefuse_T^2$ before the event $SAccept_T^1$. However, the implementation I2 will always successfully interoperate with any other implementation which fully implements ST-II. Even more significantly, the implementation I3 of the same specification $S$, depicted in Figure 5.3, also possesses the same ability to interoperate with full ST-II implementations under all circumstances, although it is not even strings equivalent to $S$ for very short traces (refer to traces $Tr(I_S)$ in Example 5.1).

There are frequent situations in network protocols where the choice to drop some of the traces in an implementation of $S$ will affect the externally observable behaviour of the implementation but not its ability to successfully interoperate with other implementations of $S$. We quote some of those:

- one external stimuli (of the type $EstStream$ request) which generates multiple protocol events which may be arbitrarily interleaved (e.g., events $SConnect_{T_1}$, $SConnect_{T_2}$, ..., $SConnect_{T_n}$, and implementation I3)

- accumulation of external stimuli (of the type $SAccept$ or $SRefuse$ Protocol Data Units in the example) which generates protocol events which may be arbitrarily temporally ordered, provided they individually satisfy timing correctness (e.g., events $ConConfInd$ and $ConRejInd$ and implementation I2)

- multiple independent network connections (represented by the arbitrary interleaving in the interleaving model of concurrency) where any one of many possible interleavings of certain events may be sufficient for interoperability
Note: Notice that, although many events initiated by the implementation under observation will be in one of these categories, not all are: consider, for example, certain priority control PDUs or express PDUs.

The observed problem is due to the fact that the formal theory of protocol testing based on observation fails to distinguish between different aspects of the inability of an implementation to evolve via specific events. The consequence is that, in testing which is based on these equivalences and implementation relations, the inability of implementations such as \( I_1 \) and \( I_2 \) to synchronize on some specific observable events is treated in the same manner.
This characteristic has a profound impact on the design of the testing process itself.

1. In such a theory a tester must be able and will try to synchronize on all traces and observable events as described by the specification of the tested system, which may result in numerous inconclusive test runs if an implementation fails to react with the exact ordering of events that the tester expects to see. Other solutions [1] may include imposing artificial requirements on the testing process (such as establishing a stable state before proceeding with testing), which may reduce the error exposition probability of the testing process.

2. In addition, test selection under such premises becomes difficult to solve effectively within the theory itself. A possible test selection scenario could drop some traces that are crucial for interoperability (intuitively, an implementation, such as I1, must be tested to be able to synchronize on $S_{Accept}$ PDU followed by a $S_{Refuse}$ PDU at any time, as well as vice versa), in favour of such traces which cannot even be observed by testing for a particular implementation (for instance, a $ConConfInd$ followed by $ConRejInd$, in case of the implementation I2), and which may be irrelevant for the interoperability of such implementation.

To overcome these problems, we propose a new formal approach aimed at resolving some practical test design issues and improve the efficiency of multiconnection protocol testing in particular. The testing will still be based on observation, but the underlying formal relation of validity will be greatly relaxed to include such implementations which, on experimentation, are observed to have many fewer traces than the specification. The successful termination of the testing process (if the theoretical upper bound were reachable) would guarantee that all implementations of the same specification which pass will be able to interoperate.
CHAPTER 5. A FRAMEWORK FOR INTEROPERABILITY TESTING

5-2 The Interoperability Relations

In this section we define a new formal notion of validity between an implementation and a specification of a communication system. Although the definition appears more involved compared to other implementation relations, it turns out that the test design based on this relation differs only slightly than the design based on other implementation relations, whereas the testing based on this relation is much more efficient. We delay these practical considerations to the next section, and concentrate on theoretical development next.

5-2.1 The \textit{intop} Relation

We presuppose the existence of two subsets \(L_{\text{req}}\) and \(L_{\text{alt}}\) of labelsets of visible actions in \(L\), such that:

\[
L_{\text{req}} \cap L_{\text{alt}} = \emptyset \quad \text{and} \quad L_{\text{req}} \cup L_{\text{alt}} = L
\]

Intuitively, we shall think of the elements of \(L_{\text{req}}\) as events which must be observable at the interface of an implementation whenever the specification allows the possibility of synchronizing on that event in the state in which the implementation is at the moment of observation (i.e., the "required" synchronization). On the contrary, the elements of \(L_{\text{alt}}\) are such events, which may not necessarily be observable at the interface of an implementation, although the specification allows the possibility of synchronizing on one such event at that point (i.e. the "alternative" synchronization).

Similarly, let \(\text{Ref}_{\text{req}}(P, \sigma) = \text{Ref}(P, \sigma) \cap \mathcal{P}(L_{\text{req}})\) and \(\text{Ref}_{\text{alt}}(P, \sigma) = \text{Ref}(P, \sigma) \cap \mathcal{P}(L_{\text{alt}})\) (where \(\mathcal{P}(A)\) denotes the powerset (the set of all subsets) of a set \(A\)) be the \(\subseteq\)-maximal subsets of the refusal set \(\text{Ref}(P, \sigma)\) which are completely contained in the powersets of \(L_{\text{req}}\) and \(L_{\text{alt}}\), respectively.

Notice that these two sets always exist and are unique. Observe that the refusal sets satisfy the following properties.
5-2. THE INTEROPERABILITY RELATIONS

Proposition 5.1 (Properties of refusal sets):

1. \( \text{Ref}_{\text{req}}(P, \sigma) \) and \( \text{Ref}_{\text{alt}}(P, \sigma) \) are subset closed

2. \( \text{Ref}_{\text{req}}(P, \sigma) \cup \text{Ref}_{\text{alt}}(P, \sigma) \subseteq \text{Ref}(P, \sigma) \) (not necessarily =)

3. \( \text{Ref}_{\text{req}}(P, \sigma) = \phi \Rightarrow \text{Ref}_{\text{alt}}(P, \sigma) = \text{Ref}(P, \sigma) \) and \( \text{Ref}_{\text{alt}}(P, \sigma) = \phi \Rightarrow \text{Ref}_{\text{req}}(P, \sigma) = \text{Ref}(P, \sigma) \)

4. \( (\text{Ref}_{\text{req}}(P, \sigma) \subseteq \text{Ref}(P, \sigma) \lor \text{Ref}_{\text{alt}}(P, \sigma) \subseteq \text{Ref}(P, \sigma)) \Rightarrow \text{Ref}_{\text{req}}(P, \sigma) \cup \text{Ref}_{\text{alt}}(P, \sigma) \subseteq \text{Ref}(P, \sigma) \) (proper subsets)

5. \( \text{Ref}(P, \sigma) \subseteq \text{Ref}(Q, \sigma) \Rightarrow (\text{Ref}_{\text{req}}(P, \sigma) \subseteq \text{Ref}_{\text{req}}(Q, \sigma) \land \text{Ref}_{\text{alt}}(P, \sigma) \subseteq \text{Ref}_{\text{alt}}(Q, \sigma)) \)
   \( \text{(the opposite does not necessarily hold)} \)

6. \( P(\{R \in \text{Ref}_{\text{req}}(P, \sigma) \mid R \cup \text{Ref}_{\text{alt}}(P, \sigma) \}) \supseteq \text{Ref}(P, \sigma) \)

Let \( I \) be an implementation of a specification \( S \).

Definition 5.2 \( I \text{ intop} S \) iff \( \forall \sigma \in \text{Tr}(S) \cap \text{Tr}(I) \) we have

1. \( \text{Ref}_{\text{req}}(I, \sigma) \subseteq \text{Ref}_{\text{req}}(S, \sigma) \), and

2. \( L_{\text{alt}} \cap \text{Out}(S, \sigma) \cap \text{Out}(I, \sigma) \neq \phi \lor L_{\text{alt}} \in \text{Ref}_{\text{alt}}(S, \sigma) \)

3. For the set \( A = L_{\text{alt}} \cap (\text{Out}(S, \sigma) \setminus \text{Out}(I, \sigma)) \) and \( R \in \text{Ref}(I, \sigma) \) we have

\[
\text{Ref}(I, \sigma) \setminus \{R \mid R \cap A \neq \phi\} \subseteq \text{Ref}(S, \sigma)
\]

Informally, \( I \text{ intop} S \) iff, when placed in an environment whose traces are limited to the traces common to \( S \) and \( I \), (i) the implementation \( I \) cannot deadlock on any events from \( L_{\text{req}} \) on which \( S \) cannot deadlock; (ii) the implementation \( I \) cannot deadlock on all events from \( L_{\text{alt}} \) on which \( S \) cannot deadlock.
As an example, consider the validity of the implementation $I_3$ in the sense of $\text{intop}$ (it is easy to observe that $I_3$ is neither in $\text{conf}$ nor $\text{red}$ relation with $S$).

**Example 5.4** Let $L_{\text{req}} = \{\text{EstStreamReq}_{T_1,T_2}, \text{SAccept}_{T_1}, \text{SRefuse}_{T_2}\}$ and $L_{\text{alt}} = \{\text{SConnect}_{T_1}, \text{SConnect}_{T_2}, \text{ConConfInd}_{T_1}, \text{ConRejInd}_{T_2}\}$.

Consider the implementation $I_3$ of $S$, after the trace $\sigma = \text{EstStreamReq}_{T_1,T_2}$.

$\text{Ref}_{\text{req}}(I_3, \sigma) = \mathcal{P}(\{\text{EstStreamReq}_{T_1,T_2}, \text{SAccept}_{T_1}, \text{SRefuse}_{T_2}\})$

$\text{Ref}_{\text{alt}}(I_3, \sigma) = \mathcal{P}(\{\text{SConnect}_{T_2}, \text{ConConfInd}_{T_1}, \text{ConRejInd}_{T_2}\})$

Then,

1. $\text{Ref}_{\text{req}}(I_3, \sigma) \subseteq \mathcal{P}(\{\text{EstStreamReq}_{T_1,T_2}, \text{SAccept}_{T_1}, \text{SRefuse}_{T_2}\} = \text{Ref}_{\text{req}}(S, \sigma)$,
   and

2. $\text{SConnect}_{T_1} \in L_{\text{alt}}$, $\{\text{SConnect}_{T_1}\} \in \text{Out}(S, \sigma)$ and $\{\text{SConnect}_{T_1}\} \in \text{Out}(I, \sigma)$,
   and

3. $\text{Ref}(I_3, \sigma) = \mathcal{P}(\{\text{SConnect}_{T_2}, \text{ConConfInd}_{T_1}, \text{ConRejInd}_{T_2}\}) \cup \{\text{EstStreamReq}_{T_1,T_2}, \text{SAccept}_{T_1}, \text{SRefuse}_{T_2}\}$,
   and $A = \{\text{SConnect}_{T_2}\}$. Therefore,

   $\text{Ref}(I_3, \sigma) \setminus \{R \mid R \cap \{\text{SConnect}_{T_2}\} \neq \emptyset\} = \mathcal{P}(\{\text{ConConfInd}_{T_1}, \text{ConRejInd}_{T_2}, \text{EstStreamReq}_{T_1,T_2}, \text{SAccept}_{T_1}, \text{SRefuse}_{T_2}\}) \subseteq \text{Ref}(S, \sigma)$.

Therefore, the conditions of Definition 5.2 are satisfied for $I$, $S$ and this trace $\sigma$.

### 5-2.2 Properties of the $\text{intop}$ Relation

We collect some easy facts about the $\text{intop}$ relation.
Proposition 5.2 Let $I$ intop $S$ and $I = \sigma \Rightarrow$.

1. $Ref_{alt}(I, \sigma) \subseteq Ref_{alt}(S, \sigma) \Rightarrow Ref(I, \sigma) \subseteq Ref(S, \sigma)$

2. $(\forall \sigma \in Tr(S) \cap Tr(I), Ref_{alt}(I, \sigma) \subseteq Ref_{alt}(S, \sigma)) \Rightarrow I \text{ conf } S$

3. $I \text{ conf } S \Rightarrow I$ intop $S$ (i.e. intop $\supseteq$ conf)

4. intop is reflexive

5. intop is not transitive

Proof. The proof for 1. follows directly from the observation that in the case when $Ref_{alt}(I, \sigma) \subseteq Ref_{alt}(S, \sigma)$ then the set $A$ from Def. 5.2, condition 3., is empty. Since also $I$ intop $S$, the condition in Def. 5.2, 3., holds which proves that $Ref(I, \sigma) \subseteq Ref(S, \sigma)$ holds. (Notice that this is exactly the condition needed for the opposite implication in Prop. 5.1, 5). 2. follows directly from 1. and the definition of conf relation.

3. Assume that $Ref(I, \sigma) \subseteq Ref(S, \sigma) \forall \sigma \in Tr(S) \cap Tr(I)$.

- We have $Ref_{req}(I, \sigma) = Ref(I, \sigma) \cap \mathcal{P}(L_{req})$ and $Ref_{req}(S, \sigma) = Ref(S, \sigma) \cap \mathcal{P}(L_{req})$. Therefore, $Ref_{req}(I, \sigma) = Ref(I, \sigma) \cap \mathcal{P}(L_{req}) \subseteq Ref(S, \sigma) \cap \mathcal{P}(L_{req}) = Ref_{req}(S, \sigma)$.

- Assume that the first part of Def. 5.2, 2., is not true. Then $\forall a \in L_{alt}$ such that $a \in Out(S, \sigma)$ we have $a \not\in Out(I, \sigma)$. Hence $\{a\} \in Ref_{alt}(I, \sigma) \subseteq Ref_{alt}(S, \sigma)$ by Prop. 5.1, 5. We clearly have that the union of $Out(S, \sigma) \cap L_{alt}$ and all one-element subsets of $Ref_{alt}(S, \sigma)$ must be equal to $L_{alt}$. However, since $Out(S, \sigma) \cap L_{alt} \in Ref_{alt}(S, \sigma)$, it follows that $L_{alt} \in Ref_{alt}(S, \sigma)$, which proves that the second condition of the Def. 5.2 holds.

- The part 3. of Def. 5.2 holds trivially.
This proves that $\text{intop} \supset \text{conf}$.

4. and 5. follow directly from the definition of $\text{intop}$ relation.

5-2.3 The $\text{intop}_{\text{red}}$ Relation

The results of Proposition 5.2 (4) and (5) are in keeping with the theory developed in [39], that a valid implementation relation must be reflexive (because specification is a valid implementation of itself), but not necessarily symmetric or transitive (because the implementation and specification are not in general interchangeable). However, notice that similarly to $\text{conf}$, the relation $\text{intop}$ allows that additional traces may exist in the implementation, that are not part of the specification. This feature becomes even more critical when interoperability of different implementations is examined, since, in general, such implementations may synchronize on traces that are not in their common specification. For such traces, the concept of interoperability is really hard to define both formally and informally. We therefore extend the formal notion of interoperability by defining a relation which restricts the traces in an implementation to those of the specification. Unlike $\text{intop}$, this relation is also transitive.

**Definition 5.3** $I \text{intop}_{\text{red}} S$ iff

1. $\text{Tr}(I) \subseteq \text{Tr}(S)$

2. $I \text{intop} S$

**Proposition 5.3** The relation $\text{intop}_{\text{red}}$ has the following properties:

1. $\text{intop} \supset \text{intop}_{\text{red}}$

2. $\text{intop}_{\text{red}} \supset \text{red}$
3. intop\textsubscript{red} is a preorder

**Proof.** The proofs for 1. and 2. are easy and follow directly from the definitions of the corresponding implementation relations.

We will prove that intop\textsubscript{red} is a preorder.

1. intop\textsubscript{red} is reflexive since \( Tr(I) \subseteq Tr(I) \) and \( I \) intop \( I \) (by Proposition 5.2., 4.)

2. intop\textsubscript{red} is transitive: Let \( I \) intop\textsubscript{red} \( J \) and \( J \) intop\textsubscript{red} \( K \). Then:

(a) \(( Tr(I) \subseteq Tr(J) \) and \( Tr(J) \subseteq Tr(K) \)) \( \Rightarrow \) \( Tr(I) \subseteq Tr(K) \)

(b) Let \( \sigma \in Tr(K) \). Then

i. \( \text{Ref}_\text{req}(I, \sigma) \subseteq \text{Ref}_\text{req}(J, \sigma) \) (from \( I \) intop \( J \)) and similarly
\( \text{Ref}_\text{req}(J, \sigma) \subseteq \text{Ref}_\text{req}(K, \sigma) \). These two relations imply \( \text{Ref}_\text{req}(I, \sigma) \subseteq \text{Ref}_\text{req}(K, \sigma) \).

ii. By the definition of the intop relation, \( \exists a \in \text{L} \cap \text{Out}(J, \sigma) \) such that
\( a \in \text{Out}(I, \sigma) \). Since \( \sigma.a \in Tr(J) \subseteq Tr(K) \) we conclude that
\( a \in \text{L} \cap \text{Out}(K, \sigma) \) and \( a \in \text{Out}(I, \sigma) \). Therefore, the second requirement of Def. 5.2 is satisfied.

iii. Denote by \( B = \text{L} \cap (\text{Out}(K, \sigma) \setminus \text{Out}(I, \sigma)) \) and by \( C = \text{L} \cap (\text{Out}(K, \sigma) \setminus \text{Out}(J, \sigma)) \)

Then, \( a \in C \Rightarrow \sigma.a \in Tr(K) \), but \( \sigma.a \notin Tr(J) \). Therefore, \( \sigma.a \notin Tr(I) \). Thus \( a \notin \text{Out}(I, \sigma) \). Hence \( C \subseteq B \).

Since \( J \) intop \( K \) we have
\( \text{Ref}(J, \sigma) \setminus \{ R \mid R \cap C \neq \emptyset \} \subseteq \text{Ref}(K, \sigma) \)

and consequently \( \text{Ref}(J, \sigma) \setminus \{ R \mid R \cap B \neq \emptyset \} \subseteq \text{Ref}(K, \sigma) \). (*)

Put \( A = \text{L} \cap (\text{Out}(J, \sigma) \setminus \text{Out}(I, \sigma)) \). Then,
\( \text{Ref}(I, \sigma) \setminus \{ R \mid R \cap A \neq \emptyset \} \subseteq \text{Ref}(J, \sigma) \).

We have \( a \in A \Rightarrow \sigma.a \notin Tr(I) \Rightarrow \sigma.a \in Tr(K) \), \( \sigma.a \notin Tr(I) \Rightarrow a \in B \).

Thus, \( A \subseteq B \). So, \( \{ R \mid R \cap A \neq \emptyset \} \subseteq \{ R \mid R \cap B \neq \emptyset \} \) and therefore
\( \text{Ref}(I, \sigma) \setminus \{ R \mid R \cap B \neq \emptyset \} \subseteq \text{Ref}(I, \sigma) \setminus \{ R \mid R \cap A \neq \emptyset \} \subseteq \text{Ref}(J, \sigma) \).
Consequently, \( \text{Ref}(I, \sigma) \setminus \{R \mid R \cap B \neq \emptyset\} \subseteq \text{Ref}(J, \sigma) \setminus \{R \mid R \cap B \neq \emptyset\} \).

By (*) it follows that
\[
\text{Ref}(I, \sigma) \setminus \{R \mid R \cap B \neq \emptyset\} \subseteq \text{Ref}(K, \sigma).
\]

This proves the third property in Def. 5.2, therefore \( I \text{ intop} K \).

(a) and (b) together prove that \( \text{intop}_{\text{red}} \) is transitive. Therefore, \( \text{intop}_{\text{red}} \) is a preorder.

The above theoretical considerations are sufficient as a basis for specifying the necessary architectural and design requirements in interoperability testing. It is worth noting that the formal notion of interoperability as an implementation relation can be extended in the sense of \( \text{imp-eq} \) and other formal theory given in [39].

5-3 The Interoperability Tester Design

After establishing the necessary theoretical basis in the previous section, we turn our attention towards some practical considerations in interoperability testing of network protocols. Based on these considerations, we outline the design of a network protocol interoperability tester whose theoretical upper bound is the satisfaction of the interoperability relation \( \text{intop} \) between an Implementation Under Test (IUT) and its specification \( S \).

5-3.1 Architectural considerations

The general theory of protocol testing allows for different test architectures and different test interfaces. Consider the test architecture given in Figure 5.4. Generally, the Points of Control and Observation (PCOs) may be positioned at the upper IUT interface (PCO1, PCO4) as in the system \( SUT1 \) in Fig. 5.4, or at the lower IUT interface (PCO2, PCO3).
For interoperability testing of protocol implementations it is necessary to observe both upper interface (service) PCOs and lower interface PCOs (as in SUT2 of Fig. 5.4.) in order to ensure the proper internetworking of different implementations in all environments. In this chapter, we concentrate on the interoperability of different implementations of the same (peer) protocols. (It may be interesting to study to what extent the same theory applies towards the interoperability of any two implementations that share the same interface.) To take advantage of our theory, we model the interoperability test architecture in the following manner:

1. $I$ is a protocol implementation

2. $IT$ is the interoperability tester

3. $NET$ is the underlying (network) connection between the $I$ and $IT$. $NET$ behaves as a reliable FIFO channel in either direction (FIFO without loss)$^1$

$^1$In our example specification, we adopt a simple strategy to prefix the name of each primitive by the address at which it occurs including $S$ for the source (calling) ST-II agent and $T$ for the target (called)
4. A tester $IT$ is capable of observing at least one set of PCOs at the upper interface of $I$ and one set of PCOs at the lower interface of $I$ or $IT$.

5. A tester $IT$ is capable of controlling PCO1 and either PCO2 or PCO3.

We will also assume that an executable tester (an implementation of the interoperability tester $IT$) is capable of executing strong control over the PCOs it controls in the following manner: it will always be able to synchronize on any events that are its output events, before synchronizing on any events that are its input. In particular, we expect that an executable tester is able to send a PDU into network or request a service from an IUT at any time. If this assumption holds, then the possibility of inconclusive test runs linked to the tester trying to observe a particular event in $L_{req}$ as the next event is eliminated. This may or may not hold for actual implementations, but will simplify our further analysis.

5.3.2 Formal Network Protocol Specification Issues

The requirements of interoperability testing regarding the observability and controllability of PCOs impose strict requirements on the formal specification style of a protocol process to be tested. In LOTOS, this style requires that gates modelling the PCO1 and one of PCO2 or PCO3 not be hidden (i.e., event synchronization at these gates is visible). The observability of PCO4 is entirely optional and depends on the executable tester design. We require that all the protocol processes be fully synchronized with the underlying process $NET$ representing the network.

For illustration, we complete our example ST-II specification of the stream establishment phase with the specification of the target process, supplying the remaining visible interactions at these gates. We simplify the specification in the manner similar to Example ST-II agent, in order to allow for a simple $NET$ process modelling. Other specification possibilities that provide distinctness of interactions exist and are equally suitable for this setting (see [19] for architectural details of interaction points in various specification formalisms).
5.1. (We will not worry about the details of the negotiation of connection establishment with the multimedia application above and let the ST-II target agent decide on acceptance or refusal of the connection itself.)

Example 5.5 The target ST-II agent $T$ is the process

\[
T := (T_{Connect\ T_1}; T_{Accept\ T_1}; D) \\
| | | \\
(T_{Connect\ T_2}; T_{Reject\ T_2}; T')
\]

The full specification of our example ST-II process is the independent parallel composition of the processes $S$ and $T$,

\[
ST := S ||| T
\]

and is represented as a parallel composition of the synchronization trees given in Figure 5.1.

The specification of the NET FIFO process can be given as in [19]. For the purposes of our brief example we will informally observe that for every event prefixed by $T$, i.e. event $Te$, on which the NET process synchronizes at interaction points $PCO2$ ($PCO3$), it will subsequently synchronize on an event $Se$ at $PCO3$ ($PCO2$ respectively) distinguishable from the event $Te$ by its prefix $S$ only, after which it is ready for a new interaction at $PCO3$ or $PCO2$. Similarly, if the NET process synchronizes on an $Se$ event at the interaction points $PCO2$ ($PCO3$) first, then it will subsequently synchronize on a $Te$ event at $PCO3$ ($PCO2$) and the process NET is ready for a new interaction. Notice that the parallel composition of the processes $ST$ and $NET$ with all gates observable, will yield exactly the traces of our full example ST-II specification. The following example illustrates this behaviour.

Example 5.6 Consider the application of $EstStreamReq(T_1, T_2)$ at the PCO1, and assume that the additional revealed PCOs are PCO2, PCO3 and PCO4 (the system specified is exactly $SUT2$ in Fig. 5.4). Then the following trace may be observable at these PCOs:
CHAPTER 5. A FRAMEWORK FOR INTEROPERABILITY TESTING

\[ \sigma = \text{EstStreamReq}_{T_1,T_2} \cdot S\text{Connect}_{T_1} \cdot S\text{Connect}_{T_2} \cdot T\text{Connect}_{T_1} \cdot T\text{Connect}_{T_2} \cdot T\text{Accept}_{T_1} \cdot T\text{Refuse}_{T_2} \cdot S\text{Accept}_{T_1} \cdot S\text{Refuse}_{T_2} \cdot \text{ConRejInd}_{T_2} \cdot \text{ConConfInd}_{T_1} \]

5-3.3 Interoperability Tester Design

In this section we introduce the design of the interoperability tester \( IT(S) \) for protocol implementations. The purpose of the interoperability tester is to properly distinguish between implementations that do or do not satisfy the \( \text{intop} \) relation with respect to their specification \( S \).

The construction of the interoperability tester \( IT(S) \) is based on the the canonical tester \([2, 39]\). The canonical tester \( T(S) \) is constructed systematically from the specification of the system \( S \) and is defined in the following manner.

**Definition 5.4** Let \( S \) be a specification. The canonical tester of \( S \), \( T(S) \), is defined implicitly as a solution \( X \) satisfying the following two equations:

1. \( Tr(X) = Tr(S) \)

2. \( \forall I \ I\text{conf} S \iff (\forall \sigma \in L^* \text{ we have } (L \in \text{Ref}(I \mid X, \sigma) \Rightarrow L \in \text{Ref}(X, \sigma))) \)

We first observe that by the Proposition 5.2, 3., \( \text{intop} \supset \text{conf} \). Using this observation, we can transfer the problem of finding the interoperability tester \( IT(S) \) to "relaxing" the structure of the canonical tester \( T(S) \). We here do not repeat the theoretical work of \([2, 39]\), but assume that the canonical tester \( T(S) \) of a specification \( S \) is given. The crucial observation in the construction of the \( IT(S) \) is that the only behaviours that are treated differently in the relations \( \text{conf} \) and \( \text{intop} \) are precisely the ones that allow the choice over actions that are in \( L_{alt} \), after some observed trace \( \sigma \).

In the canonical tester, the choice over the different actions in \( L \) is replaced by the internal choice, i.e. these actions are prefixed by the internal action \( i \). Consider the
example given in Figure 5.5, where $S$, $T(S)$ and $IT(S)$ denote the specification, its canonical tester and its interoperability tester. In the derivation of the canonical tester $T(S)$, the purpose of this choice over $i$ is to allow the tester to attempt synchronization on each action (a, b, c in the example) in $L$ at that point in the specification.

By the definition 5.2 (of the intop relation), evolving via actions in $L_{req}$ ($I = a \Rightarrow I'$ of the example) is also mandatory for every implementation $I$, and we therefore leave the design of $T(S)$ intact with respect to such actions (the trace $i.a.\delta$). The tester $IT(S)$ will also require synchronization on each such sequence.

However, by the definition of the intop relation, for actions in $L_{alt}$, not all possible actions need to be observable or observed at that particular point. In the example, either $I = b \Rightarrow I'$ or $I = c \Rightarrow I''$ must be observable. It follows that the tester $IT(S)$ must attempt synchronization on either $IT(S) = b \Rightarrow$ or $IT(S) = c \Rightarrow$. In such cases, $IT(S)$ is derived from the canonical tester $T(S)$ by substituting the $n$ internal choices of $T(S)$,
each followed by one of the \( n \) different actions in \( A \subseteq L_{alt} \), by one internal choice in \( IT(S) \) followed by the external choice over the \( n \) different actions in \( A \subseteq L_{alt} \). Therefore, \( T(S) := i; a[i]; b[i]; c \) in the example becomes \( IT(S) := i; a[i]; (b[c]) \). The interoperability tester will therefore resolve such a choice in the course of the interaction with the environment (for example, driven by the choice of the peer protocol implementation), rather than by attempting to synchronize on each one of the actions. More specifically, if a node in the synchronization tree of the canonical tester \( T(S) \) has the general form

\[
\{a_p; \ldots | p \in P\} \{i; b_q; \ldots | q \in Q\}
\]

then it is transformed into a node of the synchronization tree of the interoperability tester \( IT(S) \) of the form

\[
\{a_p; \ldots | p \in P\} \{i; b_r; \ldots | b_r \in R \subseteq Q\} \{i; \{b_a; \ldots | b_a \in A \subseteq Q\}\}
\]

where \( R \) is the set of all \( q \in Q \) such that \( b_q \in L_{req} \) and \( A \) is the set of all \( q \in Q \) such that \( b_q \in L_{alt} \).

All other nodes and branches of \( T(S) \) are left intact.

We observe that the execution of \( IT(S) \) against an implementation \( I \) of a protocol will have the following impact on the testing process:

- Within the theory itself, such a selection process can be designed, which will guarantee not to sacrifice traces needed for interoperability in favour of possibly unobservable traces.

Observe that all the events that can happen alternatively are collected under the nodes which have all emanating branches labelled with the events in \( L_{alt} \), and the branches leading to such nodes are labelled \( i \). Such nodes are uniquely distinguishable and should participate in the test selection with the weight representative of one test case only (other test selection criteria assumed to contribute separately).
• fewer traces need to be examined or observed (even if they happen to be implemented), resulting in a more efficient upper bound of the testing process

It follows immediately, from the construction of the tester and the definition of the relation \( \text{intop} \), if a branch labelled \( i \) leads to a node whose all emanating branches are labelled with the events in \( A \subseteq L_{alt} \), then the number of tests is reduced from the number of events in \( A \) to a subtree which is to be considered as one test case only.

• elimination of inconclusive test runs for test cases where one (of many) possible temporal orderings of events is sufficient to guarantee the interoperability of implementations

This observation follows directly from the fact that, for such events, the multiple internal choices of the canonical tester are substituted with one choice which is always possible to be resolved on interaction of the interoperability tester with the environment

Finally, we would like to note that we have illustrated the features of the \( \text{intop} \) relation on a theoretically sound exhaustive test generator (canonical tester). If the theory is to be practically applicable, it is crucial that fast test generators for complex systems (needles to say, that are theoretically sound in the coverage testing context), be available.

5-4 Interoperability Relation and the MB method

In this section, we finally draw the two ingredients of the coverage testing process for network protocols together: the theoretical upper bound for testing and the mode of approximation of this bound.

Recall that the choice of the testing domain (the entire control space of systems, Chapter 2-1.4), leaves us with two open questions: (1) when is a system a valid implementation
of its specification, or equivalently, what is the theoretical upper bound on the testing process whose satisfaction guarantees such validity, and (2) if the theoretical upper bound is not practically reachable, then what is the mode of its approximation.

The first question is the question of the validity relation, and the second is the question of test selection.

We suggest that the interoperability relation gives an efficient upper bound on the testing process for communication protocols. For the approximation mode, we suggest that the MB test selection method is a theoretically sound way to approximate finite and infinite spaces of protocol execution sequences based on pattern similarity and the recursion level of recursive pattern clusters.

In this context, the metric space for interoperability testing of communication protocols takes on the following topological structure.

Let \( (S, dt) \) be a metric space of execution sequences \( t (\in S) \) of atomic events of a communication protocol system. Let \( S = \{S_1, \ldots, S_n\} \) be a family of sets of sequences of events \( a_i \in L_{alt} \). Then, according to Theorem 3.8, \( (S(s), dt) \) is a totally bounded metric space. MB selection algorithm will, in this space, tend to approximate all sequences \( t(s) \in G(s) \subset S(s) \) by selected sequences \( s(s) \) to a target density \( \epsilon \) (Proposition 3.2, Consequence 3.1, Theorem 3.8). When \( \epsilon \) tends to zero, the selected set will tend to strings equivalence with the starting set, within the new metric space \( (S(s), dt) \). At the same time, the testing process based on the execution of selected sequences and the interoperability tester design, will tend to establish the interoperability relation between the implementation under test, and the set \( G(s) \).

This analysis shows, that the MB method can indeed be considered as a general method for test selection and coverage evaluation. It can support all target test relations for protocols which are primarily based, or can be interpreted in terms of string equivalence at different granularity levels of protocol interactions. Indeed, the work summarized in
[27], seems to indicate that this is the direction in which the testing of (distributed) systems should proceed. Non-sequential behavioural equivalences based on the notion of the labelled partial order (where in $p - e \rightarrow q$, e is not an atomic event but a partial order of events), or timed equivalences (again, a multitude of possibly interleaved action labels carries the system from one state to the other), introduce a much weaker notion of tests. It would be interesting to examine, to what extent these equivalences can be interpreted in the metric spaces $(S(s), dt)$ and $(S(s), dt)$, and supported by the MB method.
Chapter 6

Tests of the Theory

In this chapter the theoretical framework of the thesis is experimentally evaluated through the sensitivity analysis of the metric-based test selection and coverage evaluation processes. The sensitivity analysis is carried out with respect to a set of relevant protocol features. The performance and stability of the experimental results are also considered. Based on the experimental results, a feasibility analysis of the metric-based approach is presented in the conclusion of the chapter.

We first provide a brief overview of the ST-II protocol which is the case study used in the experimentation. This is followed by relevant implementation details of the experimental test generation, selection and evaluation tool called TESTGEN++ which was used throughout the experiments. After this introductory information on the experimental environment, we give a detailed evaluation of the problems considered in experimentation and the results reached.
6-1. **Internet Stream Protocol**

The Internet Stream Protocol has been used for several years as the primary protocol in a voice and video conferencing system which has operated over the Wideband and Terrestrial Wideband Networks. The revised experimental specification was issued in 1990, and is known as ST-II [15]. ST-II is a candidate protocol for the support of high-bandwidth, delay-constrained, multimedia applications such as conferencing, virtual environment support, and telepresence.

Its application area and novel design solutions make ST-II protocol interesting for experimentation in software engineering. There are currently three implementations of ST-II reported: (i) Swedish Institute of Computer Science (MultiG Project), (ii) IBM European Networking Center (HeiTS Project), (iii) BBN Systems and Technologies, Cambridge, MA. The research and development issues regarding the implementation of ST-II have been reported in literature [46]. The ST Control Message Protocol (SCMP), mainly used in our experiments, is reported to be a relatively complex communication protocol for implementation [46]. There has been, to our knowledge, no theoretical or practical research effort reported regarding the testing of ST-II.

### 6-1.1 General Protocol Features

The basic functionality of SCMP is to build *routing trees* between a sender, called *origin*, and one or more receiving applications, called *targets*. One such routing tree will typically include one or more additional nodes along each of its paths, called *intermediate agents*. ST (the data phase) then simply transmits data down the routing tree, from an origin to one or more targets. The data flow over the routing tree is referred to as a *stream*. Each of the communicating entities within a routing tree runs an ST-II protocol and is called an *ST-II agent*. One ST-II agent can simultaneously act as one or more origins, routers...
6-1.2 ST-II from the Testing Point of View

The following design features make ST-II challenging for testing (for details see [9]):
1. The concurrency in ST-II is due to two different sources: a) multiple protocol connections, and b) simultaneous execution of origin, target and intermediate agent functional entities within one ST-II agent.
2. With up to 17 different types of messages for SCMP part of ST-II, ST-II becomes one of the more complex protocols, especially if the combined state space composed of one origin, one target and one intermediate agent module is considered.
3. The usual concept of a "peer" partner in communication, as a convenient symmetrical counterpart in testing, has to be abandoned. The protocol is non-symmetrical since individual connections are trees.

We carried out the experimentation under the assumption that an ST-II protocol implementation is tested as an entity, composed of independent parallel executions of (several) origin, intermediate or target agents. Because of the non-symmetry of the communication pattern, the temporal order of protocol primitives arriving from IUT as a response to different upstream or downstream agents (even in one-stream environments) is unpredictable and should be abstracted from. Therefore, the interoperability relation is the proper theoretical upper bound for the testing process. The lack of the upper PCO-s for intermediate agents rules out some commonly used test architectures. The test architecture applicable in this setting is as given in Figure 5.4 (Interoperability Test Architecture), where a set of lower PCO-s must be observable.
6-2 The Implementation

The design of the functional structure of the Test Development System used for experimentation is depicted in Figure 6.1.

The TESTGEN module

TESTGEN [61] is the test generator module in an experimental protocol TEST Generation ENvironment for conformance testing developed at the University of British Columbia. The generated test suites incorporate both control flow testing and data flow testing with parameter variation. Both types of testing are controlled by a set of user-defined constraints which allows a user to focus on the protocol as a whole or just on
restricted areas of the protocol.

TESTGEN begins with a protocol description in an extended transition system formalism (closely related to Estelle [40]) and ASN.1 description of the input service primitives, output service primitives, and protocol data units. Once all constraints are defined, TESTGEN identifies subtours within the specified protocol which satisfy the minimum and maximum usage constraints. Each subtour then undergoes the parameter variation defined by the types of service primitives in the subtour and the parameter variation constraints.

The TESTSEL module

The TESTSEL module (which optionally includes an independent test coverage evaluator TESTCOV EVAL module), is an implementation of test case selection and test case coverage evaluation based on the distance and coverage metrics within the metric-based method. The TESTSEL algorithm uses a greedy approach for the test case selection. Since the test coverage metric we are using is guaranteed to be convergent when Cauchy sequences are followed, and the greedy algorithm produces such sequences, the greedy approach will iteratively yield a set of test cases which converges to the initial test suite. The implementation of the algorithm follows the design presented in Chapter 3 and [41].

The two modules, TESTGEN and TESTSEL have been integrated into a test development environment similar to the one already used for the development of the InRes protocol test sequences [41]. The seed files for test suites used in experimentation are generated by TESTGEN. These can be input into the TESTSEL module directly, or first passed through an independent interleaving algorithm. This algorithm produces random independent interleavings and random recursive levels of a specified number of streams and target agents, in order to obtain test suites exercising multiple simultaneous connections, concurrency of different agents, and higher levels of recursion.
6-3 Experimental Results

We first present some performance measurements for the integrated test development system used in the experimental ST-II protocol test generation. The sensitivity analysis of the metric based method, carried out through a series of experiments, is then presented in detail. Whereas the sensitivity analysis of test methods with respect to their ability to detect certain kinds of faults is customary, the characteristics of the MB method (that it does not target faults, but aims to cover the complete execution space in which faults are of unknown location and type), has lead us to develop a different kind of sensitivity analysis\(^1\). In our experimentation, we examine sensitivity of the method to detecting certain characteristics of the execution space covered by the test suite. We chose sensitivity to event pattern variation (Experiment 4), level of recursion (Experiment 5), multiconnection level (experiment 6) and missing events (Experiment 7), as most relevant to examine. The stability of the results observed and the feasibility of the approach are discussed in the summary following the experiments.

6-3.1 Computational Complexities and Performance Measurements

Computational complexities and performance measurements of the test selection and coverage evaluation algorithms were already thoroughly investigated and results reported in [41]. Figures 6.2 and 6.3 summarize the timing results for control sequences of length 20 and 100 (concise notation), by plotting the time needed for single iteration of the test selection algorithm to select different sizes of test sets.\(^2\) This testing was performed on a Sun4/75 with 16 Mb of available memory. Performance results were further confirmed in

---

\(^1\)Only Experiment 2 may be interpreted in the context of the sensitivity to fault identification.

\(^2\)The original implementation of test selection and coverage evaluation algorithms, as well as these performance measurements were done by Michael McAllister.
the experiments that follow, which were carried out on the same machine. The repeated measurements are not being shown separately.

![Graph showing timing results (in user seconds) for control sequences of length 20]

Figure 6.2

6-3.2 Sensitivity Analysis of the Metric Based Method

Introductory Information

The experimental input to the TESTGEN module was the Estelle-Y specification of ST-II origin, intermediate and target agents (Appendix E), and the data parameter description in ASN.1 (not given). Our ST Control Message Protocol functional description is relatively complete. Stream Setup, Data Transfer and Stream Tear Down have been specified, but Modifying an Existing Stream, Exceptional Cases and Ancillary functions have been omitted. The Estelle-Y specification deals with 11 out of 17 ST Protocol Data Units. The remaining 6 are related to the Exceptional cases and Ancillary functions (4), and
6-3. EXPERIMENTAL RESULTS

Modifying an existing stream (2). The specification has not been verified. We do not expect that minor inconsistencies with the RFC 1190 text should affect the validity of the experimental results of the thesis.

Most of the experiments initially use sets of transition tours generated by TESTGEN module for each of the agents separately. These sets are named originseeds, intermseeds and targetseeds (Appendix F). The purpose of these sets is to serve as the starting sets for the random interleaving and recursion generation process that we implemented specifically for the use in these experiments. These sets should preferably provide a thorough coverage of the relevant tours through each individual agent's state machine, in their simplest form (i.e. without concurrency or recursion). The random independent interleaving module generates test suites with concurrency of: a.) multiple streams (simultaneous connections) within the same ST-II agent, b.) multiple agent functional entities within the same ST-II agent, c.) multiple targets within the same stream. Concurrency due to the interleaving of multiple intermediate agents within the same stream has been omitted.

The first 4 PDU-s are also often not implemented [46], or their function is not clear [46].
In the experiments, we examine the effects of the recursion and concurrency of the ST Control Message Protocol only, since the data phase of this protocol is trivial.

We finally point out that each protocol event carries an identification a) to the stream it belongs to, and b) to the original TESTGEN transition tour it came from. On execution, or for the purposes of selection, identification a) identifies SAP (address), to which and event should be applied (or expected), and b) identifies data parameters of that particular PDU (SP). These can also be removed from test files if not needed. We have experimented with test suites containing input/output events, and input events only. The latter is more appropriate if the input relation is the theoretical upper bound to the testing process. Individual events of a test sequence should then be interspersed with subsets $S_1, \ldots, S_n$ of events in $L_{alt}$, (Chapter 5). This phase of the testing process has not been implemented.

### Stability and Granularity of the Test Selection Process

The next 3 experiments investigate the granularity and stability of the test selection process guided by two typical metric functions.

#### Experiment 1.1

The first experiment serves as an illustration of the test selection process. We start with the set originseeds (Appendix F), which contains 16 test cases. We select a metric such that $p_k = 1, k = 1, \ldots, 7$, and $\sum_{k=8}^{\infty} < 1$. We also set $|r_n - r_m| = 0, \forall n, m < 30$, and $r_n = 1 - \frac{1}{2^n}$, for $n \geq 30$.

Successive passes of the test selection algorithm are shown in Appendix F, as the process selects 6.56 dense, 5.90 dense, 4.78 dense, 3.87 dense, 2.82 dense, 1.85 dense and 0.98 dense test suites in originseeds.

In the next two experiments the function $p_k$ takes values $p_k = 2^{10-k}$, for $k = 1, \ldots, 10$, and $p_k = \frac{1}{2^{k-10}}$, for $k = 11, 12, \ldots$. The function $r_k$ takes values $r_k = 1 - \frac{1}{2^k}$ for all $k = 1, 2, \ldots$. 20 sets of 100 test cases were generated, representing random interleavings of
1 through 10 simultaneous network connections, and 10 through 80 ST-II target agents. Test sets are labelled tsc.t, indicating the number of simultaneous connections c and targets t that were allowed in their generation.

**Experiment 1.2**
The first series of 10 test sets contained test cases with individual events labelled by event names only. (This would be suitable for testing event variations at global levels, e.g. does an implementation correctly react to an event e after some event d.) Figure 6.4 shows the number of test cases selected versus refinements in the test density. x axis represents the density range of 0 through 1024.00, the diameter of the metric space generated by this metric. For this metric, results for ts10.35 and ts10.50 coincide. Also, the line styles of ts10.30 and ts10.35 are reused for ts1.80 and ts2.70 - the lower lines belong to ts1.80 and ts2.70.

**Experiment 1.3**
The second series of 10 test sets contained test cases with individual events labelled by event names, transition tour they belong to and the stream identifier. (This is a much finer identification which would allow detailed testing of event variations within individual connections and with respect to event parameter data variations). Figure 6.5 shows the granularity of the selection process for this case.

In both cases, i.e. given both very fine (many distinct event labels), and much coarser (relatively few distinct event labels) event space, there is a steady increase in the number of test cases selected as the density approaches 0. Figures at the low end of density spectrum indicate, that even at very fine granularity levels of density, this metric function is still capable of distinguishing still finer test suites. More test cases are selected for test suites in the second series of tests, for the same levels of density, number of simultaneous connections and recursion, since more distinct event labels can be identified. In both graphs, ts2.60 and ts2.70 occupied the middle portion of the graph, with the suites ts1.t ( ts5.t and ts10.t) almost entirely below (above, resp). Given same density levels and
Granularity of MB test selection with event name identification

Figure 6.4
Granularity of MB test selection with event name, gate and data parameters identification.

Figure 6.5
event labelling, fewer test cases are almost always selected for test suites involving fewer simultaneous connections, indicating a poorer event pattern. At the same time, given the same number of simultaneous connections, the effect of higher vertical recursion (more targets) on the number of test cases selected is not very noticeable, given the low values of the $r_k$ series for moderate values of $k$ in this example.

The stability and granularity of the MB selection which are observed in these two experiments, are confirmed throughout the test selection and coverage analysis experiments that follow.

**Identifying Transition Tours**

**Experiment 2**

This experiment investigates the efficiency of applying metric based method for transition tour selection, for single connection environments, with multiple agent functionality and with moderate vertical recursion. We allow 3 different Finite State Machines to represent individual connections.

We use the seed files from Appendix F, for each of the three agents of the ST-II protocol machine. We hypothesize that they identify all transitions of each agent. We did not have a tool to generate or verify the transition tours, consequently the value of the test is in the efficiency estimate rather than the exactness of the transition tour identification. The largest $N_T$, (Chapter 4), is for the intermediate agent, where $N_T = 9$, for ACK packets to DISCONNECT or REFUSE PDU-s. We generated 8 test suites, containing randomly selected one-connection environments with the upper bound on vertical recursion (number of targets) equal to 30. All transition tours from seed files were added to these sets in order to simulate test suites which have transition coverage. Generated test suites had 177, 277, 377, 577, 677, 777, 877 and 977 test cases, and are designated ts.$n$, where $n$ is the number of test cases they contained. Figure 6.6 (enlarged portion in Figure 6.7) shows the number of test cases selected by the progressive passes of the test selection
algorithm, for each of the test sets, as the density covers the range from 511 to 0.998047. At 0.998047 the algorithm is guaranteed to have identified transition cover of the three agents, when they work in isolation, provided the experiment's hypothesis is satisfied. The metric used is \( p_k = 2^{9-k} \) for \( k \) less than 10, and \( p_k = \frac{1}{2^{1-k}} \), for \( k \geq 10 \). Also, the recursion was given a minimal weight through the \( r \) function as in experiments 1.2 and 1.3.

The greater variety in event patterns and recursion is expected, as the randomly generated test sets grow larger. This accounts for more test cases selected, at equal density levels, for larger test sets.

The results show that, even with larger test suites, the algorithm is not overly wasteful, when used for identifying test suites with fault coverage at least equal to that of the transition tour method. Since the metric used in the experiment yields a complete metric space, the experiment simulates a test selection environment where, first, a test suite of transition tours equivalent coverage is selected, after which the process further proceeds by identifying still finer subsets of the initial set, with respect to variations in patterns of recursion and event sequencing. Completeness guarantees complete coverage of the initial set in the limit.

**Comparison of Pseudo-random Test Generation and Random Test Generation by MB Method**

**Experiment 3**

This experiment compares pseudo-random test generation, realized through refining randomly generated test sets by \( \epsilon \)-dense subsets, with random test generation. The goal is to investigate if metric-based test selection can help derive test sets with better coverage in a general case of single- and multi-connection environments, with light to moderate
Selecting Transition Tours with moderate vertical recursion

Figure 6.6
Selecting Transition Tours with moderate vertical recursion

Figure 6.7


<table>
<thead>
<tr>
<th>test set id</th>
<th>connections</th>
<th>targets</th>
<th>cases generated</th>
<th>cases selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>30</td>
<td>300</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>30</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>30</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>30</td>
<td>500</td>
<td>98</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>10</td>
<td>500</td>
<td>96</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>10</td>
<td>500</td>
<td>86</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>10</td>
<td>500</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>10</td>
<td>500</td>
<td>100</td>
</tr>
</tbody>
</table>

Experiment 3 - Characteristics of randomly generated sets
Table 6.1

2 series of test sets were generated, psrid and rid, id = 1, \ldots, 10. Sets labelled psr1,...,psr10 were obtained by randomly generating larger test sets (details in Table 6.1), and then refining them through test selection to obtain their most dense subsets containing approximately 100 test cases (Table 6.1, column 5). Sets labelled r1,...,r10 were obtained by randomly generating 100 test cases in each set, using the same bounds for simultaneous connections and number of targets as their psr counterparts with the same id. Sets with equal ids were compared for their mutual coverage, and results plotted in Figure 6.8. The metric used was the same as in the experiment 1.2. In Figure 6.8, x axis is labelled with id, id = 1, \ldots, 10. y axis represents the density of test sets rid in test sets psrid, and vice versa. Plots are in fact scatter graphs. We have connected points representing densities of rid in psrid sets by a line labelled psr/r, and points representing densities of psrid in rid sets by a line labelled r/psr, for easier viewing of the results.

For 9 out of 10 cases, psr sets were more dense in their r counterparts, than vice versa. Even with very modest effort and size of the starting set, the results show almost consistent
Pseudo-random versus random test generation

Figure 6.8
gain in coverage if randomly generated test suites are subjected to 1 or 2 iterations of the test selection algorithm.

Comparison of Pseudo-random Test Selection and Systematic Backtracking by MB Method

Experiment 4

This experiment compares test suites generated by pseudo-random test generation and systematic backtracking on the basis of the number of test cases selected at a particular density level. The pseudo-random test sets used in comparison were generated using parameters that favoured the backtracking algorithm. The metric function is the same as in the previous experiment.

First, a systematic backtracking algorithm within the TESTGEN++ system was run with certain ISP, OSP and PDU test generation constraints that produced 134 transition tours. Next, a random test generation algorithm was invoked on the same FSM (ORIGIN agent of ST-II), and run to produce 134 test cases, with 2 connections and 5 targets, which is roughly the characteristic of the test suite produced by TESTGEN. Figure 6.9 shows the results of the progressive steps of test selection algorithm, run on the sets \( rtest \) and \( btest \), produced by random and backtracking algorithms, respectively.

The experiment was repeated once more, for a more loose set of constraints on the backtracking algorithm which produced 456 test cases. The random test generator was run with 3 connections and 20 targets to produce 456 test cases. Similar results (not shown) were observed. For instance, a 3.5 dense subset of \( rtest \) contained 116 test cases, whereas a 3.5 dense subset of \( btest \) contained 48 test cases.

The informal inspection of test sets generated by the two methods reveals, that at these sizes of test sets, the systematic backtracking algorithm produced poorly structured test suites. (The first 12 test cases generated by the two algorithms can be found in Appendix F.) The suites of these sizes, produced by systematic backtracking, typically involved
Selected subset size for random and backtracking algorithms

Figure 6.9
subtours with very little variation in the initial tour segment, among different tours, and little or no vertical recursion. Another characteristics of this test suite was that if more than 1 connection is exercised in the same test case, it was after the previous one closed (i.e. no real simultaneity). The experiment indicates that the MB selection can be used to detect poorly structured test suites, through detecting low sizes of selected subsets at particular levels of density, for the levels of multiple connections and vertical recursion a test suite is known to exercise.

Sensitivity to Vertical Recursion

Experiment 5

In this experiment we investigate the effect of vertical recursion on the mutual density of test sets. We use test suites of 100 test cases each, randomly generated, with low numbers of simultaneous connections (1 - 3), and moderate vertical recursion (10 - 30) targets. The experiment shows that sets generated with a certain number of simultaneous connections and a certain event pattern are less dense in the sets with the same characteristics, if they contain less recursion than those sets.

Figure 6.10 shows this effect of vertical recursion on mutual coverage of two series of 6 test suites each. Sets $r_1, \ldots, r_6$ are randomly generated sets of test cases, and $s_1, \ldots, s_6$ are their corresponding sets (same number of simultaneous connections and event patterns), but with a certain reduction in recursion when calculated with respect to the corresponding set (i.e. the same subscript) from the $r$ series. The test suite generation specifics and the amount of reduction in recursion (in percentages) are shown in Table 6.2. No special effort was taken to fairly space the recursion levels in either $r$ or $s$ series of test sets.

The effect is plotted by representing the mutual density for each pair of sets $(r_k, s_k)$, $k = 1, \ldots, 6$ running along the x-axis. Connecting the points of the scatter graph allows for easier viewing. The density points of sets $s_1, \ldots, s_6$ in sets $r_1, \ldots, r_6$ are connected by the $r/s$ line, and the density points of sets $r_1, \ldots, r_6$ in sets $s_1, \ldots, s_6$ are connected by
6-3. **EXPERIMENTAL RESULTS**

<table>
<thead>
<tr>
<th>set</th>
<th>connections</th>
<th>targets</th>
<th>rec. reduction for sets $s_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>1</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>r2</td>
<td>2</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>r3</td>
<td>3</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>r4</td>
<td>1</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>r5</td>
<td>2</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>r6</td>
<td>3</td>
<td>20</td>
<td>70</td>
</tr>
</tbody>
</table>

Experiment 5 - Characteristics of test sets

| Table 6.2 |

The $s/r$ line. The metric function used is the same as in the previous experiment.

We generally found that with random generation, sets of 100 test cases were sufficient, under given protocol characteristics, for the mutual coverage figures to show sensitivity towards vertical recursion at reduction levels of in the range of 25-100 percent. This is due to the fact that, given a sufficient number of test cases in a test suite and a test generation algorithm that does not necessarily cluster recursive events in few clusters with poor distribution, lower bounds of recursion are more likely to generate less variety in event cluster recursion. Consequently, it will be easier for the set which has higher bounds on recursion to approximate event patterns and recursive levels of poorly recursive sets than vice versa, under such conditions.

In a related experiment (not shown in this thesis), pairs of randomly generated sets, with moderate numbers of simultaneous connections, one with low and one with high recursion, were compared for mutual coverage. Under the same metric function as in this experiment, the results were inconclusive. We identified the cause by examining sequences which contributed to high maximum distances. We concluded that a new event pattern in a test sequence, likely to appear early in a sequence with many simultaneous connections, could easily offset the effect of better recursion distribution under this metric.

Joint conclusion from both experiments is that the quality of test suite expressed through
CHAPTER 6. TESTS OF THE THEORY

Sensitivity to vertical recursion

Figure 6.10
metric based coverage functions is sensitive towards vertical recursion, however a metric with adequate sensitivity to recursion should be used if it is to be a decisive factor of the quality of test suite.

**Sensitivity to Parallelism**

**Experiment 6**

In this experiment we investigate the impact of testing with low and moderate network connection parallelism on the mutual coverage of test suites that differ with respect to the level of parallelism. The experiments were conducted under quite general circumstances. All test sets, 6 with low and 6 with moderate parallelism of network connections, were randomly generated, containing 100 test cases each. Therefore a certain potential existed for inconclusive results due to choosing a particularly rare combination of events in either case. However, the mutual coverage results were consistent in 3 different metric spaces and throughout 36 different comparisons. We therefore concluded that the results were a good indication of the MB coverage function to identify suites with greater simultaneity of network connections, in a general case. The results are reported in Figures 6.11, 6.12, 6.13.

The characteristics of individual test suites, with respect to the number of simultaneous connections and the number of targets are given in Table 6.3. Suites involved moderate vertical recursion (10-30 targets), and were put into "low simultaneous connection" (lscid) set if involving 1 to 3 simultaneous connections. Likewise, "moderate simultaneous connection" sets (mscid) involved 10 - 35 simultaneous connections. Suites with equal id-s were compared.

The comparisons were carried out in metric spaces generated by metrics with

1. $p_k = 1$, for $k < 400$, and $p_k = \frac{1}{2^{k-400}}$, for $k \geq 400$.

2. $p_k = 1$, for $k < 25$, and $p_k = \frac{1}{2^{k-25}}$, for $k \geq 25$. 
### Experiment 6 - Characteristics of randomly generated test sets

<table>
<thead>
<tr>
<th>test set id</th>
<th>connections</th>
<th>targets</th>
<th>cases generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>msc0</td>
<td>10</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>lsc0</td>
<td>1</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>msc1</td>
<td>15</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>lsc1</td>
<td>1</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>msc2</td>
<td>20</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>lsc2</td>
<td>2</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>msc3</td>
<td>25</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>lsc3</td>
<td>3</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>msc4</td>
<td>30</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>lsc4</td>
<td>1</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>msc5</td>
<td>35</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>lsc5</td>
<td>2</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

3. \( p_k = 1 \), for \( k < 15 \), and \( p_k = \frac{1}{2^{k - 15}} \), for \( k \geq 15 \).

and the series \( r_k \) were defined as in Experiment 1.2.

The plots for each of these metric spaces are in Figures 6.11, 6.12, 6.13, respectively. x axis is labelled with set id-s, from 0 through 6. The plots show the density of sets \( lsc0, \ldots, lsc5 \), in sets \( msc0, \ldots, msc5 \), resp. These scatter points are connected by a line labelled msc/lsc. Similarly, the line labelled lsc/msc connects points that show densities of sets \( msc0, \ldots, msc5 \) in sets \( lsc0, \ldots, lsc5 \), resp.

Although the test sets with moderate parallelism are consistently more dense in low-parallelism sets in all three metric spaces considered, significant difference exists in the magnitude of the mutual density variation. Metric 1. evaluates all patterns of very long sequences with equal value, up to a large \( k = 400 \). Therefore, the density variation in the space generated by this metric can be attributed to the much larger length of sequences involving many connections. This, however, is a definite indication of the number of connections exercised, given fixed vertical recursion, but not necessarily of the simultaneity of
Mutual densities of some sets with moderate and low number of simultaneous connections

Figure 6.11
Mutual densities of some sets with moderate and low number of simultaneous connections

Figure 6.12
Mutual densities of some sets with moderate and low number of simultaneous connections

Figure 6.13
such connections. Therefore, the next two metric spaces favoured the average and upper end case length of sets \( lscid \), which was found to be in the range of 11 - 36. (Lengths of \( mscid \) test cases were in the range of 60 - 145.) Consequently, they really evaluated the greater variation in pattern of test sequences generated by many simultaneous connections, especially in view of the fact that all sequences longer than 15 (25) would contribute at most 1 to the final distance. This is an extreme case which definitely did not give \textit{a priori} advantage to longer test sequences of sets \( mscid \). It is therefore conceivable that also a more moderate metric would yield results which are consistent with this experiment's results.

**Sensitivity to a Missing Event or Event Combination**

The purpose of the next two experiments is to investigate the sensitivity of mutual densities of two test suites, when one of the test suites does not contain an event of event combination, or does so only to a certain lesser extent. All test sets contain 100 test cases. Metric is the same as in Experiment 1.2, unless otherwise noted.

**Experiment 7.1**

First, the sensitivity analysis was carried out for the environments involving no parallelism and moderate vertical recursion. In this experiment, 20 randomly generated sets of 100 test cases each were compared for mutual coverage. Test sets \( s_1 \) through \( s_{10} \) are random interleavings of one connection, up to 30 ST-II targets. Test sets \( o_1 \) through \( o_{10} \) are random interleavings of one connection, up to 30 ST-II targets, where ST-II agent origin does not include sequences with a PDU HAO (HID APPROVE) as a response to a CN (CONNECT) PDU. Figure 6.14 shows the density deviation for the densities of sets \( o_k \) in randomly generated test sets \( s_k \), connected by the line \( s/o \). For comparison, also shown are plots of all other density figures available in the experiments: a) densities of sets \( s_k \) in sets \( o_k \), connected by a line \( o/s \), b) densities of sets \( s_k \) in sets \( s_{k+1} \), connected by a line
6-3. EXPERIMENTAL RESULTS

s/s, and c) densities an oracle set originoracle in sets \( O_k \), line o/originoracle.

![Graph](image)

Sensitivity to missing events - single connection with recursion

Figure 6.14

**Experiment 7.2**

In this experiment we investigated environments with moderate parallelism and moderate vertical recursion. This experiment shows the effects of a missing event (SP or PDU) in a more complex test suite. 3 groups of test sets, 100 random test cases each, were generated to randomly contain between 5 to 10 simultaneous connections, and 10 to 30
targets. The set $s_1$ through $s_6$ was generated taking into account all available SPs and PDU for all ST-II agents. Sets $r_1$ through $r_6$ were generated with DSI (DISCONNECT PDU from the direction of origin) missing from intermediate seed sets. Likewise, sets $t_1$ through $t_6$ were generated with HCT PDU missing from target agent seed sets.

Figure 6.15 shows that applying general metric to mutual coverage of such sets yields inconclusive results. Lines $s/r$, $r/s$, $s/t$ and $t/s$ connect scatter points of densities of sets $r$, $s$, $t$, and $s$ in sets $s$, $r$, $s$, and $t$ resp., where densities are calculated only between sets with equal subscripts.

We did observe, however, that at these levels of parallelism and recursion, the mutual coverage of randomly generated test sets with all events included into seed files, consistently showed better density figures.

In a related experiment, we improved sets $t$ from the previous experiment, by adding 10 percent and 70-80 percent of sequences, involving the HCT PDU. Figure 6.16 shows the range of densities of sets $t_1, \ldots, t_6$ (all HCT event sequences deleted), $t_{1\text{aehct}}, \ldots, t_{6\text{aehct}}$ with almost all HCT events missing (10 percent sequences with it), and $s_1, \ldots, s_6$ with 70-80 percent sequences with HCTs (which is the typical content in a randomly generated test set of these characteristics), in randomly generated test sets $rs_1, \ldots, rs_6$, applying a general metric function (from Experiment 1.2).

Although the results explain the densities in Figure 6.15 to a certain extent, it still would be necessary to have a knowledge of the expected densities at particular levels of vertical recursion and parallelism, in order to use general metric functions for missing event identification.

In the following experiments, a special metric function was designed to identify the coverage of a particular event (HCT PDU in this case). Its definition is the same as in Experiment 1.2., except for the cases were neither of the events at a position $k$ in two sequences is HCT: if this is the case, the summand $p_k \delta_k$ for this position $k$ is zero. This metric function measures the coverage of the event HCT both with respect to its position.
Sensitivity to missing events - general metric function, multiple connections with recursion

Figure 6.15
Sensitivity to missing events - general metric function, moderate parallelism and recursion

Figure 6.16
6-3. EXPERIMENTAL RESULTS

in sequences and the level of recursion.

Figure 6.17 shows the application of the metric function to specially identify the coverage of HCT event, and its improvement with improving t-test sets with sequences containing event HCT, up to its average 70-80 percent representativeness as observed in randomly generated test sets. These are calculated as densities of t sets in randomly generated s sets. Figure 6.18 shows the effect of improving sets \( t_1 \) to \( t_6 \) with 10, 20, 50 and 70-80 percent of sequences which contain HCT in case of general metric. These are calculated...
as densities of $t$ sets in randomly generated $s$ sets. These do not improve as fast as in the previous metric.

All subsequent examples use the special metric designed to identify missing event content.

The following figures show the sensitivity of this metric in various situations.

Figure 6.19 shows the gain in the quality of $t$ by increasing its HCT content, by plotting the density of randomly generated test suites, in sets $trd$, whose special event content
(HCT content) is improved by adding 10–70 percent test sequences involving the event considered. The next 2 figures show how the plots for 2 combinations of randomly generated test sets s and sets t from this same example, meet in the range of 16-64, as the coverage of the event HCT increases. (These are excerpts for fixed combinations of test sets from Figures 6.17 and 6.20.) Figure 6.22 shows mutual coverages of the randomly generated sets and set combinations in the same metric. Sets typically include 70-80 percent tests with HCT. Random sets were generated using 5-10 connections, 10-30 targets.
Sensitivity to missing events - special metric, moderate parallelism and recursion

Figure 6.20
6-3. EXPERIMENTAL RESULTS

Sensitivity to missing events - special metric, moderate parallelism and recursion

Figure 6.21
Experiments show that in such cases, mutual coverages of randomly generated sets also tend to settle in this same range.

Sensitivity to missing events - special metric on random sets

Figure 6.22
6-4 Discussion

The sensitivity analysis of the MB-method in the previous section is intended to provide only a rough assessment of its ability to identify or select test suites with certain levels of vertical recursion, multiconnection capabilities and event, transition or event pattern content. While we have endeavored to give a reasonably complete assessment of the sensitivity of the method with respect to these protocol features, there are at least two additional factors that may influence the validity of the results and the suitability of the algorithm implementation in a general case.

The first has to do with the complexity of the state space we have considered in the experiments. The spaces we have experimented with typically include 5 - 20 simultaneous streams (horizontal recursion complexity), and 10 - 30 target applications (vertical recursion complexity). For more complex state spaces (involving hundreds of streams and target agents), an independent set of experimental results should be obtained.

Secondly, one may argue that the sampling basis of the experiments (in the range of 10 - 30 sample test suites or comparisons) is not statistically significant to draw any conclusion about the stability of the sensitivity results. However, since results of these experiments are also analysed and supported by the theoretical explanations wherever possible, we feel that they may be considered as an indication that, under similar protocol and experimental setting, repeated results would tend to settle in the same range.

Based on these experiments, MB-method appears attractive for test selection and coverage evaluation in a general case of a relatively complex protocol, and moderately complex state space. Since no expert knowledge is required for the successful application of the method (except, perhaps, the knowledge of $N_S$ or $N_T$), then it is feasible for a single implementation to be utilized over a range of protocols - greatly improving test

---

For a typical test case, this may mean up to 20\*30 simultaneous network connections from source to destination Service Access Point.
development time compared to specialized protocol test generators.

It is useful to discuss the efficiency of MB-method in terms of the size and complexity of the state space it can successfully handle. As discussed in [36, 43, 26], known test generation and coverage analysis tools do not scale well to larger "programs". It is not known to us that there exists another implementation, that can systematically select and analyse test suites with the level of concurrency and recursion that we have used in these experiments.

In contrast, MB-method handled "moderately" complex state spaces reasonably well, with performance results settling in the range of Section 6-3.1. The process degraded gracefully as the number of simultaneous connections and vertical recursion were allowed to rise to levels considered in experiments. Although neither (static) test selection or coverage evaluation are time critical processes, the experiments showed a need for better algorithm and implementation, if larger space analysis is to be attempted. This is especially true if test suites larger than 1000 test cases in moderately complex state spaces are to be mutually compared for coverage.

The performance of the method was greatly improved in the experiments when input events only were considered. This is however applicable only in cases where interoperability relation is the theoretical upper bound on the testing process. In contrast, all experiments were carried out at the level of granularity of an atomic event. The metric-based theory does not constrain us to such granularity - a mixed level hierarchical test selection and analysis is equally supported by the theoretical results. It is expected that the algorithm would be more efficient for coarser granularity of test subsequences or sets of subsequences, or at module level.

The efficiency of the MB-method could be improved somewhat further by the following

---

5 We characterize state spaces of 10 - 600 simultaneous network connections as moderately complex, as compared to "low" complexity test suites generated by typical systematic methods [16], which contain 1 - 5 connections.
Partitioning of the input sets into disjunct regions (Voronoi diagrams were considered in the early algorithm implementation), processing the regions separately and then putting the results together, would possibly alleviate the performance problem related to large numbers of initial test cases.

Computing the distance only to a certain depth of the execution sequence, using the knowledge of the maximum sum of the tail of the series, would shorten the time of individual sequence comparison in spaces with increased recursion and simultaneity of connections.

All computations in the present implementation were carried out in floating point mode. Scaling the values for $p_k$ and $r_k$ to large integers, and carrying out the computation in integer mode has been observed to noticeably shorten the time needed for individual algorithm passes.
Chapter 7

Conclusion

In this concluding chapter we summarize the contributions reported in the previous chapters, and suggest some topics for further research.

7-1 Summary of Contributions

7-1.1 A Metric Characterization of Protocol Control Behaviour Space

Metric characterizations of the behaviour of systems have been proposed earlier ([7, 18]) for theoretical investigations of some aspects of systems' behaviour in the theory of concurrency. However, metrics used in earlier work were typically the simple metrics on strings (da, Appendix C, Def. C.2). This metric distinguishes strings of events with respect to the first event at which they differ in a temporally ordered sequence of events of a system’s control behaviour.

Large (and infinite) systems are ordinarily not specified (or cannot be directly specified) in terms of strings of events. Specification formalisms use hierarchical representation
of such systems, in which recursively specified objects may generate large and infinite control behaviour spaces of strings of events. Analysis of the behaviour of such systems is not possible, in the previously researched metric, at different levels of hierarchy that specification mechanisms allow.

This thesis gives an improved metric characterization of the control behaviour space of such systems. The metric $dt$ defined in this thesis includes recursive characteristics of strings of events into its definition. This allows analysis of (infinite) systems which are specified hierarchically, with recursion, at various levels of granularity with respect to such a hierarchy: atomic actions, sequences of actions and their sets, processes (modules) and groups of processes. Being primarily concerned with the practical aspects of the approach, we stop short of expanding these results to "any level of granularity" with respect to the specification hierarchy. In a series of results we show the theoretical validity of the new metric characterization of the control behaviour of systems. The results on completeness (Theorem 3.3), total boundedness (Theorem 3.2), the existence of finite approximants to infinite sequences of interactions (Theorem 3.6), and the metric characterization of trace preorder and equivalence (Propositions 4.5 and 4.6), derived in this thesis, are comparable to those obtained earlier for simpler metric spaces.

7-1.2 Method-independent Coverage Evaluation

A simple method for measuring mutual coverage of the control behaviour of test suites has been defined based on the metric $dt$. This method of test coverage evaluation has several advantages over other test coverage evaluation methods.

First, the metric based coverage measures are defined for infinite behaviour spaces of systems (Def. 3.4. together with the total boundness result provide theoretical foundation for a coverage evaluation method in such spaces). With respect to other coverage criteria

---

1 The constraint is contained in Theorems 3.7 and 3.8, where only finite sets are allowed to be extracted.
defined for infinite test domains, e.g. number of paths covered (which is inconclusive for recursive systems), or the all paths criteria (which is practically infeasible), this definition is both theoretically sound for recursive systems (the coverage of low, high, and thorough coverage of recursion in between the extremes is evaluated, Chapter 3.2.2), and provides theoretical basis for coverage evaluation methods for large and infinite metric spaces which are totally bounded (through the stratification of infinite spaces into finite intermediate sets of truncations of traces, Definition 3.3, and 3.4).

Secondly, the definition of coverage is analytical in character. It does not involve any testing concept (e.g. mutants, transitions or branches) or non-specified information (e.g. valuations or expert opinion). The generality of the method resulting from its analytical character is further demonstrated by establishing the metric hierarchy of some seemingly unrelated fault coverage models in protocol testing: FSM methods with transition and transfer fault coverage, general software test methods with branch and path coverage, and finally trace equivalence as a fault model for infinite labelled transition systems (Chapter 4).

Next, the definition of MB coverage is applicable to mutual coverage evaluation of test suites derived by different test methods. Whereas any method can be (either theoretically or experimentally) used to evaluate another, there are two original features of this method that make it a fair comparison tool.

The fact that it is defined on the entire space of a specified behaviour in atomic terms (i.e. unconstrained by particular requirements of some method), and that it evaluates coverage on the basis of generic behaviour information, make it an unbiased method for any underlying test suite structure. In this way, a frequent error [24], of judging one test method inside the space generated by the constraints of another method, is avoided.

Also, the method is not based on a relationship in which one set of test cases "subsumes" another in terms of coverage (e.g. maximum depth of paths covered in depth bound methods, or inclusion of test sets ([4, 20])). Subsumes relationship is both inap-

---

2We consider that a system is fully specified by a valid (formal) specification of its behaviour.
plicable to non-deterministic systems (a subset of test cases on an execution may expose a fault that a superset did not), and practically obscure (for depth bound methods on recursive systems it forces counterintuitive test preference, whereas the set inclusion criteria is bound to yield inconclusive results most of the time). The MB coverage method is both always computable, and avoids the caveat of enforcing the subsumes relationship in test coverage. This is further underscored by the fact that test sets may be compared in a family of metric spaces parameterized by functions $p_k$ and $r_k$, (Chapter 3.2B, also Chapter 6, Experimental Results), and with respect to two different coverage measures, $CovMax(T)$ and $CovAve(T)$ (Chapter 3.3). It is therefore geared towards examining the quality of test suites, rather than passing a definitive judgement on coverage in large nondeterministic environments.

7-1.3 The MB Test Selection Method

The metric based test selection method for the control behaviour of protocol behaviour, presented in this thesis, has several features that set it apart from other test selection algorithms.

First, the method is equipped with both the coverage target (all finite and infinite sequences, in metric spaces $S$, $S_{(s)}$, and $S_{(s)}$, Theorems 3.7 and 3.8), and the testing relation (which may be strings equivalence, $conf$ (Chapter 3.4.2), or $intop$, $intop_{red}$, (Chapter 5.4)). The testing relation will hold between the tested and specified systems, after the completion of the testing process based on selected test cases, if the coverage target is achieved.

This original way of looking at the testing process is theoretically sound. The iterative character of test selection algorithms that metric spaces permit, through achieving finer $\epsilon$-dense approximations of original sets in a series of steps, provides for their practical feasibility for large systems. The thesis also gives a theoretical basis for the feasibility of
the method for infinite systems. The proof of the existence of finite $\epsilon$-dense covers for
infinite systems in Theorem 3.2, which offers a computation method for discarding all
but a finite number of elements of infinite spaces that should participate in any specific
pass of the algorithm, together with the systematic Cauchy sequence generation by the
algorithm itself, show that this indeed is possible. The ability to deal with infinite tests
and infinite spaces in a systematic, intuitive and theoretically sound manner is a unique
feature of the MB method.

This method is entirely based on the information contained in the description of the
system, and can therefore be fully automated. Full automation is crucial if thorough
testing of large systems is the goal, and improves the practice of software engineering by
shortening the software development time.

7-1.4 Interoperability Relation

Current testing theory of nondeterministic, concurrent and communicating systems in-
volves relational characterizations of how well one system implements a valid specifi-
cation of its behaviour. The characterizations are based on the observation of system’s
behaviour. The underlying notion is that, if two systems cannot be told apart by ob-
serving their behaviour, then they cannot be told apart by testing the same behaviour.
Therefore, they are testing equivalent. One system is then, in particular, "less" than the
other with respect to a testing relation, if fewer traces of identified kind can be observed on
its interaction with environment (tester). The only possible exception are traces resulting
from internal nondeterministic choices, and those related to the treatment of divergence.

In the definition of the interoperability relations $\text{in}top$ and $\text{in}top_{red}$, we challenge this
basic premise of the theory of testing in the interleaving model of concurrency. In exam-
inig specifications of real systems in environments more complex than usually considered
in theoretical investigations, we have determined, that a different kind of testing relation
would be more appropriate for testing of communication protocols. The testing of pro-
cesses with the theoretical upper bound of intop relation is still based on observation, but the possibility of observing and testing a behaviour are not seen as necessarily equivalent notions. In particular, the possibility that a trace may be observed (by specification of a system) does not necessarily imply that it may be observed in testing (i.e. we allow that it may not). The intop relation allows the definition of sets of events ($L_{alt}$ and Def. 5.2-2), and refusal sets (Def. 5.2-3), such that the observation of one behaviour in a set of alternative behaviours is sufficient to establish the validity of a system and its specification with respect to intop relation. This, in particular, may mean that the two systems are not even trace equivalent. The intop relation is a more accurate definition of validity of an implementation for testing multiconnection protocol environments for interoperability. Previously defined testing relations based on observation require that such behaviours be also observable in testing, even if they are irrelevant for interoperability. At the same time, it is a more efficient upper bound on the testing process. It eliminates inconclusive test runs due to non-synchronized next event selection between the tester and implementation under test, and redundant test runs of alternative behaviours which may even be unobservable. It supports fair selection strategies in that possibly unobservable and behaviours irrelevant for interoperability are uniquely identified in the interoperability tester tree, and may be dropped in favour of required behaviours which are essential for interoperability.

7-1.5 Contributions to Specific Areas of Study

The results of this thesis are relevant to several areas of study.

The new metric characterization of the behaviour of concurrent systems allows for the same theoretical investigations in the theory of concurrency to be carried out in a more finely characterized space. Issues such as fair and unfair behaviours, divergence and equivalences can be reformulated to allow for the level of detail currently not possible.

The MB theory can be used as a theoretical basis of algorithms that may allow system-
atic selection, analysis and possibly generation of test suites with the levels of recursion and parallelism usually not attempted in practice, thereby improving the practice of protocol engineering.

The theory of extensional equivalences in process algebras with interleaving semantics of concurrency, is challenged in this work with respect to the predominant view of observation and testing being equivalent notions.

Finally, the research in this thesis has been inspired by the related work in software testing. Whereas the related work states the superiority and need for coverage testing (of complete state spaces) in a "uniform" manner (i.e. no a priori preferred test cases), it does not discuss a method, other than stating that one is presently unavailable [25]. We believe that this thesis is a step towards a possible solution to this general problem and that it does bring the theory and practice of software testing that much closer.

7-2 Suggestions for Further Research

A number of interesting research problems arise in connection with the results derived in this thesis.

A challenging and long standing problem in software research [53] is to relate test coverage to the reliability of tested implementations. We believe that this problem is solvable within our proposed analytical framework.

In the thesis we stop short of repeating the work of [18] in demonstrating connections of the two theories of concurrency: metric spaces and synchronization trees, for the metric spaces \((S, dt)\). However, the results in Chapter 3, Section 4.3 are a good indication that this may be possible. This would be of relevance to protocol testing, if efficient methods are found for execution of tree-specified test sets.

The experimentation experience suggests that the MB method may also be used as
a test generation method, perhaps on infinite sets defined by specification. A pseudo-random test generation of $\varepsilon$-dense sets in complete spaces is one possible solution that deserves investigation. Another possibility is to generate $\varepsilon$-dense sets using finite descriptions of infinite systems in the style of [17], and the constraints derived in the proof of the total boundness theorem.

Experimental extension of the work of this thesis in the direction of experimental calibration of MB method against some well known testing tools, on very large systems, would be of definite interest for the comparison of this method to other established test methods.

It may be interesting to use this metric function (or other metric functions), for the analysis of the specified control behaviour space of protocols. Using metric functions to investigate and possibly characterize protocol features such as testability, distribution of events in the metric space, error detection probability of faults, hard to detect faults and other issues may help improve the efficiency and quality of the testing process of communication protocols.
Bibliography


Appendix A

Labelled Transition Systems

Definition A.1 ([12]) A Labelled Transition System is a quadrupole \((Q, E, -a \rightarrow, q_0)\) where \(Q\) is a countable set of states, \(E\) is a countable set of elementary actions, \(-a \rightarrow\) with \(a \in E \cup \{i\}\), is a set of binary relations on \(Q\) and \(q_0 \in Q\) is the initial state. □

Notation A.1 ([39])

Processes (denoted by \(P\), and ranged over \(T_1, T_2, T_1, T_2, P_1, P_2, P', \ldots\)) will be sets of labelled transition systems over an alphabet \(E \cup \{i\}\) (\(i\) is the unobservable action) of elementary actions, ranged over by \(a, b, c, \ldots\).

\(P - a \rightarrow P'\) means that process \(P\) may engage in an action \(a \in E\) and, after doing so, behaves like process \(P'\).

\(P - i^k \rightarrow P'\) means that process \(P\) may engage in the sequence of \(k\) internal actions and, after doing so, behaves like process \(P'\).

\(P - a.b \rightarrow P'\) means \(\exists P'',\) such that \(P - a \rightarrow P''\) and \(P'' - b \rightarrow P'\)

\(P = a \Rightarrow P'\) means \(\exists k_0, k_1 \in \mathbb{N}\) such that \(P - i^{k_0}.a.i^{k_1} \rightarrow P'\)

\(P = a \Rightarrow\) means \(\exists P'\) such that \(P = a \Rightarrow P'\), i.e. \(P\) may engage in an action \(a\)

\(P \neq a \Rightarrow\) means \(\neg(P = a \Rightarrow)\) i.e. \(P\) cannot engage in an action \(a\)

\(P = \sigma \Rightarrow P'\) means that process \(P\) may engage in a sequence of observable actions
\[\sigma\] and, after doing so, behaves like process \(P'\).

\(P = \sigma \Rightarrow\) means that \(\exists P'\) such that \(P = \sigma \Rightarrow P'\)

\(P = a^\infty \Rightarrow\) means \(\forall k\), we have \(P = a^k \Rightarrow\)

\(Tr(P)\) is the trace set of \(P\), i.e. \(\{\sigma \mid P = \sigma \Rightarrow \}\); \(Tr(P)\) is a subset of \(E^*\)

\(P \text{ after } \sigma = \{P' \mid P = \sigma \Rightarrow P'\}\), i.e. the set of all behaviour expressions (or states) reachable by \(\sigma\)

\(Out(P, \sigma)\) is the set of possible observable actions after the trace \(\sigma\), i.e.

\[Out(P, \sigma) = \{a \in E \mid \sigma.a \in Tr(P)\}\]

\(Ref(P, \sigma)\) is the refusal set of \(P\) after trace \(\sigma\), i.e.

\[Ref(P, \sigma) = \{X \subseteq E \mid \exists P' \in P \text{ after } \sigma, \text{ such that } P' \neq a \Rightarrow, \forall a \in X\}\]
Appendix B

LOTOS

LOTOS (Language of Temporal Ordering Specification) is a Formal Description Technique developed within ISO (International Organization for Standardization) for the formal specification of open distributed systems. The systems are specified in LOTOS by defining a temporal relation among the interactions, that constitute the externally observable behaviour of the system. Labelled Transition Systems (Appendix A) are used as models for LOTOS processes.

The operational semantics is given by a system of inference rules that correspond to the language operators. These rules generate a Labelled Transition System for each behaviour expression of the language. Let $B_1, B'_1, B_2, B'_2$ represent behaviour expressions, and $m \in E$ observable actions. The special action $i$ represents the unobservable, internal action that can be used to model nondeterminism.

The purpose of this annex is just to present briefly the mapping of a LOTOS specification onto such model. We give axioms and inference rules for the subset of LOTOS that is used in the thesis in Table B.1. A detailed presentation may be found in [31].
<table>
<thead>
<tr>
<th>Combinator</th>
<th>Axioms or Inference Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop</td>
<td>none</td>
</tr>
<tr>
<td>m; B</td>
<td>m; B - m → B</td>
</tr>
<tr>
<td>i; B</td>
<td>i; B - i → B</td>
</tr>
<tr>
<td>B₁</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- B₁</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>B₂ - m → B₂', name(m) ∉ {g₁, ..., gₙ}</td>
</tr>
<tr>
<td></td>
<td>- B₁</td>
</tr>
<tr>
<td>B₁</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B₁</td>
</tr>
<tr>
<td>B₁</td>
<td></td>
</tr>
</tbody>
</table>

Axioms and Inference Rules in LOTOS

Table B.1
Appendix C

Theory of Metric Spaces - Mathematical Definitions

Definition C.1 (Metric space)
A metric space is a pair \((M, d)\) with \(M\) a non-empty set and \(d\) a mapping (a metric or distance) of \(M \times M\) onto the set of non-negative real numbers, that satisfies the following properties:

(a) \(\forall x, y \in M\) \(d(x, y) = 0 \iff x = y\)
(b) \(\forall x, y \in M\) \(d(x, y) = d(y, x)\)
(c) \(\forall x, y, z \in M\) \(d(x, y) \leq d(x, z) + d(z, y)\).

Definition C.2 Let \(A\) be an alphabet, and let \(A^\infty\) denote the set of all finite and infinite words over \(A\). Let, for \(x \in A^\infty\), \(x(n)\) denote the prefix of \(x\) of length \(n\), in case \(\text{length}(x) \geq n\), and \(x\) otherwise. We put

\[
\delta_a(x, y) = 2^{-\sup\{n|\text{x(n)}=\text{y(n)}\}}
\]
with the convention that $2^{-\infty} = 0$. Then $(A^\infty, da)$ is a metric space.

**Definition C.3** Let $(M, d)$ be a metric space, let $\{x_i\}_{i=0}^\infty$ be a sequence in $M$.

(a) We say that $\{x_i\}_{i=0}^\infty$ is a Cauchy sequence whenever we have:

$$\forall \epsilon > 0 \ \exists N \in \mathbb{N} \ \forall n, m > N \ d(x_n, x_m) < \epsilon$$

(b) Let $x \in M$. We say that $\{x_i\}_i$ converges to $x$ and call $x$ the limit of $\{x_i\}_i$ whenever we have:

$$\forall \epsilon > 0 \ \exists N \in \mathbb{N} \ \forall n > N \ d(x, x_n) < \epsilon$$

Such a sequence we call convergent. Notation: $\lim_{i \to \infty} x_i = x$.

(c) The metric space $(M, d)$ is complete whenever each Cauchy sequence converges to an element of $M$.

Let $(M, dt)$ be a metric space and $T, S$ be two subsets of that metric space.

**Definition C.4** A family $\mathcal{F}$ of subsets of $M$ is said to be a covering for $S$ if $S \subseteq \bigcup \{F : F \in \mathcal{F}\}$, i.e. if $S$ is contained in the union of all sets in $\mathcal{F}$.

**Definition C.5** A set $T$ is said to be $\epsilon$-dense in $S$ (for $\epsilon > 0$) if for every $s \in S$ there is $t \in T$ such that $d(s, t) < \epsilon$.

**Definition C.6** A metric space $S$ is totally bounded if for every $\epsilon > 0$ it is possible to cover $S$ by a finite number of spheres with radius $\epsilon$.

**Definition C.7** Let $T$ be a subset of a metric space $S$. Distance of a point $x \in S$ from $T$ is defined as

$$d(x, T) = \inf\{d(x, y) : y \in T\}$$
Definition C.8 The Hausdorff distance $d_H$ [28], between two subsets $S, T$ of a metric space is defined as follows:

$$d_H(S, T) = \max\{\sup\{d(s, T) : s \in S\}, \sup\{d(t, S) : t \in T\}\}$$

Definition C.9 A sequence $\{x_n\}_{n=1}^{\infty}$ is said to be eventually constant if there exists $N \in \mathbb{N}$ such that $x_n = x_m \ \forall n, m > N$. 
Appendix D

A Formal Description of ISO 8072 in LOTOS

\[
\begin{align*}
&\text{behaviour} \\
&T\text{Connections} \ [t] \\
&\quad \text{||} \\
&T\text{CIdentification} \ [t] \\
&\quad \text{||} \\
&T\text{CAcceptance} \ [t] \\
&\quad \text{||} \\
&T\text{Backpressure} \ [t] \\
&\text{where} \\
&\text{process} \quad T\text{Connections} \ [t] : \quad \text{noexit} \\
&\quad \quad \quad \text{:=} \\
&T\text{Connection} \ [t] \\
&\quad \quad \quad \text{||} \\
&T\text{Connections} \ [t] \\
&\text{endproc} \quad (* \ T\text{Connections} *) \\
&\text{process} \quad T\text{Connection} \ [t] : \quad \text{exit} \\
&\quad \quad \quad \text{:=} \\
&(T\text{CEP} \ [t] \ (\text{CallingRole}) \ ||| \ T\text{CEP} \ [t] \ (\text{CalledRole})) \\
&\quad \quad \quad \text{||} \\
&(T\text{CEPAssociation} \ [t] \quad [> \text{exit} \ ) \\
&\text{endproc} \quad (* \ T\text{Connection} *) \\
&\text{process} \quad T\text{CEP} \ [t] \ (\text{role} : \text{TSUserRole}) : \quad \text{exit} \\
&\quad \quad \quad \text{:=} \\
&T\text{CEPAddress} \ [t] \\
&\quad \quad \quad \text{||} \\
&T\text{CIdentification} \ [t] \\
&\quad \quad \quad \text{||} \\
&T\text{CEPSPOrdering} \ [t] \ (\text{role}) \\
&\text{endproc} \quad (* \ T\text{CEP} *)
\end{align*}
\]
Appendix D. A Formal Description of ISO 8072 in LOTOS

process TCEPSPOrdering [] (role:TSUserRole) :exit :=
TCEPConnect1(role) >> accept tsp:TSP in
(TCEPConnect2(tsp)) >> accept x:TEXOption in
TCEPDataTransfer [] (x)
) [- TCEPRelease []
)
[role = CalledRole] -> exit
endproc (* TCEPSPOrdering *)

process TCEPConnect1[] (role:TSUserRole) :exit :=
TCEPConnect2[] (tc1:TSP) :exit :=

TCEPConnect2[] (tc1:TSP) :exit :=
[tc2 IsValidTCON2For tc1];
( choice x:TEXOption [] [x IsTEXOptionOf tc2] -> exit (x )
endproc (* TCEPConnect2 *)

process TCEPDataTransfer [] (x:TEXOption) :noexit :=
TCEPNormalDataTransfer []
[[x = UseTEX] -> TCEP ExpeditedDataTransfer []
endproc (* TCEPDataTransfer *)

process TCEPNormalDataTransfer []: noexit :=
TCEPRelease []: exit :=
endproc (* TCEPRelease *)
Appendix E

A Specification of ST-II Protocol in Estelle-Y

Origin Agent

Specification ST2ORIGIN;
CONST
VAR
ISP
    SOO ST2OSAP; /* STREAM OPEN */
    SCO ST2OSAP; /* STREAM CLOSE */
    SDO ST2OSAP; /* STREAM DISCONNECT */
OSP
    AIO ST2OSAP; /* APPLICATION INFORM MESSAGE */
PDU
    CNO recv_in SOO; /* CONNECT */
    DSO recv_in SCO; /* DISCONNECT */
ACO sent_in AIO;  /* ACCEPT */
RFO sent_in AIO;  /* REFUSE */
EQO sent_in AIO;  /* ERROR IN REQUEST */
ACO ST2OPDU;    /* ACCEPT */
AKO ST2OPDU;    /* ACKNOWLEDGE */
DTO ST2OPDU;    /* DATA */
HAO ST2OPDU;    /* HID APPROVE */
HRO ST2OPDU;    /* HID REJECT */
HCO ST2OPDU;    /* HID CHANGE */

TIMER

STATE

    idle, wfst, hidc, hida, cona, dtph, wfcl;

INITIALIZATION

TO idle
BEGIN
END;
TRANS
from idle
to wfst
    when SOO
       OUTPUT CNO;

TRANS
from wfst
to idle
    when RFO
       OUTPUT AKO, AIO;
to idle
when EQO
    OUTPUT AKO,AIO;
  to idle
    when SCO;
  to hidc
    when HRO
    OUTPUT HCO;
  to hida
    when HAO;

TRANS
  from  hidc
  to  hidc
    when HRO
    OUTPUT HCO;
  to hida
    when HAO;
  to idle
    when SCO;

TRANS
  from  hida
  to  hida
    when ACO
    OUTPUT AKO;
  to  dtph
    when ACO
    OUTPUT AKO,AIO;
  to  hida
    when RFO
OUTPUT AKO;
  to dtph
    when RFO
    OUTPUT AKO,AIO;
  to wfcl
    when SCO
    OUTPUT DSO;
TRANS
  from dtph
    to dtph
      when SDO
      OUTPUT DTO;
    to wfcl
      when SCO
      OUTPUT DSO;
    to idle
      when RFO
      OUTPUT AKO,AIO;
TRANS
  from wfcl
    to idle
      when AKO;
end.
Intermediate Agent

Specification ST2INTERM;

CONST
VAR
ISP
OSP

AKH ST2DTSAP; /* ACKNOWLEDGE (outgoing) */
HAH ST2DTSAP; /* HID APPROVE (outgoing) */

PDU

CNI ST2IPDU; /* CONNECT (incoming) */
CNH ST2IPDU; /* CONNECT (outgoing) */
DSI ST2IPDU; /* DISCONNECT (incoming) */
DSH ST2IPDU; /* DISCONNECT (outgoing) */
ACI ST2IPDU; /* ACCEPT (incoming) */
ACH ST2IPDU; /* ACCEPT (outgoing) */
RFI ST2IPDU; /* REFUSE (incoming) */
RFH ST2IPDU; /* REFUSE (outgoing) */
DTI ST2IPDU; /* DATA (incoming) */
DTH ST2IPDU; /* DATA (outgoing) */
AKI ST2IPDU; /* ACKNOWLEDGE (incoming) */
HAI ST2IPDU; /* HID APPROVE (incoming) */
HRI ST2IPDU; /* HID REJECT (incoming) */
HRH ST2IPDU; /* HID REJECT (outgoing) */
HCI ST2IPDU; /* HID CHANGE (incoming) */
HCH ST2IPDU; /* HID CHANGE (outgoing) */

TIMER
STATE

idle, wfst, hidc, hida, hidn, dtph, wfac, wfcl;

INITIALIZATION

TO idle

BEGIN

END;

TRANS

from idle

to hida

when CNI

OUTPUT CNH, HAH;

trans

to hidc

when CNI

OUTPUT HRH;

trans

to idle

when CNI

output RFH;

TRANS

from hida

to hidn

when HRI

output HCH;

trans

to wfst

when HAI;

trans

to wfcl

when RFI

output RFH, AKH;

TRANS
Appendix E. A Specification of ST-II Protocol in Estelle-Y

from hidn
  to hidn
    when HRI
    output HCH;
  to wfst
    when HAI;
TRANS
from hidc
  to hidc
    when HCI
    OUTPUT HRH;
  to hida
    when HCI
    OUTPUT HAH,CNH;
  to wfcl
    when HCI
    OUTPUT RFH;
  to idle
    when DSI
    OUTPUT AKH;
TRANS
from wfst
  to wfac
    when ACI
    OUTPUT AKH,ACH;
  to wfcl
    when RFI
    OUTPUT RFH, AKH;
trans to wfcl
to wfcl
when DSI
output DSH, AKH;

trans from wfac
to dtph
to dtph
to wfcl
when RFI
output RFH, AKH;
when DSI
output DSH, AKH;
when DTI
output DTH;

trans from idle
to idle
when AKI;

end.
Appendix E. A Specification of ST-II Protocol in Estelle-Y

Target Agent

Specification ST2TARGET;

CONST
VAR
ISP
SAT ST2TSAP; /* STREAM ACCEPT */
SRT ST2TSAP; /* STREAM REFUSE */
OSP
SIT ST2TSAP; /* STREAM OPEN INFORM */
AIT ST2TSAP; /* APPLICATION INFORM MESSAGE */
DIT ST2TSAP; /* DATA INFORM */
PDU
CNT ST2TPDU; /* CONNECT */
DST ST2TPDU; /* DISCONNECT */
ACT ST2TPDU; /* ACCEPT */
RFT ST2TPDU; /* REFUSE */
EQT ST2TPDU; /* ERROR IN REQUEST */
AKT ST2TPDU; /* ACKNOWLEDGE */
HAT ST2TPDU; /* HID APPROVE */
HRT ST2TPDU; /* HID REJECT */
HCT ST2TPDU; /* HID CHANGE */
DTT ST2TPDU; /* DATA */
TIMER
STATE
idle, wfst, hidc, hida, cona, dtph, wfcl;
INITIALIZATION
   TO idle
       BEGIN
       END;
TRANS
   from idle
       to hida
           when CNT
           OUTPUT HAT,SIT;
       to hidc
           when CNT
           OUTPUT HRT;
       to idle
           when CNT
           OUTPUT RFT;
TRANS
   from hidc
       to hidc
           when HCT
           OUTPUT HRT;
       to hida
           when HCT
           OUTPUT HAT,SIT;
       to idle
when DST;

TRANS
from hida
to wfst
when SAT
OUTPUT ACT;
to wfst
when SRT
OUTPUT RFT;
to idle
when DST
OUTPUT AKT, AIT;

TRANS
from wfst
to idle
when DST
OUTPUT AKT, AIT;
to idle
when EQT
OUTPUT AIT;
to wfst
when SAT
OUTPUT ACT;
to wfst
when SRT
OUTPUT RFT;
to wfst
when AKT;
to dtph
  when AKT;

TRANS
from dtph
to dtph
  when DTT
  OUTPUT DIT;
  to idle
  when DST
  OUTPUT AIT, AKT;
  to wfcl
  when SRT
  OUTPUT RFT;

TRANS
from wfcl
to idle
  when AKT;

end.
Appendix F

Test Files and Sample Results

F-1 Seed Files

Origin
originseeds

S00 RFO
S00 EQO
S00 SCO
S00 HRO HAO ACO SDO SDO SCO AKO
S00 HRO HAO ACO SDO SDO RFO
S00 HRO HAO ACO SDO SCO AKO
S00 HRO HAO ACO SDO RFO
S00 HRO HAO ACO SCO AKO
S00 HRO HAO ACO RFO
S00 HRO HAO SCO AKO
S00 HRO HAO SCO
S00 HRO SCO
F-1. SEED FILES

SOO HAO ACO SDO SDO RFO
SOO HAO ACO SDO SCO AKO
SOO HAO ACO SDO RFO
SOO HAO ACO SCO AKO
SOO HAO ACO RFO
SOO HAO SCO AKO

originseeds.atomic

SOO CNO RFO AKO AIO
SOO CNO EQO AKO AIO
SOO CNO SCO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO SDO DTO SCO DSO AKO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO SDO DTO RFO AKO AIO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO SCO DSO AKO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO RFO AKO AIO
SOO CNO HRO HCO HAO SCO AKO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO DSO AKO
SOO CNO HRO HCO HAO ACO AKO AIO SDO DTO RFO AKO AIO
SOO CNO HRO HCO HAO SCO AKO
SOO CNO HRO HCO ACO AIO DSO AKO
Intermediate
intermseeds

CNI HRI HAI ACI AKI RFI AKI
CNI HRI HAI ACI AKI DTI RFI AKI
CNI HRI HAI ACI AKI DTI DSI AKI
CNI HRI HAI RFI AKI
CNI HAI ACI AKI DTI RFI AKI
CNI HAI ACI AKI DTI DSI AKI
CNI HAI RFI AKI
CNI RFI AKI
CNI HCI HRI HAI ACI AKI DTI RFI AKI
CNI HCI HRI HAI ACI AKI DTI DSI AKI
CNI HCI HRI HAI RFI AKI
CNI HCI HAI ACI AKI DTI RFI AKI
CNI HCI HAI ACI AKI DTI DSI AKI
CNI HCI HAI RFI AKI
CNI HCI RFI AKI
CNI HCI AKI
CNI DSI
CNI
CNI HRI HAI DSI AKI
CNI HAI DSI AKI
CNI HCI HRI HAI DSI AKI
CNI HCI HAI DSI AKI
F-1. SEED FILES

intermseeds.atomic

CNI CNH HAH HRI HCH HAI ACI ACH AKH AKI RFI RFH AKH AKI
CNI CNH HAH HRI HCH HAI ACI ACH AKH AKI DTI DTH RFI RFH AKH AKI
CNI CNH HAH HRI HCH HAI ACI ACH AKH AKI DTI DTH DSI DSH AKH AKI
CNI CNH HAH HRI HCH HAI RFI RFH AKH AKI
CNI CNH HAH HAI ACI ACH AKH AKI DTI DTH RFI RFH AKH AKI
CNI CNH HAH HAI ACI ACH AKH AKI DTI DTH DSI DSH AKH AKI
CNI CNH HAH HAI RFI RFH AKH AKI
CNI CNH HAH RFI RFH AKH AKI
CNI HRH HCI CNH HAH HRI HCH HAI ACI ACH AKH AKI DTI DTH RFI RFH AKH AKI
CNI HRH HCI CNH HAH HRI HCH HAI ACI ACH AKH AKI DTI DTH DSI DSH AKH AKI
CNI HRH HCI CNH HAH HRI HCH HAI RFI RFH AKH AKI
CNI HRH HCI CNH HAH HAI ACI ACH AKH AKI DTI DTH RFI RFH AKH AKI
CNI HRH HCI CNH HAH HAI ACI ACH AKH AKI DTI DTH DSI DSH AKH AKI
CNI HRH HCI CNH HAH HAI RFI RFH AKH AKI
CNI HRH HCI CNH HAH RFI RFH AKH AKI
CNI HRH HCI RFH AKI
CNI HRH DSI AKH
CNI RFH
CNI CNH HAH HRI HCH HAI DSI DSH AKH AKI
CNI CNH HAH HAI DSI DSH AKH AKI
CNI HRH HCI CNH HAH HRI HCH HAI DSI DSH AKH AKI
CNI HRH HCI CNH HAH HAI DSI DSH AKH AKI
CNI HRH HCI CNH HAH HAI DSI DSH AKH AKI
CNI HRH HCI CNH HAH HAI DSI DSH AKH AKI
Target

<table>
<thead>
<tr>
<th>Target seeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNT DST</td>
</tr>
<tr>
<td>CNT SAT AKT DST</td>
</tr>
<tr>
<td>CNT SAT AKT DTT DST</td>
</tr>
<tr>
<td>CNT SAT AKT DTT SRT AKT</td>
</tr>
<tr>
<td>CNT SAT AKT EQT</td>
</tr>
<tr>
<td>CNT SAT AKT SRT AKT</td>
</tr>
<tr>
<td>CNT SAT DST</td>
</tr>
<tr>
<td>CNT SAT EQT</td>
</tr>
<tr>
<td>CNT SAT SRT AKT DST</td>
</tr>
<tr>
<td>CNT SAT SRT AKT DTT DST</td>
</tr>
<tr>
<td>CNT SAT SRT DST</td>
</tr>
<tr>
<td>CNT SAT SRT EQT</td>
</tr>
<tr>
<td>CNT DST</td>
</tr>
<tr>
<td>CNT HCT DST</td>
</tr>
<tr>
<td>CNT HCT SAT AKT DST</td>
</tr>
<tr>
<td>CNT HCT SAT AKT DTT DST</td>
</tr>
<tr>
<td>CNT HCT SAT AKT DTT SRT AKT</td>
</tr>
<tr>
<td>CNT HCT SAT AKT EQT</td>
</tr>
<tr>
<td>CNT HCT SAT AKT SRT AKT</td>
</tr>
<tr>
<td>CNT HCT SAT DST</td>
</tr>
<tr>
<td>CNT HCT SAT EQT</td>
</tr>
<tr>
<td>CNT HCT SAT SRT AKT DST</td>
</tr>
<tr>
<td>CNT HCT SAT SRT DST</td>
</tr>
<tr>
<td>CNT HCT SAT SRT EQT</td>
</tr>
<tr>
<td>CNT HCT DST</td>
</tr>
</tbody>
</table>
\documentclass{article}
\usepackage{listings}
\begin{document}
\section*{F-1. SEED FILES}
\begin{verbatim}
CNT HCT HCT DST
CNT HCT HCT SAT AKT DST
CNT HCT HCT SAT AKT DTT DST
CNT HCT HCT SAT AKT DTT SRT
CNT HCT HCT SAT AKT EQT
CNT HCT HCT SAT ACT AKT SRT RFT AKT
CNT HCT HCT SAT DST
CNT HCT HCT SAT EQT
CNT HCT HCT SAT SRT AKT DST
CNT HCT HCT SAT SRT AKT DTT DST
CNT HCT HCT SAT SRT DST
CNT HCT HCT SAT SRT EQT
CNT RFT

targetseeds.atomic

CNT HAT SIT DST AKT AIT
CNT HAT SIT SAT ACT AKT DST AKT AIT
CNT HAT SIT SAT ACT AKT DTT DIT DST AKT AIT
CNT HAT SIT SAT ACT AKT DTT DIT SRT RFT AKT
CNT HAT SIT SAT ACT AKT EQT AIT
CNT HAT SIT SAT ACT AKT SRT RFT AKT
CNT HAT SIT SAT ACT DST AKT AIT
CNT HAT SIT SAT ACT EQT AIT
CNT HAT SIT SAT ACT SRT RFT AKT DST AKT DIT
CNT HAT SIT SAT ACT SRT RFT AKT DTT DIT DST AKT AIT
CNT HAT SIT SAT ACT SRT RFT DST AKT AIT
CNT HAT SIT SAT ACT SRT RFT EQT AIT
CNT HRT DST
\end{verbatim}
\end{document}
Appendix F. Test Files and Sample Results

CNT HRT HCT HAT SIT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT AKT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT AKT DTT DIT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT AKT DTT DIT SRT RFT AKT
CNT HRT HCT HAT SIT SAT ACT AKT EQt AIT
CNT HRT HCT HAT SIT SAT ACT AKT SRT RFT AKT
CNT HRT HCT HAT SIT SAT ACT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT EQt AIT
CNT HRT HCT HAT SIT SAT ACT SRT RFT AKT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT SRT RFT DST AKT AIT
CNT HRT HCT HAT SIT SAT ACT SRT RFT EQt AIT
CNT HRT HCT HRT DST
CNT HRT HCT HRT HCT HAT SIT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT AKT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT AKT DTT DIT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT AKT DTT DIT SRT RFT AKT
CNT HRT HCT HRT HCT HAT SIT SAT ACT AKT EQt AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT AKT SRT RFT AKT
CNT HRT HCT HRT HCT HAT SIT SAT ACT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT EQt AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT SRT RFT AKT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT SRT RFT DST AKT AIT
CNT HRT HCT HRT HCT HAT SIT SAT ACT SRT RFT EQt AIT
CNT RFT
F-2  Experiment 1.1

:................:
seloriginseeds
:................:
:................:
seloriginseeds 6.56
:................:
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
:................:
seloriginseeds 5.90
:................:
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(S00, 1) (RFO, 1)
:................:
seloriginseeds 4.78
:................:
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(S00, 1) (RFO, 1)
(S00, 1) (HAO, 1) (ACO, 1) (SDO, 1) (SCO, 1) (AKO, 1)
:................:
seloriginseeds 3.87
:................:
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(S00, 1) (RFO, 1)
(S00, 1) (HAO, 1) (ACO, 1) (SDO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (HRO, 1) (HAO, 1) (SCO, 1) (AKO, 1)
:................:
Appendix F. Test Files and Sample Results

seloriginseeds 2.82

........................................
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(SOO, 1) (RFO, 1)
(SOO, 1) (HAO, 1) (ACO, 1) (SDO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HAO, 1) (ACO, 1) (RFO, 1)

........................................

seloriginseeds 1.85

........................................
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(SOO, 1) (RFO, 1)
(SOO, 1) (HAO, 1) (ACO, 1) (SDO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HAO, 1) (ACO, 1) (RFO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (RFO, 1)
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (RFO, 1)
(SOO, 1) (HRO, 1) (SCO, 1)
(SOO, 1) (HRO, 1) (ACO, 1) (SDO, 2) (RFO, 1)
(SOO, 1) (HAO, 1) (ACO, 1) (SCO, 1) (AKO, 1)
(SOO, 1) (HAO, 1) (SCO, 1) (AKO, 1)

........................................

seloriginseeds 0.98

........................................
(SOO, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (SCO, 1) (AKO, 1)
(SOO, 1) (RFO, 1)
F-3. EXPERIMENT 4

An excerpt from a randomly generated test suite

(S00, 1) (HAO, 1) (ACO, 1) (SDO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (HRO, 1) (HAO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (HAO, 1) (ACO, 1) (RFO, 1)
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (SDO, 2) (RFO, 1)
(S00, 1) (HRO, 1) (HAO, 1) (ACO, 1) (RFO, 1)
(S00, 1) (HRO, 1) (SCO, 1)
(S00, 1) (HAO, 1) (ACO, 1) (SDO, 2) (RFO, 1)
(S00, 1) (HAO, 1) (ACO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (HAO, 1) (SCO, 1) (AKO, 1)
(S00, 1) (EQO, 1)
(S00, 1) (SCO, 1)
An excerpt from a test suite generated by systematic backtracking
F-3. EXPERIMENT 4

SOO CNO HRO HCO HRO HCO HAO ACO AKO AIO RFO AKO AIO SOO CNO SCO
SOO CNO HRO HCO HRO HCO HAO RFO AKO SCO DSO AKO
SOO CNO HRO HCO HRO HCO HAO RFO AKO SCO DSO AKO SOO CNO SCO
SOO CNO HRO HCO HRO HCO HAO RFO AKO AIO SDO DTO SCO DSO AKO