BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
DEPT. OF TELECOMMUNICATIONS AND MEDIA INFORMATICS

DETECTION, DIAGNOSIS AND CORRECTION OF FAULTS IN TELECOMMUNICATIONS SOFTWARE DEVELOPMENT BASED ON FORMAL METHODS

Zoltán Pap

Summary of the Ph.D. Dissertation

Supervised by

Dr. Gyula Csopaki, Dr. Sarolta Dibuz and Dr. Katalin Tarnay
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Budapest, Hungary
2006

1 Introduction

Telecommunications software plays an increasingly important role for both business and wider society, since it is an indispensable brick in the foundation of today's communication infrastructure. As it becomes more and more significant, there are contradictory requirements these systems have to meet. On the one hand they have to provide more complex services with tightening time limits, on the other hand they have to be reliable, since thousands of applications are built on them.

The increase in complexity of the software increases the probability that faults will be introduced into the system somewhere along the development lifecycle. Faults are permanent defects in the system causing errors, i.e., deviations from intended behavior. Such erroneous behavior may lead to the failure of the system, which is an observable effect outside the system boundary arising from an internal error. Thus, the risk of failures rises along with increasing complexity of the system.

Still, there is a strong need for reliable telecommunications software functioning as required, without failures. Developers have to devote growing effort to ensure that telecommunications systems are of the highest possible integrity, containing as few faults as possible. Furthermore, as product lifecycles shorten speed becomes an ever important factor; therefore the development process should be as short as possible to reduce time-to-market.

The most promising way to cope with these requirements is the use of formal methods for software development. Formal methods are languages and methodologies with well established mathematical background aiding the precise specification and implementation of software and other types of systems. They are often backed by tool support, and can be used not just to describe a system but to analyze its behavior as well. The mathematical foundations for two of the most widely used formal description techniques (FDTs) – Estelle [TC988] SDL (Specification and Description Language) [IT00] – are provided by the (extended) finite state machines-based modeling technique.

As in the case of other systems, a telecommunication software development should follow a lifecycle including different phases such as requirements analysis, architecture design, specification, implementation, testing and maintenance illustrated in Figure 1. The fundamental idea behind formal methods-based software development is that a precise specification of the system can support this lifecycle in a number of ways, and thus help create a higher quality product.

- Precise and unambiguous formal specifications can be applied for the standardization of systems. They can help attain interoperable systems even if components are coming from many different vendors.

- Formal specifications can be used to properly understand the system at an early

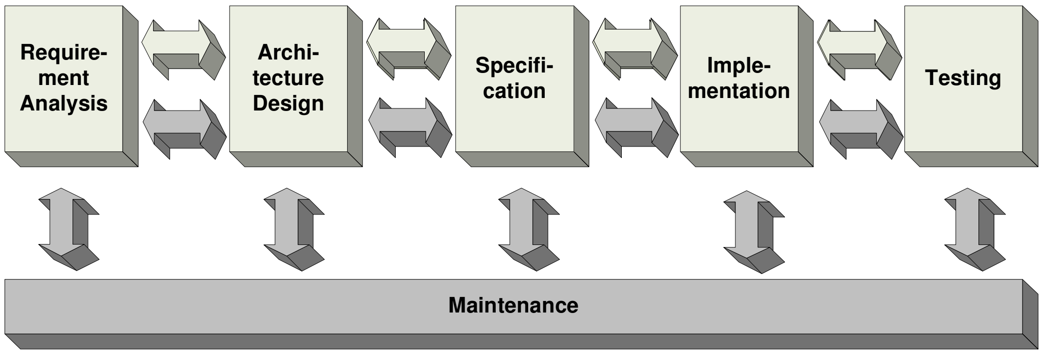

Figure 1: Software development lifecycle

phase, possibly revealing any design problems or incomplete features.

- Formal specifications may serve as a driving force behind the development. The implementation process may be aided either by the refinement of the specification or by direct code generation from the specification.

- A formal specification can be used to support the testing process. It may help, for example, the tester to distinguish right and wrong behavior of an implementation, i.e., to derive correct oracles to check the results of tests. A formal specification could also be used to generate tests that are likely to be effective in detecting faults.

This thesis concentrates on the testing and maintenance phases of the telecommunication software development lifecycle, and considers systems specified using finite state machine (FSM) and extended finite state machine (EFSM) modeling techniques.

Software testing is an important and extremely expensive part of the software development process. Although advances in computer science help improve the quality of software, studies argue that testing is still time-consuming and accounts for around fifty percent of the total development cost. Thus, for economical reasons, it is crucial to make the testing process more efficient in time consumption, cost and quality. The most promising approach to reach this objective is the automation of testing activities. In particular, automation is an especially important challenge for the initial phase of testing, the test set development.

Maintenance is a phase of the software development lifecycle after deployment of the software into the field. It is concerned with changes to the software to correct faults and deficiencies found after deployment, either through additional testing or field usage. It is also used to modify or add functionality to comply with changes of the requirements. While for some systems the only solution for maintenance is to recall distributed implementations. In the case of software systems patches – or updates –

are the most widely used means to change the behavior of a system. There are many benefits of employing patches. It is a faster and less expensive approach than recalls, enabling developers to constantly follow user demands and quickly respond to them.

2 Research Objectives

The purpose of this dissertation is to investigate problems concerning the detection, diagnosis and correction of faults during the testing and maintenance phases of the telecommunication software development lifecycle. The focus is on a development process supported by formal methods involving FSM and EFSM modeling techniques. The objectives cover tree main aspects that are organized around the notion of faults. The aspects include fault detection, fault diagnosis and the correction of faults.

The first objective of my research was to clarify the feasibility of the fault diagnosis problem. I showed that it is not always possible to precisely diagnose (locate) a single fault in an FSM. My target was to work out an algorithm that – in contrast to existing methods – newer produces false results.

My second goal was to improve the efficiency of conformance testing by optimizing test sets. The main purpose of my work was to give an algorithm to reduce test sets for systems defined in SDL without sacrificing coverage. My approach utilized ideas of mutation analysis to automate all steps of the test selection process.

My third objective was to study the problem of patching, i.e., modifying an existing system, which – among other things – may be used to correct faults of a system after its deployment. My goal was to elaborate the means of modeling patches in FSM systems, and to create a framework for optimizing patches.

3 Research Methodology

I have applied a variety of techniques in my thesis to achieve the research objectives discussed above. These include mathematical tools from graph theory, complexity theory and computer science as well as empirical studies. If possible, I have always provided exact complexity figures for the presented algorithms, and demonstrated them through examples. In case of heuristic methods I have used both analytical and empirical investigations to evaluate and validate the approach.

Given that most formal methods model the behavior of telecommunications software as (extended) finite state machines, I have often built on existing methods of the FSM theory in my research. In Thesis 1 I have investigated and improved existing fault diagnosis algorithms [LS93] [GvB92]. The main results of the thesis have been elaborated by analytical investigation. I have given an exact complexity measure for the proposed algorithm and demonstrated it through an example.

Thesis 2 is concerned with test selection for EFSM systems, a hard problem that can only be solved in general using heuristic methods. In my approach I have employed the fundamental idea of mutation analysis [MLS78] to create an algorithm for the problem. I have evaluated the algorithm both by analytical studies and by performing a number of experiments.

In Thesis 3 I have built on existing results of automata theory and elaborated the means of modeling patches based on Chow's work on fault models [Cho78]. I have proposed a new problem of optimizing patches and approached the problem using the tools of graph matching theory. Complexity theory has been applied to show the hardness of the problem. As it turned out to be NP-complete it has been transformed to a state-space search problem that can be approximated with existing heuristic algorithms.

4 New Results

I have ordered my new results – according to my Ph.D. dissertation – into three groups. A more detailed explanation of each statement can be found in the corresponding chapter of my dissertation.

4.1 Feasibility of the FSM-based Fault Diagnosis Problem

Fault diagnosis addresses the problem of locating difference(s) between a system specification and a black box implementation if they are found to be different. A solution to this problem has various applications [LY96]. One of the most important is the correction of an implementation so that it conforms to its specification.

There has been much research concerning fault diagnosis using different assumptions on the specification and on the implementation. There are two papers on fault diagnosis for deterministic finite state machines, using the assumption that the implementation contains only one fault of the following types [GvB92] [LS93]:

- Operation (Output) fault – for a given state and input, the implementation provides an output different from the one specified by the output function.
- Transfer fault – for a given state and input, the implementation enters a different state than specified by the transition function.

Limiting the number of differences between the specification and the implementation to a single fault, both of these papers claim to guarantee the precise localization of the difference. In this Thesis I have investigated the feasibility of the problem. I have used the same (common) assumptions as in [GvB92] [LS93]:

- The specification machine is reduced, completely specified, strongly connected and deterministic with reliable reset capability.
- The implementation machine contains at most one difference – an output or a transfer fault.

Thesis 1. *I have shown that it is not always possible to precisely diagnose a single output or transfer fault in a finite state machine. I have determined a set of sufficient conditions for the guaranteed exact localization of a single fault. Based on the analytical results I have created an algorithm for the fault diagnosis problem. If it is possible, the method exactly locates the difference between the implementation and the specification; in case exact localization is unfeasible it provides the minimal set of all potential single faults.*

4.1.1 Failure of Exact Fault Diagnosis in FSMs

Thesis 1.1. *[C1] I have shown that the exact localization of a single – transition or output – fault in a finite state machine cannot always be guaranteed, not even considering a deterministic, completely specified, strongly connected and reduced specification machine with reliable reset capability.*

I have identified and demonstrated the following problem:

Let us assume specification machine *Spec* and two implementation machines: $Impl_a$ differing from *Spec* by a single fault $Fault_a$, and $Impl_b$ differing from the specification by a single fault $Fault_b$. $Fault_a$ and $Fault_b$ are different faults. Evidently, neither $Impl_a$ nor $Impl_b$ can be equivalent to *Spec*, since *Spec* is deterministic, completely specified, strongly connected and reduced. $Impl_a$ and $Impl_b$, however, may be equivalent to each other despite the fact that the faults they contain differ. In this case it is impossible to decide between the faults, i.e., it is impossible to exactly locate the fault.

4.1.2 Conditions for Guaranteed Fault Diagnosis

I have determined a set of sufficient conditions for the guaranteed exact localization of a single output or transfer fault. That is, I have analyzed when two (or more) implementation machines, each differing from the specification by a single dissimilar fault, cannot be equivalent.

Thesis 1.2. *[C1] I have proven that – given a deterministic, completely specified, strongly connected and reduced specification machine with reliable reset capability – two implementation machines, each differing from the specification by a single and dissimilar fault, cannot be equivalent if the implementation machines are reduced.*

Thesis 1.2 indicates that if there is only one difference between an implementation and a specification and the implementation is minimal then it is unique, no other fault can induce the same change in behavior. Thus, it is possible to exactly identify the given fault.

I have proven Thesis 1.2 by confirming three lemmas with the following statements:

- It is always possible to distinguish two machines with different output faults.
- It is always possible to distinguish a single output and a single transfer fault if the faulty machines are reduced.
- It is always possible to distinguish two different single transfer faults if the faulty machines are reduced.

Note that Thesis 1.2 studies the properties of the implementation FSM. As the implementation machine is black box, these results can not be directly utilized, but they can be used to improve the efficiency of the modified fault diagnosis algorithm.

4.1.3 Exact Algorithm for Fault Diagnosis

Previous efforts on fault diagnosis did not consider the problem discussed in Thesis 1.1, i.e., the exact localization of a single fault in an FSM was regarded as always possible. Existing algorithms, therefore, may provide wrong results: In case the exact localization is not possible they still provide only one result, which may be incorrect (it may be a different fault creating an equivalent machine).

Thesis 1.3. *[C1] I have proposed an algorithm for the fault diagnosis problem. Based on the analytical results the algorithm exactly locates the difference between the implementation and the specification only if it is possible; when the exact localization is not possible, it provides the minimal set of all potential single faults. I have provided two slightly different versions of the algorithm, complete with complexity figures, and demonstrated them through an example.*

I have incorporated the analytical results of Thesis 1.2 to quickly verify if the first fault candidate the algorithm identifies is certainly the only possible one. If it is, then the fault is diagnosed and the algorithm terminates; otherwise the algorithm moves on and provides the minimal set of all potential single faults.

My algorithm is based on Lee's method [LS93], and is made up of the following fundamental steps:

Step 1 – Detection of the fault Conformance testing is conducted on the implementation machine with respect to the specification.

- If no inconsistency is found then the implementation machine is equivalent to the specification. Thus, there is either no fault in the implementation or there are more than one; end of the algorithm.
- If a difference is found then goto Step 2.

Step 2 – Localization of the fault Based on the result of Step 1 all fault candidates are identified. For each fault:

1. A candidate machine is built according to the given fault.
2. Conformance testing is conducted on the implementation machine with respect to the given candidate machine.
    - If they do not conform, it is not a possible fault; goto Step 2.1.

- If they conform then a possible fault is identified. It has to be confirmed if it is the only possible fault: The candidate machine is reduced and compared to the specification.
  - If they have equivalent number of states, then the given fault is certainly the only possible – the fault is exactly located; end of algorithm.
  - Otherwise all remaining fault candidates have to be checked; goto Step 2.1.

The complexity of the algorithm – in the worst case – is $O(pn^5)$ where p is the number inputs $p = |I|$ and n is the number of states $n = |S|$ of the specification. A fast version of the algorithm is presented as well, with complexity of $O(pn^3 \log n)$. I have demonstrated the algorithm through a detailed example in my dissertation.

4.2 Automatic Fault-based Test Selection for SDL Systems

For most telecommunication systems, automatic test generation algorithms – or the manual effort of multiple human experts – result in huge number of test sequences. There might be thousands of test sequences in the given test set. In such cases it is impractical to execute all of the tests, as it takes a significant amount of time to run each test. A possible solution is to reduce the test set by test selection.

The test selection problem can be defined as follows [LH01]: given a large set of test sequences, a minimal subset of tests is to be selected without sacrificing fault coverage. My goal was to automate all steps of the test selection process, and propose a practical approach to optimize conformance test sets for SDL systems.

Thesis 2. *I have proposed an algorithm for automatic conformance test selection building on the ideas of mutation analysis. I have presented a set of mutation operators based on Chow's widely accepted FSM fault model and used them to provide selection criteria. I have conducted a thorough study of my approach both analytically and empirically.*

4.2.1 Mutation Operators Proposed for SDL Systems

Much research has been done concerning mutation operators for different formalisms, but no operator set has been defined for SDL systems. The closest study was on operators for specifications written in Estelle [SFSM00]. Although both Estelle and SDL are based on the EFSM model, I have decided to build on the traditional FSM fault model defined by Chow [Cho78], and create a mutation operator set by extending his model. The rationale of the decision was that existing operators – including the ones defined in [SFSM00] – did not meet three very important requirements:

Atomic operators As previous studies on fault models suggested the use of atomic operators, I have decided to apply as small changes to a system as possible. It is also in accordance with the principles of the mutation analysis technique [MLS78].

Automatically applicable operators While previous studies proposed operators that may be applied manually, my fundamental motivation was to create a fully automatic tool for test selection. Therefore, I had to define operators that can be applied automatically to any specification.

Operators for SDL-specific mechanisms SDL-specific features – like timers, parameterized signals, the "save" mechanism and the implicit consumption of inopportune signals (essentially SDL's completeness assumption[1]) – had to be dealt with.

For the detailed definition of the operators see Section 5.1.1 of my dissertation.

According to the results of Offut et al. [OLR$^+$96], I have proposed a mutant creation strategy that intends to build a representative mutant set, not an exhaustive one. That is, operators are applied systematically to all parts an SDL system, but only some mutants are created for each element of the specification, instead of all possible mutants for the given symbol. Moreover, the operators themselves were elaborated with this strategy in mind, as their definitions already restrict the number of mutants they may create.

Thesis 2.1. *[J0, J4, C4, C6] I have introduced a mutation operator set for SDL specifications created based on a traditional FSM fault model, along with an application strategy. The operators can be applied automatically to any SDL specification, and induce atomic changes to a system. The set includes operators for SDL-specific features as well. I have designed both the operators and the mutant creation strategy to build a representative mutant set.*

4.2.2 Algorithm for Test Selection

Thesis 2.2. *[J0, J4, C4, C6] I have proposed a novel algorithm for test selection. It takes the SDL specification of the system to be tested along with a large set of initial test sequences as inputs, and provides a subset of the original test set with equivalent mutant coverage. The algorithm applies the mutation operators discussed in Thesis 2.1 to provide selection criteria automatically. I have conducted a thorough study of my approach both analytically and empirically.*

Figure 2 presents my algorithm.

[1]If an input arrives for which no transition is defined then it is implicitly consumed meaning that the input is taken from the queue and the system remains at the given state.

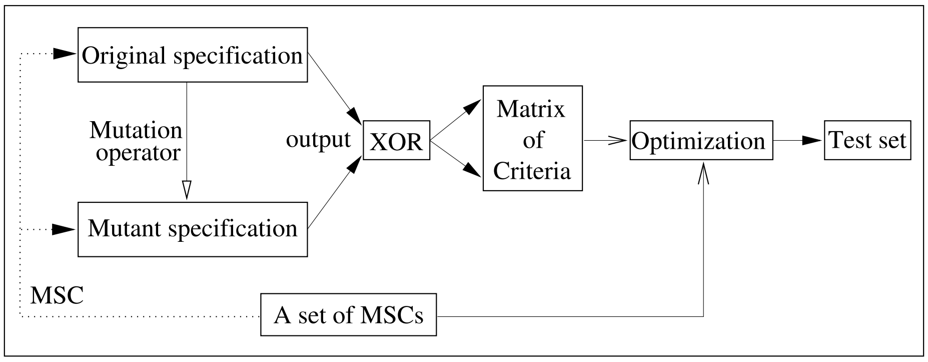

Figure 2: Test selection algorithm based on mutation analysis

The outline of the algorithm is as follows:

1. Create a mutant specification.
2. Run the test set on the mutant specification and check for inconsistency.
3. Mark the test sequences killing the given mutant.
4. Repeat steps 1-3 for all mutants created for the given system according to the mutant creation strategy.
5. Acquire the matrix of criteria, where rows represent the mutants and columns represent the test sequences in the original set. The matrix describes for each test sequence the mutations detected by the given test sequence.
6. Select an optimal test set from the original set, using an optimization method.

The last step of the algorithm is an optimization problem: One has to find the minimal number of tests with the highest possible mutant coverage. Note that a mutant is considered covered if at least one of the test sequences killing him is in the final test set. For an overview of exact and heuristic algorithms for the optimization problem see [Csö01].

My algorithm has the following advantages and improvements over existing methods [2]:

[2]Obviously, the algorithm augmented with any (for example random) state-space exploration approach may be viewed as a test generation method composed of two steps, test derivation and selection. Therefore, – although this research focuses on the test selection problem – it should also be analyzed how my algorithm compares to existing EFSM test generation methods.

Test selection methods There are a number of papers on test selection but none of them is based on mutation analysis ideas. The most relevant research has been done by Lee et al. in [LH01]; they optimize test sets based on the coverage of edges in the reachability graph. Although their results have a very clear theoretical background, the approach is for most systems impractical, as even for small EFSMs the size of the reachability graph is too large. Conversely, my algorithm may be applied to systems with realistic size.

Fault coverage-based test generation methods My algorithm can be compared directly to other mutation testing approaches. There are a number of differences and improvements in my approach compared to previous research (including [SFSM00]). The most important are:

- Mutation operators – The algorithm is using a special set of mutation operators developed based on Chow's FSM fault model [Cho78] for SDL. For detailed discussion on the operators and their background see Thesis 2.1.
- Test selection algorithm – I have proposed an algorithm to optimize existing test sets by selecting a subset of the initial tests with equivalent (mutant) coverage. Previous studies, in turn, used mutation operators only as an aid for manual test creation (generation) process.
- Tool and experiments – The testing strategy described in previous studies was applied manually, no tool has been developed to aid the process. My algorithm, on the other hand, has been designed to work automatically, and a tool has been developed based on it.

Specification coverage-based test generation methods Most EFSM test generation methods use the specification itself as a coverage criterion and require the test to execute each transition in the specification EFSM at least once. Since in my algorithm mutation operators are applied systematically to all parts of the specification, the specification coverage of the optimized test set will be equivalent to the coverage of the initial set for virtually all practical protocols. That is, my algorithm does not reduce the coverage of a test set assuming a specification coverage metric, but reduces it's size (if possible).

Validation I have carried out several experiments on well-known protocols like INRES [Hog91], the conference protocol [BFV$^+$99], or the WAP-WTP (Wireless Application Protocol – Wireless Transaction Protocol) to investigate the presented algorithm. The experiments have followed two different schemes: The initial test sets have either been created automatically with a random state-space exploration algorithm or manually by multiple experts. All experiments have shown significant decrease in the size of the test set (between 74% and 93% reduction), and have shown that the initial and final test sets have equivalent coverage according to the most prevalent metric – the structural coverage. Human experts have evaluated both the initial and final test sets, and have found the approach adequate. Furthermore, the performance results have been acceptable considering the speed problems of the Java language, and the testing hardware used.

4.3 On the Correction of Faults – The Theory of Patching

Patches – or updates – are the most widely used means to change the behavior of an existing system. They – among other things – may be used to modify functionality or to correct faults of a system after its deployment. The key benefit of employing patches is that one has to only distribute the difference – the modifications in the system – instead of replacing the whole.

Despite the practical importance and extensive use of patches there have been no studies on modeling and analyzing patches. The main purpose of my research was to elaborate the means of modeling patches in FSM systems, and to create a framework for optimizing patches. I have focused on deterministic machines, and considered two FSMs identical if they are isomorphic – a natural consideration since Mealy machines are in Principe edge-labeled directed graphs for which the actual labeling of the states is irrelevant. For my discussions I have used finite state machines $M = (I, O, S, \delta, \lambda)$, $M' = (I, O, S', \delta', \lambda')$, etc., where I, O and S are the finite set of input symbols, output symbols and states, respectively; δ: $S \times I \rightarrow S$ is the transition function and λ: $S \times I \rightarrow O$ is the output function.

I have considered a common completeness assumption to handle incompletely specified machines.[3]

Thesis 3. *I have presented a novel approach to model patches. I have introduced the new problem of optimizing patches and conducted a complexity analysis of the problem. As it is unlikely to have a polynomial time solution, I have discussed how to turn the optimal patch problem to a state-space search problem that can be approximated with existing heuristic algorithms.*

4.3.1 Modeling Patches

My approach of modeling patches is based on existing fault models. For completely specified and deterministic machines the most widely accepted model was defined by

[3]According to the completeness assumption, a unique output symbol Υ_o is added to the set of output symbols O and a unique state Υ_s to the set of states S denoting the null (error) output symbol and state, respectively.

Chow [Cho78].

Thesis 3.1. *[C0] I have shown that Chow's fault types with some constraints can be considered as edit operators. Moreover, I have pointed out that sequences of edit operations may be used to model patches.*

I have defined the following types of edit operators based on Chow's fault model: A transfer operator is $TRO : \delta(s,i) = s_1 \rightarrow s_2$,[4] where $s_1, s_2 \in S, S' = S$ (S includes Υ_s). An output operator is $OO : \lambda(s,i) = a \rightarrow b$, where $a, b \in O$ (O includes Υ_o). An extra state operator is $ESO : a,\ S' := \{S \cup a\}$; a missing state operator is $MSO : a,\ S' := \{S \setminus a\}$.

I have proposed the following two constraints on the edit operations:

1. An extra state operator implies the addition of state s with null transition $\delta(s,i) = \Upsilon_s$ and null output functions $\lambda(s,i) = \Upsilon_o$ for all $i \in I$, i.e., it is an addition of a "blank state".

2. A missing state operator can only be applied to state s if s has no incoming transitions and all of its transition and output functions are $\delta(s,i) = \Upsilon_s$ and $\lambda(s,i) = \Upsilon_o$ respectively for all $i \in I$, i.e., only blank states with no incoming transitions may be removed.

I have defined sequences of edit operators. Let's consider an edit operation e applied to finite state machine M_1 resulting in FSM M_2; this is written $M_1 \Rightarrow M_2$ via e. Let E be a sequence $e_1, e_2, ..., e_k$ of edit operations. E changes FSM M to FSM M' if there is a sequence of machines $M_0, M_1, ..., M_k$ such that $M = M_0$, $M' = M_k$, and $M_{i-1} \Rightarrow M_i$ via e_i for $1 < i < k$.

Clearly, the set of deterministic finite state machines with a given set of input and output symbols I and O is closed under the edit operations defined above. Moreover, for any two deterministic FSMs M_1 and M_2 there is always a sequence of edit operations E changing M_1 to M_2, i.e., to a machine isomorphic to M_2. I have pointed out that sequences of edit operations, therefore, can be considered as models of patches defining modifications – changes – to an FSM system.

4.3.2 The Optimal Patch Problem

Obviously, there are infinitely many possible sequences of edit operations – patches – between two particular finite state machines changing one to the other. My objective has been to find the "best" among them.

[4]For notational simplicity, $\delta(s,i) = s_1 \rightarrow s_2$ is used to represent a transfer operator instead of $[\delta(s,i) = s_1] \rightarrow [\delta'(s',i) = s_2']$, where M is the original and M' the edited machine. Similar notation is used for output operators as well.

Thesis 3.2. *[C0] I have improved the model of patches by assigning costs to edit operations. I have proposed a new problem I call the optimal patch – or optimal update – problem. Given a requirement for a change, it is concerned with finding the optimal patch – minimal cost edit operations – modifying the system accordingly.*

I have refined the model of patches using a cost function ρ that assigns a nonnegative real number $\rho(e)$ to each type of edit operation. I have constrained ρ to be a distance metric, i.e., it satisfies nonnegative definiteness, symmetry and triangle inequality. I have extend ρ to a sequence of edit operations $E = e_1, e_2, ..., e_k$ by letting $\rho(E) = \sum_{i=1}^{k} \rho(e_i)$.

I have defined the optimal patch problem for finite state machines: Let us consider an FSM M modeling the behavior of an existing system already implemented and distributed among users. For some reason the need arises to modify the behavior and a new design is created, modeled by M'. The problem is to determine the optimal patch updating the system, i.e., the edit operations changing M to M' that are minimal according to a given cost function.

As ρ is constrained to be a distance metric, this problem can be stated as finding the (edit) distance between two machines M and M', where the distance between M and M' is defined to be the minimum cost of all sequences of edit operations that change M to M':

$$dist(M, M') = min\{\rho(E)\ |E\ is\ a\ sequence\ of\ edit\ operations\ changing\ M\ to\ M'\}$$

I have emphasized that the solution to optimal patch problem (finding the edit distance) can be viewed as a measure of similarity for two FSM systems. Thus, it may have different uses such as assessing the effort of a developer or detecting plagiarism.

4.3.3 Transformations and Edit Operations

Unfortunately, it is not convenient to work directly with sequences of edit operations as there are infinitely many ways of changing a given system to the other. Thus, I have proposed an alternative approach to describe changes.

Thesis 3.3. *[C0] I have introduced transformations that are used as an alternative way of describing changes. I have created an algorithm determining the sequence of edit operations corresponding to a transformation. I have proved that the algorithm is not an arbitrary choice; it provides the lowest cost sequence of edit operations for a given transformation. Thus, I have proved that the problem of finding the optimal patch can be solved by finding the minimum cost transformation.*

Given two FSMs M and M' a transformation T is a set of pairs of states (s, s') where $s \in S_{sub} \subseteq S$, $s' \in S'_{sub} \subseteq S'$ and for any pair (s_1, s'_1) and (s_2, s'_2) in T, $s_1 = s_2$ if and only if $s'_1 = s'_2$ (state one-to-one). Informally, transformations describe ways to change one deterministic FSM to an other.

I have given an algorithm determining the sequence of edit operations corresponding to a transformation T from FSM M to M'. The essence of the algorithm is as follows:

- Apply extra state operators to create a new blank state M for each state of M' not in T.[5]
- Now there is a bijection between S' and the states of M in T along with the new blank states of M.
- Apply output and transfer operators to change M according to the bijection.
- Apply missing state operators to remove the – now blank – states of M not in T.

The algorithm is polynomial time with a complexity of $O(pn)$, where $n = |S|$ and $p = |I|$.

I have assigned costs to transformations: $\rho(T) = \sum_{i=1}^{k} \rho(e_i)$ where e_i are the edit operations corresponding to T (according to the algorithm above).

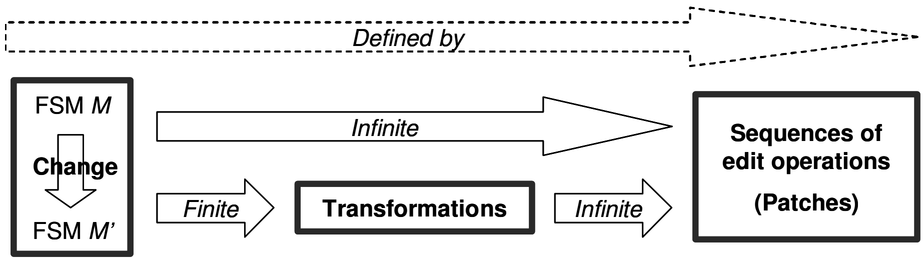

Figure 3: Cardinality of transformations and patches for a given change

Clearly, there are infinitely many sequences edit operations fulfilling a given transformation T – with different costs. Figure 3 shows the cardinality of transformations and patches for a given change.

I have proven that the sequence of edit operations I have assigned to T by the algorithm above is not an arbitrary choice, but the best possible sequence of edit

[5]For simplicity the term a state is (not) in T is used meaning that the given state does (not) occur in any of the state-pairs of T.

operations for T with the lowest cost. I have shown this by proving the following statements:

- $\rho(T_1 \circ T_2) \leq \rho(T_1) + \rho(T_2)$, where $T_1 \circ T_2$ is the composition of transformations: $T_1 \circ T_2 = (s, s'')|\exists s' \in S'$ such that $(s, s') \in T_1 \wedge (s', s'') \in T_2$.
- For any sequence of edit operations $E = e_1, e_2, ..., e_k$ changing FSM M to M'' there is a transformation T from M to M'' such that $\rho(T) \leq \rho(E)$. Conversely, for any transformation T there exist a sequence of edit operations E where $\rho(E) = \rho(T)$.

By proving the previous statements I have clarified the relation of transformations and edit distance. I have shown that the cost of the minimal transformation is equal to *dist*, i.e., a minimal cost transformation provides an optimal patch.

$$dist(M, M') = min\{\rho(T)\ |T\ is\ a\ transformation\ from\ M\ to\ M'\}.$$

Transformations, therefore, provide a more constructive way to work with the optimal patch problem.

4.3.4 Complexity Analysis and Heuristics

Thesis 3.4. *[C0] I have conducted a complexity analysis of the optimal patch problem and proved that it is NP-complete. I have discussed how to turn the optimal patch problem to a state-space search problem that can be approximated with existing heuristic algorithms.*

I have reduced the (Perfect) Tripartite Matching problem to the optimal patch problem, i.e., to the problem of finding $dist(M, M')$). As Tripartite Matching was among the original few problems that were shown to be NP-complete [Kar72], I have proved that the optimal patch problem is NP-complete as well. This result indicates that heuristic approaches are needed to efficiently approximate the result.

I have shown how to turn the optimal patch problem to a state-space search problem, which can be approximated with existing heuristic algorithms. I have focused on the formulation of the problem, not on the heuristics themselves. The discussion of different approximation algorithms, their comparative performance analysis, parameter settings, etc. are beyond the scope of my thesis; it would take a whole dissertation to thoroughly investigate these problems.

The essence of my approach is the following:

Consider two FSMs M and M', and create a state-space where each state corresponds to a transformation T between M and M'. Let σ_T represent the state of the state-space corresponding to transformation T, and associate the cost of T to the given state σ_T: $\rho(\sigma_T) = \rho(T)$.

States σ_{T_1} and σ_{T_2}, corresponding to transformation T_1 and T_2 respectively, are neighbors if T_2 can be obtained from T_1 by adding or removing one pair of states in the mapping defined by T_1. Neighboring states can be reached from one another by one move in the state space. A walk is a sequence of moves between states. Clearly the state space is connected, that is, for each pair of states σ_{T_i} and σ_{T_j} there is a walk from σ_{T_i} to σ_{T_j}.

With that I have turned the problem of finding $dist(M, M')$ to find the the lowest cost among all states of the state space. As an example I have demonstrated how simulated annealing (SA) [KGV83] [Cer85] – a widely applied heuristic – may be used to approximate the result.

5 Application of the Results

The results and algorithms I have presented in my dissertation may be applied primarily in the testing and maintenance phases of the formal methods-based software development lifecycle.

Beside of clarifying the theory of fault diagnosis, Thesis 1 provides the means of locating faults in a black box system, avoiding any false results. The algorithm may be applied directly to systems modeled by FSMs or to control parts of EFSM-based formal descriptions to diagnose faults. For best results the method should be used to monitor an implementation under change frequently enough, before multiple faults occur.

Thesis 2 proposes a practical method to optimize conformance test sets. The presented algorithm may be applied to reduce existing test sets created for any telecommunications software, given an SDL description of the system exists. As the use of formal methods in the software development lifecycle is becoming increasingly prevalent, the need of an SDL description is becoming less restrictive.

Thesis 3 is concerned with patching, an approach extensively used in practice to change the behavior of deployed software. The results of Thesis 3 may be applied either to systems modeled by FSMs or to control parts of EFSM-based formal descriptions for the following purposes:

- To model, describe and understand patches.
- To optimize patches.
- To measure the similarity of systems that may be used to assess the effort of a developer or to detect plagiarism.

Acknowledgements

Many people have contributed to the development of the ideas and methods presented in this dissertation over the last few years. First and foremost I wish to express my deepest gratitude to my advisors Gyula Csopaki, Sarolta Dibuz and Katalin Tarnay. This thesis would not have been possible without their constant support and guidance. I would like to thank the Department of Telecommunications and Media Informatics at Budapest University of Technology and Economics for providing an environment in which it has been possible to freely pursue independent thoughts. Many thanks to all the members of the Telecommunications Software Group for making it a stimulating and enjoyable place to work. I am indebted to my colleagues for their thoughtful comments, criticism, and encouragement provided throughout our collaborate work.

References

[BFV$^+$99] A. Belinfante, J. Feenstra, R.G. de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *Proceedings of the* 12th *International Workshop on Testing of Communicating Systems*, pages 179–196. Kluwer Academic Publishers, 1999.

[Cer85] V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, pages 41–51, 1985.

[Cho78] T. Chow. Testing software design modelled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, May 1978.

[Csö01] T. Csöndes. *Conformance Test Suite Optimization*. PhD thesis, Budapest University of Technology and Economics, 2001.

[GvB92] A. Ghedamsi and G. v. Bochmann. Test result analysis and diagnostics for finite state machines. In *Proceedings of the 12th International Conference on Distributed Systems*, 1992.

[Hog91] D. Hogrefe. OSI formal specification case study: The INRES protocol and service (revised). Technical Report IAM-91-012, Universitat Bern, Institut fur Informatik, 1991.

[IT00] ITU-T. Recommendation Z.100: Specification and description language, 2000.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[LH01] D. Lee and R. Hao. Test sequence selection. In *Formal Techniques for Networked and Distributed Systems, FORTE 2001, IFIP TC6/WG6.1 - 21st International Conference on Formal Techniques for Networked and Distributed Systems, Cheju Island, Korea*, pages 269–284, 2001.

[LS93] D. Lee and K. Sabnani. Reverse-engineering of communication protocols. In *Proceedings of the IEEE International Conference on Network Protocols, California*, pages 208–216, 1993.

[LY96] D. Lee and M. Yiannakakis. Principles and methods of testing finite state machines – a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[MLS78] R. A. De Millo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *IEEE Computer*, 11(4):34–41, April 1978.

[OLR+96] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(2):99–118, 1996.

[SFSM00] S. R. S. Souza, S. C. P. F. Fabbri, W. L. Souza, and J. C. Maldonado. Mutation testing applied to Estelle specifications. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, Washington, DC, USA, 2000. IEEE Computer Society.

[TC988] ISO TC97/SC21. Estelle – a formal description technique based on an extended state transition model. international standard 9074, 1988.

Publications

Journal papers

[J0] Gábor Kovács, **Zoltán Pap**, Gyula Csopaki and Katalin Tarnay. Iterative automatic test generation method for telecommunication protocols. *Computer Standards & Interfaces*, Accepted paper (Corrected proof in Press), 2005.

[J1] Gábor Vincze, **Zoltán Pap** and Róbert Horváth. Peer-to-Peer Based Distributed File Systems, *International Journal of Internet Protocol Technology (IJIPT)*, Accepted paper, 2005.

[J2] Gábor Vincze, **Zoltán Pap**, Róbert Horváth. Peer-to-peer alapú elosztott fájlrendszerek. *Hiradástechnika*, Volume LX, Pages 21-26, 2005. (in Hungarian).

[J3] Gusztáv Adamis, Róbert Horváth, **Zoltán Pap** and Katalin Tarnay. Standardized languages for telecommunication systems. *Computer Standards & Interfaces*, Volume 27, Number 3, Pages 191-205, March 2005.

[J4] Gábor Kovács, **Zoltán Pap**, Gyula Csopaki. Automatic Test Selection Based on CEFSM Specifications. *Acta Cybernetica*, Volume 15, Number 4, pages 583-599, 2002.

[J5] **Zoltán Pap**, Zoltán Rétháti, Róbert Horváth, Gusztáv Adamis. Standardized Event Pair Based Test Generation Method Using TSS&TP. *Acta Cybernetica*, Volume 15, Number 4, Pages 653-667, 2002.

[J6] Róbert Horváth, **Zoltán Pap**, Zoltán Rétháti. Development of Telecommunications Software Using Formal Languages. *Magyar távközlés selected papers 2000*, Pages 48-50, 2000.

[J7] Róbert Horváth, **Zoltán Pap**, Zoltán Rétháti. Távközlési szoftver fejlesztés formális nyelvek felhasználásával. *Magyar távközlés*, Volume X, Number 7, Pages 3-6, July 1999. (in Hungarian).

[J8] **Zoltán Pap**. IP alapú távközlés. *Magyar távközlés*, Volume X, Number 6, Pages 19-22 June 1999. (in Hungarian).

Conference papers

[C0] **Zoltán Pap**, Gyula Csopaki, Sarolta Dibuz. On the Theory of Patching, in *Proceedings of the 3rd IEEE International Conference on Software Engineering and Formal Methods, SEFM 2005,* Pages 263-271, Koblenz, Germany, September 5-9, 2005.

[C1] **Zoltán Pap**, Gyula Csopaki, Sarolta Dibuz. On FSM-based Fault Diagnosis, in *Proceedings of the Testing of Communicating Systems, 17th IFIP TC6/WG 6.1 International Conference, TestCom 2005,* Pages 159-174, Montreal, Canada, May 31 - June 2, 2005.

[C2] Gusztáv Adamis, **Zoltán Pap**, Róbert Horváth. Introduction of Aspect-Oriented Methodology to Formal Description Techniques, in *Proceedings of the IFIP WG6.3 Workshop on Next Generation Networks & EUNICE'2003,* Balatonfüred, Hungary, September 8-10, 2003.

[C3] Gusztáv Adamis, **Zoltán Pap**, Róbert Horváth. Self-adaptive Testing of Communication Protocols, in *Proceedings of IWSAS 2003, International Workshop on Self-Adaptive Software,* Arlington, VA, USA, June 9-11, 2003.

[C4] Gábor Kovács, **Zoltán Pap**, Dung Le Viet, Antal Wu-Hen-Chang, Gyula Csopaki. Applying Mutation Analysis to SDL Specifications, in *Proceedings of the Eleventh SDL Forum "System Design",* Pages 269-284, Stuttgart, Germany, 1-4 July, 2003.

[C5] **Zoltán Pap**, Zoltán Rétháti, Gusztáv Adamis. Standardized Beta Test Subsequence Based Test Generation Method Using Formal Documents, in *Proceedings of the 4th Symposium on Wireless Personal Multimedia Communications - WPMC'01,* Aalborg, Denmark, September 9-12 2001.

[C6] Gábor Kovács, **Zoltán Pap**, Gyula Csopaki. Automatic Test Selection Method Applied to WAP, in *Proceedings of IFIP Workshop on IP and ATM Traffic Management WATM'2001 & EUNICE'2001,* Pages 115-121, Párizs, France, September 3-5, 2001.

[C7] **Zoltán Pap**, Zoltán Rétháti, Gusztáv Adamis. Alternative Test Generation Method Based on Atomic Test Cases, in *Proceedings of IFIP Workshop on IP and ATM Traffic Management WATM'2001 & EUNICE'2001,* Pages 107-114, Párizs, France, September 3-5, 2001.

[C8] Róbert Horváth, **Zoltán Pap**, Zoltán Rétháti. Protokoll fejlesztés és tesztelés formális nyelveken, in *Proceedings of Students Scientific Conference,* Budapest University of Technology and Economics, Budapest, Hungary, November, 1999.