AUTOMATED TEST CASE SELECTION BASED ON SUBPURPOSES

Tibor Csöndes

Conformance Center, Ericsson Ltd.

Laborc u. 1., Budapest, 1037, Hungary

Tibor.Csondes@eth.ericsson.se

Balázs Kotnyek

Eötvös Loránd University

Budapest, Hungary

kotnyekb@cs.elte.hu

Abstract In this paper we present an approach to test selection from ATS. The method can help test laboratories to decide which Test Cases should be executed. The method uses TTCN, it is flexible in terms of the different preferences of the laboratories, it is efficient and mostly automated. We introduce the new concept of *subpurpose*, which is a detectable part of the Test Purpose and convey some parts of the parallelism between the tests. We define three different models and two types of optimization problem, and give their mathematical formulations. In the second half of the paper we present empirical results obtained by applying the method to a quite complex real-life protocol.

Keywords: Protocol testing, test selection, ATS, TTCN

1. INTRODUCTION

Conformance Testing is usually put into practice by test laboratories. Their task is to check whether the Implementation Under Test (IUT) is conform to the standards or not. In order to establish their verdict they have to use a standardized test suite the Abstract Test Suite (ATS). The ATS contains a considerable number of Test Cases (TC), which focus on specific parts of

protocol behaviour, and can be executed independently from each other. The time test laboratories can spend on testing of an IUT is usually strongly limited. That is the reason why they have to make a selection, to decide which TCs will be executed and which ones will be omitted. Of course, they try to select a set of tests that covers as widely the protocol behaviour as possible, in other words, the coverage of which is as big as possible. A good selection has important economic consequences. Test laboratories can complete testing faster or, in other point of view, they can achieve bigger coverage in a given time. Therefore a method that can help them in efficient selection has considerable practical importance. Such a method should have the following properties:

- It is efficient as it makes the test process faster by losing only a relatively small amount of coverage.
- It is worth doing because it needs less time than it can save.
- It is applicable to different types of real-life protocols.
- It is flexible as special preferences of the test laboratories can be incorporated as well as their knowledge of the specific protocol.
- It is reproducible, the selection can be repeated if it is needed.
- It selects TCs from the ATS, so the testers can execute them straightforwardly.

So far, to our knowledge, there has not been reported such a method. The approach that is the closest to these aims is the metric based test selection [1, 2]. This method selects from a pre-generated set of "execution sequences" based on their "distances". Although it is surely possible to convert the ATS into a set of execution sequences, to define an adequate distance function, and then to select a set of sequences to execution with the metric based method, it needs considerable work and time. Furthermore, the execution sequences do not correspond exactly to Test Cases, so the testers cannot execute them straightforwardly.

Our proposed approach is based on the formal specification of the ATS, the TTCN format [3]. In general, this is the only formal specification that is available. Most of the test laboratories know this notation and use it for testing, so they do not need to learn a new notation and create the model of the ATS in that format. As the testers are able to use TTCN it is possible to carry on the whole process of selection and testing automatically and continuously having input the limits and demands at the beginning. Our method decides which TCs are the best to execute within a time limit or which ones should be executed to reach a given lower bound for the coverage in the shortest possible time.

In this paper we present our method and show how it can be applied to a real-life protocol, the ISDN DSS1 Layer 3 Protocol [4]. Although the results may not be extrapolated to all other protocols, we believe it can demonstrate the efficiency of the method.

The rest of the paper is organized as follows. In next section the basic theory of the method is presented as well as the possible models and their mathematical formulations. In Section 3 we present how the theory can be turned into practice, and Section 4 gives a series of experiments. Finally, we conclude by discussing on observations and further research work.

2. THEORETICAL BASIS

The aim of Conformance Testing is to check whether the Implementation Under Test (IUT) fulfils the requirements it should fulfil. The set of requirements a Test Case (TC) can check is described in its Test Purpose (TP). The TPs of different Test Cases usually overlap, there are requirements more tests can check. If we have to select Test Cases to execute, we want to leave out tests the requirements of which will be checked by the selected ones. That is the reason why we would like to detect the connections between the tests on the basis of their purposes. For this we should determine the requirements related to a TC.

There are protocols the conformance requirements of which are given in the ATS standard accompanied by the Test Cases which are able to check them. We have shown an example of such protocols (ISDN DSS1 Layer 2 protocol [5]) in a former paper [6].

Most of the protocols do not have this pleasant property, the only thing given is the Test Purpose of every Test Case. The TP is written in informal language, so it is practically impossible to detect the requirements and the connection between different TCs using only their TPs. We should find the building blocks of the purposes, and we should do it automatically in order to make the selection faster and more reproducible avoiding the different interpretations of the prose TP. These building blocks have to characterize the requirements and convey some part of the parallelism between the tests. We will call such a block **subpurpose**.

Subpurpose can be the requirement itself if we are able to detect them. Otherwise we have to choose them depending on the type of the ATS. For example the Protocol Data Units (PDU) used in the TC or the parameters of the PDUs can play the role of subpurposes. Subpurposes are properly chosen if they help us to make efficient selection. One of the main aims of this paper is to present an example where the PDUs are the subpurposes.

2.1 THE MODEL

Let us define the **Subpurpose - Test Case Incidence Matrix** and let us denote it A . Subpurposes $(S_1, S_2, \dots, S_k)$ are placed in the rows, Test Cases $(T_1, T_2, \dots, T_n)$ are placed in the columns. The entry of the matrix standing in the (i, j) cell is 1 if S_i is a subpurpose of T_j . Otherwise the entry is 0. If we find some 1 entries in a given row (a_i denotes the i^{th} row), then the TCs related to the 1s are connected in terms of their subpurposes. We can assume that executing one of these tests we get some information about the possible result of the execution of the others.

We append a **cost** to every TC. The cost of the j^{th} test, $c(T_j)$ measures the amount of resources (usually the time) needed to execute it. We may be able to determine it exactly or, more frequently, we may use estimation. The cost of a set of TCs can be defined as the sum of the individual TCs in the set.

We define the **coverage** function of the subpurposes. The coverage of a subpurpose ($cov(S_i)$) may be obtained from theoretical consideration or we can simply mark the priority of the subpurpose with it. The test laboratories can express their possible individual preferences or knowledge of the specific protocols and potential faults by choosing an appropriate coverage function.

Let $\mathcal{T}$ be a set of the Test Cases, $\mathcal{T} \subset \{T_1, T_2, \dots, T_n\}$. Let us denote the coverage we get if we execute $\mathcal{T}$ with $cov(\mathcal{T})$. This coverage is composed of the values $cov(S_i, \mathcal{T})$, which equal the partial coverage we achieve of $cov(S_i)$ ($i = 1, 2, \dots, k$) executing only $\mathcal{T}$.

$$cov(\mathcal{T}) = \sum_{i=1}^k cov(S_i, \mathcal{T})$$

If we execute all the tests related to the i^{th} subpurpose $cov(S_i, \mathcal{T})$ equals $cov(S_i)$, but executing only parts of the tests we may achieve smaller coverage. Let b_i be the number of TCs related to S_i (equals the number of the 1 entries in the i^{th} row of the matrix). The increasing function $f_i : \{0, 1, \dots, b_i\} \rightarrow [0, cov(S_i)]$ measures the coverage that can be obtained executing m TCs of ones related to S_i : $f_i(m) = cov(S_i, \mathcal{T})$. Naturally, $f_i(0) = 0$ and $f_i(b_i) = cov(S_i)$. We define three models that differ in choosing functions f_i .

- 1 **"Linear" model:** The coverage is in direct proportion to the number of executed tests i.e.

$$f_i(m) = \frac{m}{b_i} cov(S_i) \quad m = 0, \dots, b_i$$

- 2 **"All or nothing" model:** We consider the subpurpose being checked only when we have executed all the related tests.

$$f_i(0) = f_i(1) = \dots = f_i(b_i - 1) = 0 \quad \text{and} \quad f_i(b_i) = cov(S_i)$$

- 3 **"One is enough" model:** If only one test is executed among the ones that correspond to S_i , we will get the whole coverage.

$$f_i(0) = 0 \quad \text{and} \quad f_i(1) = \dots = f_i(b_i) = \text{cov}(S_i)$$

2.2 OPTIMIZATION

Let x be the characteristic vector of $\mathcal{T}$, so $x \in \{0, 1\}^n$, and $x_j = 1$ if $T_j \in \mathcal{T}$ and $x_i = 0$ if $T_i \notin \mathcal{T}$. The cost of $\mathcal{T}$

$$c(\mathcal{T}) = \sum (c(T_j) \mid T_j \in \mathcal{T}) = \sum_{j=1}^n c(T_j)x_j = cx$$

where $c = (c(T_1), \dots, c(T_n))$ is the cost vector. The number of tests that are related to S_i and are in $\mathcal{T}$ (above we used the notation m for this number) is $a_i x = \sum_{j=1}^n a_{ij}x_j$ where a_i is the row of the matrix corresponding to S_i so

$$\text{cov}(\mathcal{T}) = \sum_{i=1}^k \text{cov}(S_i, \mathcal{T}) = \sum_{i=1}^k f_i(a_i x)$$

We will use the notation $v = (\text{cov}(S_1), \dots, \text{cov}(S_k))$ for the coverage vector.

We introduce two possible optimization problems. In the first one our aim is to select a test set from the test suite with **minimal cost**, supposing a constraint (K) bounding the coverage from below.

$$\begin{aligned} & \min \quad c(\mathcal{T}) \\ & \text{subject to} \quad \text{cov}(\mathcal{T}) \geq K \\ & \quad \mathcal{T} \subset \{T_1, T_2, \dots, T_n\} \end{aligned}$$

Using the characteristic vector and functions f_i this problem can be written as

$$\begin{aligned} & \min \quad cx \\ & \text{subject to} \quad \sum_{i=1}^k f_i(a_i x) \geq K \\ & \quad x \in \{0, 1\}^n \end{aligned} \tag{1}$$

Let us see now how this formula looks like in the case of the three mentioned models.

"Linear" model for the minimal cost problem.

$$\sum_{i=1}^k f_i(a_i x) = \sum_{i=1}^k \frac{a_i x}{b_i} \text{cov}(S_i) = \left(\sum_{i=1}^k \frac{a_i \text{cov}(S_i)}{b_i} \right) x$$

Thus (1) turns into a binary minimization with a single linear constraint:

$$\begin{array}{ll} \min & cx \\ \text{subject to} & \left(\sum_{i=1}^k \frac{a_i \text{cov}(S_i)}{b_i} \right) x \geq K \\ & x \in \{0, 1\}^n \end{array} \quad (2)$$

“All or nothing” model for the minimal cost problem. Let us introduce a new variable vector $z = (z_1, \dots, z_k)$ defined in the following manner:

$$z_i = \begin{cases} 1 & \text{if } a_i x = b_i \\ 0 & \text{if } a_i x < b_i \end{cases}$$

In other words, $z = \max\{Ax - b + e, 0\}$, where $e = (1, 1, \dots, 1)$ and the maximization is made componentwise. Using this vector problem (1) can be written in the following manner

$$\begin{array}{ll} \min & cx \\ \text{subject to} & vz \geq K \\ & z = \max\{Ax - b + e, 0\} \\ & x \in \{0, 1\}^n \end{array}$$

It is easy to see that this is equivalent to

$$\begin{array}{ll} \min & cx \\ \text{subject to} & z_i \leq \frac{a_i}{b_i} x \quad i = 1, \dots, k \\ & z \geq Ax - b + e \\ & vz \geq K \\ & x \in \{0, 1\}^n \\ & z \in \{0, 1\}^k \end{array} \quad (3)$$

This is a binary minimization with linear constraints. The number of the variables is $n + k$, the number of the constraints is $2k + 1$.

“One is enough” model for the minimal cost problem. This model can be handled similarly to the previous one. Let now z be the following vector:

$$z_i = \begin{cases} 0 & \text{if } a_i x = 0 \\ 1 & \text{if } a_i x \geq 1 \end{cases}$$

namely $z = \min\{Ax, e\}$. In this case (3) can be transformed into the following problem:

$$\begin{aligned}
 & \min \quad cx \\
 & \text{subject to} \quad z_i \geq \frac{a_i}{b_i}x \quad i = 1, \dots, k \\
 & \quad \quad \quad z \leq Ax \\
 & \quad \quad \quad vz \geq K \\
 & \quad \quad \quad x \in \{0, 1\}^n \\
 & \quad \quad \quad z \in \{0, 1\}^k
 \end{aligned} \tag{4}$$

Above we assumed homogenous models, but we are not restricted to them. We defined the functions f_i separately for each subpurpose, so one can imagine an optimization problem where different subpurposes follow different models. These "mixed" models do not cause new problems, that is why we omit them.

Our second optimization problem is to find a **maximal coverage** test set supposing an upper bound (L) for the cost.

$$\begin{aligned}
 & \max \quad cov(\mathcal{T}) \\
 & \text{subject to} \quad c(\mathcal{T}) \leq L \\
 & \quad \quad \quad \mathcal{T} \subset \{T_1, T_2, \dots, T_n\}
 \end{aligned}$$

This optimization problem, just like the previous one, can be formulated as a binary minimization problem with functions f_i :

$$\begin{aligned}
 & \max \quad \sum_{i=1}^k f_i(a_i x) \\
 & \text{subject to} \quad cx \leq L \\
 & \quad \quad \quad x \in \{0, 1\}^n
 \end{aligned} \tag{5}$$

Without further details let us see the formulae for the three models.

"Linear" model for the maximal coverage problem.

$$\begin{aligned}
 & \max \quad \left(\sum_{i=1}^k \frac{a_i cov(S_i)}{b_i} \right) x \\
 & \text{subject to} \quad cx \leq L \\
 & \quad \quad \quad x \in \{0, 1\}^n
 \end{aligned} \tag{6}$$

“All or nothing” model for the maximal coverage problem.

$$\begin{aligned}
 & \max \quad vz \\
 \text{subject to} \quad & z_i \leq \frac{a_i}{b_i} x \quad i = 1, \dots, k \\
 & z \geq Ax - b + e \\
 & cx \leq L \\
 & x \in \{0, 1\}^n \\
 & z \in \{0, 1\}^k
 \end{aligned} \tag{7}$$

“One is enough” model for the maximal coverage problem.

$$\begin{aligned}
 & \max \quad vz \\
 \text{subject to} \quad & z_i \geq \frac{a_i}{b_i} x \quad i = 1, \dots, k \\
 & z \leq Ax \\
 & cx \leq L \\
 & x \in \{0, 1\}^n \\
 & z \in \{0, 1\}^k
 \end{aligned} \tag{8}$$

We have transformed each type of the optimal selection problem into linear integer (binary) optimization problems. These problems, in theory at least, can be solved by adequate mathematical methods. Although integer programming is NP-complete, we have not met a problem we could not solve so far. In the following sections we are going to present a protocol the ATS of which contains 668 Test Cases, and we will show that our method is able to find relatively small test set to execute. Though much bigger ATSs rarely exist, we are ready to use heuristic methods in order to find suboptimal solutions in case we are not able to find optimal ones.

3. REALIZATION

In the previous section we introduced subpurposes. In this section we would like to show how the theory based on them can be turned into practice. We have to decide first what segment of the protocol is going to play the role of subpurposes.

Requirements describe well-defined parts of protocol behaviour. They can be classified according to their internal properties, which consists of the messages and their parameters, the time periods, the states, and the protocol mechanism [7]. To test the messages (Protocol Data Units, PDUs) and their parameters is the most fundamental control. That is the reason why we choose the PDUs to be the subpurposes. Of course, the whole purpose is not covered entirely by the PDUs but they form a good approximation, and unlike requirements the PDUs related to a TC are easy to detect. The connection of purposes, requirements and PDUs is illustrated in Figure 1.

Figure 1 The connection of Test Purposes (TP), requirements (R) and Protocol Data Units (PDU)

Although PDUs are the most natural candidates for subpurposes, sometimes it is worth stepping one layer up or down if it results in more efficient selection. That is, choosing the PDU types or the parameters of the PDUs respectively to be subpurposes. Which the most appropriate subpurpose is depends on the structure of the ATS.

The fact that we choose PDUs to be subpurposes has an other important advantage, namely the Subpurpose-Test Case Incidence Matrix can be filled up automatically using the TTCN-mp format of the ATS. We implemented a program that is able to find the PDUs used in the TCs and to create the incidence matrix independently of the size of the ATS. This program can also detect PDUs that are used in the Test Steps or Defaults appearing in the Test Case. So the PDUs related to a TC are sent either in its main body or in a referred Test Step or Default. The program deletes the all zero rows if there are any, and merge the rows which are equal, because in this latter case the PDUs related to these rows can be supposed to compose one subpurpose together.

Determining the cost of the tests and the coverage of the subpurposes is up to the test laboratories' individual preferences, but we feel that choosing them to be the same for each TC and PDU respectively is the most natural solution. As for the coverage this is because at first sight there is no theoretical basis to distinguish between the PDUs. The costs can be chosen to be the same because the most time consuming task during the conformance assessment process is preparation and parametrization, the time of which is in direct proportion to the number of the executed tests.

We introduced three different models regarding the coverage: the "linear", the "all or nothing", and the "one is enough" models. Supposing that the PDUs play the role of the subpurposes the "linear" model means that the more PDUs

Figure 2 The process of selection

were sent in the executed TCs, the bigger coverage we obtain. In the "all or nothing" model we regard a PDU to be checked if we execute all the TCs related to it. In the case of the "one is enough" model a Test Case is already assumed to be checked if we execute only one TC that contains the PDU. This third model, we think, is not adequate.

Figure 2 illustrates the process of selection. The main steps are (1) creating the Subpurpose-Test Case Incidence matrix using the TTCN-mp format of the ATS; (2) determining the cost and coverage functions, deciding which model will be used (minimal cost/maximal coverage, "linear", "all or nothing" or "one is enough"), and inputting the cost or coverage bound; (3) loading the selected TCs into the tester. For a protocol the first step should be done only once. If the incidence matrix is available, we can try and carry out the optimization for different cost, coverage functions, or different models and bounds in order to find the best test set that meet our aims.

4. EXPERIMENT RESULTS

Having described the method let us look at the experiments now. We applied the method to the ATS of ISDN DSS1 Layer 3 protocol [4]. We decided to use this protocol for it is well-known, widely used, and relatively complex.

First we applied the above mentioned program to find the PDUs, the Test Cases, the Test Steps (TS) and the Defaults (DEF) defined in the ATS. The standard contains 229 PDUs, 668 TCs, 46 TSs and 5 DEFs. The program also determines which PDUs are used in which TC, TS or DEF, and which TSs or DEFs are included in which TC. Using this information we could create the PDU-TC incidence matrix. The $(i, j)^{th}$ entry of this matrix is 1 either if the i^{th} PDU is sent in the j^{th} TC or if it is sent in a TS (DEF) that is referred in the j^{th} TC or, even deeper, if it is sent in a TS (DEF) referred in an other TS (DEF) which is a part of the j^{th} TC, and so on. Then we merged the equal rows and deleted the all zero rows (there were some). By means of these operations we reduced the size of the incidence matrix to 192 rows (subpurposes=PDUs or more PDUs together) and 668 columns (Test Cases).

Figure 3 Minimal cost in "linear" model

Figure 4 Maximal coverage in "linear" model

For the reasons we have mentioned in the previous section we defined the cost and the coverage to be the same for each TC and subpurpose respectively, and we chose the "linear" and the "all or nothing" models in optimization. To solve the optimization problems we used CPLEX MIP Solver [8].

4.1 RESULTS USING THE "LINEAR" MODEL

We calculated the minimal cost test set for coverage bounds $K = 5, 10, \dots, 185, 190, 192$ in problem (2). The results are shown in Figure 3. As it can be seen, increasing the coverage bound the cost of the optimal test set does not increase linearly. This means that using our method we can obtain better (shorter in time) test sets to execute than we would get if we chose it at random. In fact, our method gives us the best possible test set within the constraints.

In problem (6) we calculated the maximal coverage test set for cost bounds $L = 25, 50, 75, \dots, 625, 650, 653, 668$. Figure 4 shows the results. There are two things that are worth remarking. The first is the same phenomenon we mentioned above: we got better results than they would have been in case of random selection. The second is that we reached the whole coverage at $L = 653$, so 15 Test Cases are unnecessary. Indeed, we found 15 empty TCs, which do not contain any PDUs.

Let us see a specific example. If we let 10% of the coverage be lost (so $K = 173$ in the minimal cost problem), then it is enough to execute only 331 of the 668 tests (49.5%). While the whole coverage of these TCs is 173.11 the individual coverages of different subpurposes vary between 0 and 1. There are 155 subpurposes the coverages of which are equal to 1, which means that they are checked completely. All the subpurposes that appears in less than 11 TCs are that kind. The remaining 37 incompletely checked subpurposes are mostly those that appear in lots of TCs (the average number is 267). If a PDU appears

Figure 5 The number of the related TCs to the 37 incompletely checked PDUs, and the number of the checked ones among them.

in such an enormous number of TCs, then it is possibly sent in a Test Step or Default that is used in most of the Test Cases. Therefore a coverage of 50% for this PDU is a promising result. Figure 5 presents how many TCs are related to these 37 subpurpose and how many times they were checked.

Let us examine now the results from the point of view of the TCs. For example the 32nd Test Case - L3N_N00_V_032 - is not selected. Figure 6 presents this TC in TTCN.

This TC first initiates a parallel test component (PTC1_IN), takes the IUT into Null call state N00 with the preamble L3N_PR_N00, indicates a SETUP message, then checks whether the valid (SU_R10) or the invalid (SU_R12) PDU is received, checks the final state (Call Present call state N06) with L3N_CS(6,1), and in the end finishes with the postamble L3N_PO(1). All these stages can be characterized by PDUs or Test Steps. We can characterize the Test Steps by the PDUs that appear in them, therefore the subpurposes related to this TC characterize its behaviour. Of course, they do not tell us much about the structure, but they do carry an important part of the information contained by the Test Case.

Test Case Dyamic Behaviour					
Test Case Name : L3N_N00_V_032 Group : N00/Valid/Incoming_point_to_point/ Purpose : Ensure that the IUT in the Null call state N00, to indicate the arrival of a call to a point-to-point configuration, sends a SETUP message using the point-to-point data link and enters the Call Present call state N06. Configuration : CONFIG1 Default : L3N_DEF(1) Comments : subclause 5.2.1					
Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		CREATE(PTC1:PTC1_IN)			(1)
2		+L3N_PR_N00			preamble N00
3		CPA1ICP_M START TWAIT	S_SETUP		(2)
4		L0?SETUPr (CREF := SETUPr.mun.cr.cr_r) CANCEL TWAIT	Sr(SU_R10(CHlb_R1,CHlp_R1))	(P)	(3)
5		+L3N_CS(6,1)			state = 6?
6		L0?SETUPr (CREF := SETUPr.mun.cr.cr_r) CANCEL TWAIT	Sr(SU_R12)	(F)	(4)
7		+L3N_PO(1)			postamble N0
8		?TIMEOUT TWAIT		I	no response
Detailed Comments : (1) The slave component PTC1 is started. (2) This coordination message indicates to PTC1 to send a SETUP message. (3) A valid SETUP message is received. (4) A SETUP with an invalid Channel identification information element is received. The test fails.					

Figure 6 The TTCN format of the L3N_N00_V_032 Test Case.

The 225th Test Case - L3N_N06_I_002 - is selected. The purpose of this TC assumes that the IUT is in Call Present call state N06, and ensures it with the preamble L3N_PR_N06. This preamble is basically contains the 32nd TC. It can be said that the 225th TC is a continuation of the 32nd one, thus if it is executed, then the other one is (indirectly) executed as well. That means, although the 32nd TC is not selected (so it is not executed), we do check its purpose by executing the 225th TC.

The situation is very similar in the case of the 148th TC (L3N_N04_V_001). Its preamble L3N_PR_N04 and the TC itself together is the same word for word as the preamble L3N_PR_N0100. Therefore, though the 148th TC is not selected, every selected test which contains the latter preamble checks its purpose.

4.2 RESULTS USING THE "ALL OR NOTHING" MODEL

We calculated the minimal cost test set for coverage bounds $K = 5, 10, \dots, 185, 190, 192$ in problem (3) and the maximal coverage test set for cost bounds $L = 25, 50, 75, \dots, 625, 650, 653, 668$ in problem (7). Figure 7 and Figure 8 show the results.

The node limit of the branch-and-bound algorithm implemented in CPLEX was 20000. There are values of K and L for which we did not get optimal

Figure 7 Minimal cost in "all or nothing" model

Figure 8 Maximal coverage in "all or nothing" model

solutions with this small node limit. But the suboptimal feasible solutions give us attractive results. The values we got were better in each cases than we would have obtained by random selection.

5. CONCLUSIONS

We have presented the theory and practice of a new selection method which selects Test Cases from the Abstract Test Suite. We have performed experiments to explore its efficiency. The results show that the method can reduce the time of testing. We have applied the method to a fairly complex, widely used, real-life protocol, the ISDN DSS1 Layer 3 protocol. We find our method to be empirically promising, however, further experiments can be conducted for other protocols.

Further research may be devoted to the possible other subpurpose candidates, like the parameters. We plan to refine the method in order to make it capable to choose the best candidate.

References

- [1] S.T. Vuong, J.A. Curgus. On test coverage metrics for communication protocols. In *4th Int. Workshop on Protocol Testing System*, 1991.
- [2] J.A. Curgus, S.T. Vuong and J. Zhu. Sensitivity analysis of the metric based test selection. In *IFIP 10th Int. Workshop on Testing of Communicating Systems*, Cheju Island, Korea, 1997.
- [3] ISO/IEC 9646-3, Tree and Tabular Combined Notation (TTCN), 1998.
- [4] ETSI draft prETS 300 403-7 (1998): "Integrated Services Digital Network (ISDN); Digital Subscriber Signalling System No. one (DSS1) protocol; Signalling network layer for circuit-mode basic call control; Part 7: Abstract

Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma specification for the network".

- [5] ETSI Draft prTBR4 Integrated Services Digital Network (ISDN); Attachment requirements for terminal equipment to connect to an ISDN using ISDN primary rate acces.
- [6] T. Csöndes, S. Dibuz and B. Kotnyek. Test Suite Reduction in Conformance Testing, *Acta Cybernetica*, to appear in 1999.
- [7] K. Tarnay. *Protocol Specification and Testing*, Akadémiai Kiadó, Budapest, 1991.
- [8] CPLEX, Version 3.0, 1989-1994, CPLEX Optimization Inc.
- [9] ISO/IEC DIS 13245/ ITU-T Z.500: Formal methods in conformance testing, 1997
- [10] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*, Wiley, New York, 1988.