Analyzing Response Inconsistencies in Test Suites

Benjamin Zeiss and Jens Grabowski

Institute for Computer Science, University of Göttingen, Germany
{zeiss,grabowski}@cs.uni-goettingen.de

Abstract. Extensive testing of modern communicating systems often involve large and complex test suites that need to be maintained throughout the life cycle of the tested system. For this purpose, quality assurance of test suites is an inevitable task that eventually has an impact on the quality of the system under test as well. In this work, we present a means to analyze response inconsistencies in test suites. We define a response consistency relation and describe a method that identifies candidates for the analysis. Using these candidates, we find response inconsistent states. The applicability of this method is discussed for local test cases, local test cases with different response orders, and distributed test cases with concurrent behavior.

1 Introduction

Current industrial test suites for modern communicating systems are often huge in size and complex in their behavior. The tested devices are becoming increasingly sophisticated and at the same time, such devices have to become more reliable than ever before. Extensive testing nowadays involves not only the specification, the selection, and the execution of test cases, but also includes the maintenance of test suites throughout the life cycle of the tested system. For this purpose, quality assurance of test suites is an inevitable task that eventually has an impact on the quality of the *System Under Test* (SUT) as well. There is always a reciprocal effect between the quality of the test suite and the quality of the tested system. In addition, just like normal software, manually written test suites suffer from software aging [1] and it is thus sensible to find quality issues in tests as early as possible.

In our previous work, we introduced a quality engineering approach for test specifications. For the quality assessment, a quality model is given and metrics are used to quantify quality characteristics of the test specification. The quality improvement is based on refactoring and smell detection for issue discovery [2, 3]. We prototyped this approach with our *TTCN-3 Refactoring and Metrics Tool* (TRex) [4] for the *Testing and Test Control Notation* (TTCN-3) [5]. An extension of the proposed quality engineering approach includes the reverse-engineering of test case models and a subsequent analysis of property patterns that should never occur in these models [6]. However, all these analyses were based on the test case as an entity and never regarded the test suite as a whole, i.e., the analyses disregarded how test cases relate to each other. In this paper, we present first results describing such inconsistencies in test suites that may indicate quality problems for the test suite.

Test suites are composed of multiple test cases. In this work, we contribute a method to analyze response inconsistencies. Response inconsistencies are situations that arise when multiple test cases in a test suite have the same stimuli sequences, but expect different responses after the stimuli have been sent. Using this response inconsistency criterion, we identify test cases that are contained in each other or that expect completely different messages despite their equivalent stimuli sequences. We discuss response inconsistencies for sequential test cases with a strict stimuli–response order and then for responses that may arrive in different orders as well as concurrent test behavior.

In general, the work on quality assurance of test cases or test suites is rare. Besides our own work mentioned above, Vega, Din, and Schieferdecker have worked on guideline checking of TTCN-3 test suites in order to improve the maintainability of test suites [7]. Besides guideline checking, they have worked on quality measures for test data [8, 9]. Together with Zeiss, Neukirchen and Grabowski, they participated in the definition of a quality model for test specifications [10].

The test suite consistency description by Boroday, Petrenko, and Ulrich [11] is related to our work. The paper describes mutual consistency criteria on test cases. In their description, two test cases are inconsistent when the expected SUT outputs for one state of their product are different. In our work, we deal with cases that they call *strongly inconsistent* and define a consistency relation that is based on the idea of weakening the bisimulation relation. In addition, we describe the analysis based on linear extensions of partially ordered models.

Numerous works on test case selection deal with similarity measures between test cases. For example, Cartaxo et al. [12] describe a similarity function based on the observed number of identical transitions. Similarly, Alilovic-Curgus and Vuong [13] have proposed a distance metric that penalizes mismatching symbols in execution sequences. The overall number of different proposals to measure test case similarity for test case selection are too high to list. Books such as [14] by Utting and Legeard provide a general overview on the typical criteria involved and the corresponding literature. To our knowledge, there is no work that bases a similarity measure on equivalent stimuli sequences though. Equivalent stimuli sequences are primarily interesting for analyzing inconsistencies in a test suite rather than being a generic similarity measure.

The paper is structured as follows: in Section 2, we provide a basic definition for the model in which we represent our test cases. Sections 3 and 4 comprise the main contributions of this paper. Section 3 presents how we define response consistency and correspondingly response inconsistency between two test cases. Section 4 describes a method on how to analyze response inconsistencies by finding candidate pairs and then analyzing the candidates. Section 5 describes how the presented relations are applicable to test cases with different response orders. Section 6 discusses concurrent test cases. Finally, we conclude with a summary and an outlook in Section 7.

2 Test Suite Model

We first introduce a basic transition system that we use to describe our test cases $T_1, T_2, \ldots, T_n$ in a test suite TS. Our base model is the *Labeled Transition System* (LTS)

with actions partitioned into inputs and outputs as described, for example, by Tretmans [15]. In addition, we describe unobservable actions by their own partition.

Definition 1 (Labeled Transition System (LTS) with Inputs and Outputs). *An LTS T is defined by the tuple (S, A, λ, s_0) where*

- *S is a finite and non-empty set of states,*
- *A is a set of actions that is partitioned into the sets of input actions A_I, the set of output actions A_O, and a set unobservable actions A_U with $A = A_O \cup A_I \cup A_U$, $A_I \cap A_O \cap A_U = \emptyset$.*
- *λ is a transition relation with $\lambda \subseteq S \times A \times S$,*
- *s_0 is the initial state.*

A transition from the set λ is written as triple (s, a, s') or $s \xrightarrow{a} s'$.

We also refer to the elements of the four-tuple by using them as index of T, e.g., T_S refers to the set of states in T. The elements of each set may have an upper index to refer to the model they belong to, for example, $s_i^{T_S}$ refers to a state $s_i \in T_S$. To ease the distinction between input actions and output actions, we also use the notation $?a$ if $a \in A_I$ and $!a$ if $a \in A_O$. We use the notation $p!a$ or $p?a$ if a message a sent through a channel p or received through a channel p respectively. Since channels are not explicitly a part of our model, the channels can be interpreted as a label prefix. We continue by providing various further definitions that extend the formal framework. Note: in this paper, we discuss properties of test cases as subject rather than properties of the tested system. Therefore, we switch the meaning of inputs and outputs and take the view of the test case to have a more intuitive understanding from its perspective. This means that inputs in our models are inputs to the test case, i.e., responses from the system whereas outputs are outputs to the system, i.e., the stimuli.

Definition 2 (Path). *A path $s_i \overset{\sigma}{\Rightarrow} s_n$ of an LTS T is a finite and non-empty sequence $\langle s_i, a_i, s_{i+1}, a_{i+1}, ..., a_{n-1}, s_n \rangle$ with $i, n \in \mathbb{N}$ such that the transitions $s_j \xrightarrow{a_j} s_{j+1}, j \in \mathbb{N}, i \leq j < n$ exist in λ.*

Definition 3 (Traces). *A trace σ in an LTS T is a finite sequence $\langle a_i, a_{i+1}, ..., a_{n-1} \rangle$ such that the sequence $\langle s_i, a_i, s_{i+1}, a_{i+1}, ..., a_{n-1}, s_n \rangle$ is a path in T. We denote the set of all traces over a set of actions A with A^*. We concatenate actions to denote action sequences using the $\cdot$ sign, e.g., $?a \cdot !b \cdot ?c \cdot !d$ would denote a sequence of actions that is read from left to right.*

The notation $s \xrightarrow{a_1 \cdot a_2 \cdot ... \cdot a_n} t$ means that transitions $s \xrightarrow{a_1} s' \xrightarrow{a_2} \ldots \xrightarrow{a_n} t$ exist. We write $s \xrightarrow{a_1 \cdot a_2 ... \cdot a_n}$ if there exists a state t with $s \xrightarrow{a_1 \cdot a_2 ... \cdot a_n} t$.

With the double arrow, we denote paths that skip unobservable actions in A_U, i.e., if we have a path $s \xrightarrow{a_1 \cdot a_2 \cdot a_3 \cdot a_4} s'$ where $a_1, a_4 \in A_I \cup A_O$ and $a_2, a_3 \in A_U$, we may write $s \xRightarrow{a_1 \cdot a_4} s'$ for the abstracted sequence of observable actions. To be more concrete: $s \overset{\varepsilon}{\Rightarrow} s' \Leftrightarrow s = s'$ or $s \xrightarrow{a_1 \cdot a_2 \cdot ... \cdot a_n} s', a_i \in A_U$ and $s \overset{a}{\Rightarrow} s' \Leftrightarrow (s \overset{\varepsilon}{\Rightarrow} t \xrightarrow{a} t' \overset{\varepsilon}{\Rightarrow} s', a \in A_I \cup A_O \cup A_U) \vee (s = s'$ and $a \in A_U)$. Analogously, we use the notations $s \xRightarrow{a_1 \cdot a_2 \cdot ... \cdot a_n} t \Leftrightarrow s \overset{a_1}{\Rightarrow} s' \overset{a_2}{\Rightarrow} \ldots \overset{a_n}{\Rightarrow} t$ and $s \overset{\sigma}{\Rightarrow}$ iff there exists a state t with $s \overset{\sigma}{\Rightarrow} t$.

Furthermore, with $traces(T)$ we denote the set of of all traces that can be produced in model T from the start state s_0, i.e., $traces(T) := \{\sigma \in A^* | T \stackrel{\sigma}{\Rightarrow}\}$. Here, the LTS T refers to the initial state s_0 of T.

Definition 4 (Determinism). *An LTS is deterministic if $t = t'$ for paths $s \stackrel{\sigma}{\Rightarrow} t$ and $s \stackrel{\sigma}{\Rightarrow} t'$, for every $s,t \in S$.*

Definition 5 (Parallel Composition Operator). *Given two LTSs T_1 and T_2, the synchronous parallel composition $P = T_1 \| T_2$ is defined as follows:*

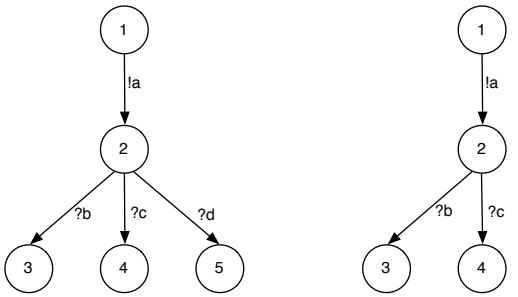
- $P_S = T_{1_S} \times T_{2_S}$.
- $P_A = T_{1_A} \cup T_{2_A}$.
- P_λ *is defined by the following inference rules:*
  - $(s,t) \xrightarrow{a} (s',t) \in P_\lambda$ *if* $s \xrightarrow{a} s' \in T_{1_\lambda}$ *and* $a \in T_{1_A} \setminus T_{2_A}$,
  - $(s,t) \xrightarrow{a} (s,t') \in P_\lambda$ *if* $t \xrightarrow{a} t' \in T_{2_\lambda}$ *and* $a \in T_{2_A} \setminus T_{1_A}$,
  - $(s,t) \xrightarrow{a} (s',t') \in P_\lambda, a \in A_U$ *if* $s \xrightarrow{a} s' \in T_{1_\lambda}$ *and* $t \xrightarrow{a} t' \in T_{2_\lambda}$ *and* $(a \in T_{1_{A_I}} \wedge a \in T_{2_{A_O}}) \vee (a \in T_{1_{A_O}} \wedge a \in T_{2_{A_I}})$.
- $P_{s_0} = (T_{1_{s_0}}, T_{2_{s_0}})$.

The third rule connects corresponding input actions with output actions of the same label where the action becomes an unobservable action in A_U. If we used a channel as label prefix for the synchronized input and output actions, we will indicate them in the synchronized action by separating channel and label with a ".", e.g., $p!a$ and $p?a$ becomes $p.a$. Note that due to our partitioning in inputs A_I, outputs A_O, and internal actions A_U, we are not in need of a special τ symbol that avoids multiway synchronization as actions in A_U are never synchronized. With the proposed composition operator and no queues in between, the send operations are blocking until a corresponding receive operation takes place.

3 Response Inconsistencies

The inconsistencies presented in this paper are based on the fact that the behavior of two different test cases T_1 and T_2 coincides up to a certain state s^{T_1} and respectively s'^{T_2}. Here, coinciding behavior intuitively means that there is a common action sequence σ such that the transitions $s_0 \stackrel{\sigma}{\Rightarrow} s$ can be found in T_{1_S} and $s_0 \stackrel{\sigma}{\Rightarrow} s'$ can be found in T_{2_S}. When s and s' respectively are states that expect only responses, they should expect the same responses — after all, the preceding action sequence between T_1 and T_2 matched and thus the responses should be the same.

Imagine the test case models illustrated in Figure 1a and 1b. Both models start with a stimulus $!a$ and thus have the same observable prefix. However, in test case T_1, the test case expects messages $?b$, $?c$, or $?d$ while test case T_2 only expects messages $?b$ or $?c$. T_2 expects a subset of the messages that T_1 expects and is essentially contained in T_1. In this situation, it is unclear why T_1 does not handle a possible incoming message d — it simply may be an inconsistency due to a human mistake or a redundancy. Three general cases can be distinguished how such test cases differ in a receiving state with the same preceding stimuli sequence:



(a) Test Case T_1 (b) Test Case T_2

Fig. 1: Different Responses

- the test cases expect the same messages,
- one test case handles a subset of the expected messages of another test case,
- one test case expects completely different messages than another test case. The receive sets are disjoint.

The example in Figure 1 illustrates case 2. The general underlying assumption, in this local scenario is that a test case is initiated by stimuli, responses follow in answer to the stimuli, and then again possibly a new stimuli–response sequence is initiated or the test case ends. In other words, we are assuming that we deal with test cases that have a repeating stimuli–response pattern. Within these patterns, we want to find contradictions in the responses when comparing test case pairs. Of course, the scenario of the presented Figure 1 is essentially the simplest possible case. In practice, we also deal with test cases that may possibly have varying response orders or concurrent behavior altogether.

For this purpose, we will define a binary relation that describes what a response consistent test case pair is. The relation is very general and is applicable for local test cases, test cases with varying response orders, and test cases with concurrent behavior. Informally, there are three distinct cases when we consider a test case pair response consistent:

- the test cases are observationally equivalent in their behavior,
- the test cases have entirely different stimuli sequences,
- the test cases have coinciding stimuli sequence prefixes and the stimuli responses within this prefix are consistent.

The first two cases can be considered as borderline cases for the last case. Obviously, if two test cases are observationally equivalent, there are no contradictions in the responses. The traditional relation that describes systems whose observable moves cannot be distinguished from each other is weak bisimulation [16]. We provide a definition of weak bisimulation defined for two separate transition systems (whereas the usual definition is given over a single transition system). Weak bisimulation, as opposed to strong bisimulation, abstracts from internal transitions in the behavior and thus only regards observable behavior for its notion of equivalence.

Definition 6 (Weak Bisimulation). *Given two LTSs T_1 and T_2, a binary relation $R \subseteq T_{1_S} \times T_{2_S}$ is a weak bisimulation iff the following conditions hold for every $(s,t) \in R$ and an action $a \in (T_{1_A} \cup T_{2_A})$:*

- *$s \xrightarrow{a} s' \in T_{1_\lambda}$ implies that there is a t' in T_{2_S} such that $t \stackrel{a}{\Rightarrow} t' \in T_{2_\lambda}$ and $(s',t') \in R$*
- *Symmetrically: $t \xrightarrow{a} t' \in T_{2_\lambda}$ implies that there is a s' in T_{1_S} such that $s \stackrel{a}{\Rightarrow} s' \in T_{1_\lambda}$ and $(s',t') \in R$*

We call two states s and t weakly bisimilar or $s \approx t$ iff $(s,t) \in R$. Similarily, we call two LTSs T_1 and T_2 weakly bisimilar, or $T_1 \approx T_2$, iff for every $s \in T_{1_S}$, there exists a $t \in T_{2_S}$ such that $s \approx t$ and for every $t \in T_{2_s}$, there exists an $s \in T_{1_S}$ such that $s \approx t$.

We assume that response contradictions can only occur when the stimuli sequences coincide. By stimuli sequences, we formally mean rewritten traces where only those actions are concatenated that are outputs from the test case. For this purpose, we define the *stimuliseq* operator that essentially slices all actions from a trace except the stimuli.

$$stimuliseq(a_1 \cdot a_2 \cdot \ldots \cdot a_n) := \forall a_i, a_j \in A_O, i < j : a_i \cdot a_j$$

Due to our assumption that we have a repeating stimuli–responses pattern, deviating stimuli sequences from the first stimulus indicate that the test case is intended to test something different. However, if the stimuli sequences coincide in both test cases having a common stimuli sequence prefix and then diverge into different stimuli suffixes, the responses to the coinciding stimuli still have to be consistent.

Definition 7 (Response Consistency (reco)). *Let T_1, T_2 be two LTSs. Then T_1 and T_2 are response consistent, or T_1 **reco** T_2 iff one of the following conditions hold:*

- *$\nexists \sigma \in traces(T_1), \varsigma \in traces(T_2) : stimuliseq(\sigma) = stimuliseq(\varsigma)$,*
- *$\exists s_0 \stackrel{\sigma}{\Rightarrow} s, \sigma \in traces(T_1), t_0 \stackrel{\varsigma}{\Rightarrow} t \in T_{2_\lambda}, \varsigma \in traces(T_2) : stimuliseq(\sigma) = stimuliseq(\varsigma)$, where for all $s \xrightarrow{a} s'$ with $a \in A_I^{T_1}$, there exists a $t \stackrel{a}{\Rightarrow} t'$ with $a \in A_I^{T_2}$ and vice versa: for all $t \xrightarrow{a'} t'$ with $a' \in A_I^{T_2}$, there exists an $s \stackrel{a}{\Rightarrow} s'$ with $a \in A_I^{T_1}$.*

In comparison to the weak bisimulation relation, the **reco** relation is weak in its condition. It essentially demands that for all states in a test case T_1 reachable by a common stimuli sequence and with an outgoing response transition, there must be corresponding response transition in T_2 in a state that is reachable by the same stimuli sequence and vice versa. In addition, the matching stimuli sequence condition also implies that test cases can be response consistent that drift apart in their stimuli at some point. The consistency criterion is only concerned with those states that are reachable by coinciding stimuli sequences.

As mentioned above, weak bisimulation and entirely different stimuli sequences are essentially borderline cases of the **reco** relation. In the following, we demonstrate that two response consistent test cases are weakly bisimilar when the stimuli sequences for all traces symmetrically match and the response orders are the same. It helps to differentiate **reco** and weak bisimulation and illustrates where the two relations meet.

Proposition 1. *If two test cases T_1 and T_2 with inputs and outputs fulfil the following conditions*

- T_1 ***reco*** T_2,
- $\forall s_0 \stackrel{\sigma}{\Rightarrow} s$ *with* $\sigma \in traces(T_1) \exists \varsigma \in traces(T_2) : stimuliseq(\sigma) = stimuliseq(\varsigma)$ *and vice versa:* $\forall t_0 \stackrel{\varsigma}{\Rightarrow} t$ *with* $\varsigma \in traces(T_2) \exists \sigma \in traces(T_1) :$ $stimuliseq(\varsigma) = stimuliseq(\sigma)$,
- *there is a relation* $R \subseteq S^{T_1} \times S^{T_2}$, *where for all pairs* (s,t) *reachable by the the same stimuli sequences and all* $(s,t) \in R$, $s \stackrel{a}{\rightarrow} s'$ *with* $a \in A_I^{T_1}$, *there exists a* $t \stackrel{a}{\Rightarrow} t'$ *with* $a \in A_I^{T_2}$ *with* $(s,t) \in R$ *and vice versa: for all* $t \stackrel{a'}{\rightarrow} t'$ *with* $a' \in A_I^{T_2}$, *there exists an* $s \stackrel{a}{\Rightarrow} s'$ *with* $a \in A_I^{T_1}$ *and* $(s,t) \in R$,

then T_1 and T_2 are weakly bisimilar.

Proof. Imagine two test cases T_1 and T_2 that are not weakly bisimilar. This means there is a state pair in the bisimulation relation $(s,t) \in R$ where there is a $s \stackrel{a}{\rightarrow} s' \in T_{1_\lambda}$ for which there is no $t \stackrel{a}{\Rightarrow} t' \in T_{2_\lambda}$ or there is a $t \stackrel{a}{\rightarrow} t' \in T_{2_\lambda}$ for which there is no $s \stackrel{a}{\Rightarrow} s' \in T_{1_\lambda}$. The fact that (s,t) is in the bisimulation relation implies that either $s = s_0$ and $t = t_0$ or that there are transitions $s_0 \stackrel{\sigma}{\Rightarrow} s$ and $t_0 \stackrel{\sigma}{\Rightarrow} t$ where for every observable transition, there is a bisimilar state pair that leads to (s,t). The hypothesis must hence be false for transitions following (s,t).

In the case that $a \in A_U$, then there is always a transition $s \stackrel{a}{\rightarrow} s'$ with a corresponding $t \stackrel{a}{\Rightarrow} t'$ and vice-versa as $s \stackrel{a}{\Rightarrow} s$ and $t \stackrel{a}{\Rightarrow} t$ always exist for internal actions (See Definition 3).

In the case that $a \in A_I$, then we know that for all $s \stackrel{a}{\rightarrow} s'$ with $a \in A_I^{T_1}$, there exists a $t \stackrel{a}{\Rightarrow} t'$ with $a \in A_I^{T_2}$ and vice versa: for all $t \stackrel{a'}{\rightarrow} t'$ with $a' \in A_I^{T_2}$, there exists an $s \stackrel{a}{\Rightarrow} s'$ with $a \in A_I^{T_1}$ if states s and t respectively are reachable by a common stimulus sequence. In addition, the relation R in the hypothesis ensures structural equivalence. Furthermore, according to the hypothesis, the stimuli sequences mutually exist for all traces in T_1 and T_2. Thus, there are always sequences $s_0 \stackrel{\sigma}{\Rightarrow} s$ and $t_0 \stackrel{\varsigma}{\Rightarrow} t$ where $stimuliseq(\sigma) = stimuliseq(\varsigma)$.

Finally, let $a \in A_O$ be an output action. If $s \stackrel{a}{\rightarrow} s' \in T_{1_\lambda}$, then there must be a $t \stackrel{a}{\Rightarrow} t' \in T_{2_\lambda}$. Otherwise, we must have traces σ and ς with $s_0 \stackrel{\sigma}{\Rightarrow} s'$ and $t_0 \stackrel{\varsigma}{\Rightarrow} t'$ where $stimuliseq(\sigma) \neq stimuliseq(\varsigma)$ which contradicts the hypothesis. The symmetric case can be shown analogously.

For input and output transitions, the hypothesis must be false in order to violate the bisimulation criteria for the pair (s,t). Hence, its contrapositive, i.e., the original statement, must be true.

If we once again take a look at the example in Figure 1, we notice that T_1 **reco** T_2 is false. The test cases do not differ in their stimuli sequences as both test cases start with a $!a$ transition. Therefore, we know that the third condition must hold. But it fails. There are no traces with the same stimulus sequence where the responses between the stimuli match completely. The traces where the stimuli sequence matches reach state 2,3,4, or 5 in T_1. However, a corresponding transition in T_2 for $?d$ is missing. For T_2, however, there are corresponding response transitions in state 2. Therefore, the condition fails.

4 Response Inconsistency Analysis

Given a a test suite TS which is a set of of test case models $T_1, T_2, \ldots, T_n$, we suggest a response inconsistency analysis involving the following steps:

- Finding candidate test case pairs for the analysis,
- For each candidate test case pair, we find states responding to stimuli that contradict each other in their responses.

In the following, we describe each of the two analysis steps in more detail.

4.1 Partial Stimuli Equivalent Test Case Pairs

The first step in finding response inconsistent test case pairs is to identify candidates for the analysis. To identify such candidates, we classify all test cases by the stimuli sequences that they can produce. The reasoning behind this is the stimulus–response pattern. We assume that responses happen as reaction to the stimuli sent to the system and thus stimuli sequences are independent from possible response contradictions while they still identify the gist of a test case, namely those behaviors of the test case that control the behaviors and reactions of the SUT. However, since there may be causal relationships between responses and subsequent stimuli, we define a partial order that is used to preserve these relationships.

Definition 8 (Linear Extension of a *Partially Ordered Set* (poset)). *Given a poset $(S_P, \leq)$, i.e., a binary relation "$\leq$" over a set S_P that is reflexive, antisymmetric, and transitive, a linear extension of $(S_P, \leq)$ is a total order S_T for which, whenever $x \leq y$ in P, $x \leq y$ also holds in S_T. We define $\mathcal{L}(S_P)$ to be the set of all possible linear extensions of S_P.*

We use posets for each unique stimuli sequence that essentially match all traces with the same stimuli sequences. These posets for each stimuli sequence order the stimuli among each other and responses to their corresponding stimuli, but not the responses themselves. Using these posets, we can require that there exists at least one common linear extension in the test case pair that leads to the next stimulus.

Definition 9 (Partial Stimuli Equivalence (psteq)). *Let T_1, T_2 be two LTSs and let $PO_{\sigma_O} := (A^{T_1}, \leq)$ and $PO_{\varsigma_O}(A^{T_2}, \leq)$ be partially ordered sets for each unique stimulus sequence σ_O and ς_O defined over the actions of T_1 and T_2 respectively. We then say that T_1 is partially stimuli equivalent to T_2, i.e., T_1 **psteq** T_2 if and only if all of the following conditions hold:*

- $\exists \sigma \in traces(T_1) \cap A^*$ *with a corresponding* $\varsigma \in traces(T_2) \cap A^*$ *such that* $stimuliseq(\sigma) = stimuliseq(\varsigma)$,
- $\forall a_i, a_j \in \sigma$ *with* $i < j : a_i, a_j \in PO_{stimuliseq(\sigma)}$ *and* $a_i \leq a_j$ *iff* $a_i, a_j \in stimuliseq(\sigma)$ *or* $a_i \leq a_j$ *iff* $a_i \in A_I^{T_1} \wedge a_j \in stimuliseq(\sigma)$,
- *vice-versa:* $\exists \varsigma \in traces(T_2) \cap A^*$ *with a corresponding* $\sigma \in traces(T_1) \cap A^*$ *such that* $stimuliseq(\varsigma) = stimuliseq(\sigma)$.,

- $\forall a_i, a_j \in \varsigma$ *with* $i < j : a_i, a_j \in PO_{stimuliseq(\varsigma)}$ *and* $a_i \leq a_j$ *iff* $a_i, a_j \in stimuliseq(\varsigma)$ *or* $a_i \leq a_j$ *iff* $a_i \in A_I^{T_2} \wedge a_j \in stimuliseq(\varsigma)$,
- $\exists \sigma \in traces(T_1) \cap A^*$, $\varsigma \in traces(T_2) \cap A^*$ *such that* $stimuliseq(\sigma) = stimuliseq(\varsigma)$ *such that the intersection of the linear extensions of the corresponding posets is non-empty, i.e,* $\mathcal{L}(PO_{stimuliseq(\sigma)}) \cap \mathcal{L}(PO_{stimuliseq(\varsigma)}) \neq \emptyset \vee (stimuliseq(\sigma) = \sigma \wedge stimuliseq(\varsigma) = \varsigma)$.

While the stimuli sequences are required to be equal in any case, the requirement that there is a linear extension in both T_1 and T_2 for each stimuli sequence that also includes responses that are ordered to their stimuli makes sure that causal relationships between responses and subsequent stimuli are preserved (see Section 5).

Using this partial stimuli equivalence criterion, we can partition the test cases in TS, i.e., $T_1, T_2, \ldots, T_n$ into partially output equivalent pairs $TS_C := \{(T_i, T_j) | \forall T_i, T_j \in TS : T_i \textbf{ psteq } T_j\}$.

4.2 Finding Response Inconsistent States

Given the set of candidates that have coinciding stimuli prefixes TS_C, we can now analyze test case pairs (T_i, T_j) for possible response inconsistencies. For this purpose, we define the set of inconsistent states $IS_i \subseteq S^{T_i}$ that contains those states of T_i that are inconsistent with T_j.

$$
\begin{aligned}
IS_i := \{ & s | s \in S^{T_i} : \\
& \exists \sigma, \varsigma \text{ with } stimuliseq(\sigma) = stimuliseq(\varsigma), s_0^{T_i} \stackrel{\sigma}{\Rightarrow} s^{T_i}, t_0^{T_j} \stackrel{\varsigma}{\Rightarrow} t^{T_j} \wedge \\
& \exists t := (s^{T_i} \xrightarrow{a} s'^{T_i}), a \in A_I^{T_i} \text{ for which there exists no } (t^{T_j} \xrightarrow{a} t'^{T_j}), a \in A_I^{T_j} \}
\end{aligned}
\tag{4.1}
$$

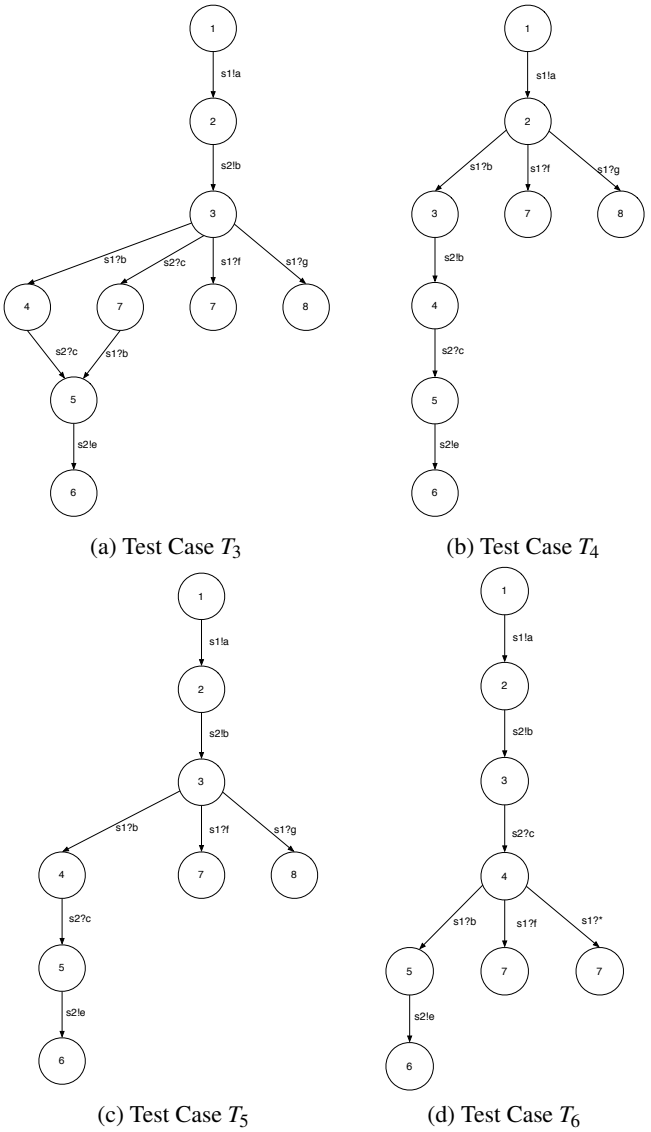
Informally, the conditions describe that there is a trace σ in T_i and ς in T_j that reach corrsponding states s_{T_i} and S_{T_j} by applying the same stimulus sequence. The state s_{T_i} reached is a state where only responses take place and there is a response transition with an input action a that does not exist in T_j.

Analogously, IS_j defines those states $IS_j \subseteq S^{T_j}$ that contains the states of T_j that are inconsistent with T_i, i.e., i and j are swapped. We then say that T_i and T_j are response inconsistent if $IS_i \cup IS_j \neq \emptyset$.

5 Response Inconsistency Analysis with Different Response Orders

The provided examples so far always assumed that the response events of the compared test cases are ordered in the same way. Similarly, we have only discussed local test cases without concurrent behavior so far. In general, we need to distinguish the following cases when we compare test cases for response inconsistencies:

- we compare a local test case to another local test case where both have the same response orders,



(a) Test Case T_3 (b) Test Case T_4

(c) Test Case T_5 (d) Test Case T_6

Fig. 2: Local Test Cases with Different Response Orders

- we compare a local test case to another local test case where both have different response orders,
- we compare a concurrent test case to a local test case,
- we compare two concurrent test cases.

The differentiation already indicates that there is a high degree of variety how test cases can be designed. Also, different test case designs can be used to model essen-

tially the same behavior. For example, test cases with concurrent behavior can produce the same behavior as a local test case when they are synchronized properly. Similarly, deterministic local test cases, can systematically regard different message receival orders. We explicitly want to state that we do not find every kind of test design mentioned above and in the following reasonable for real-world testing. However, they may occur and therefore we need to discuss them.

We continue the discussion by comparing test cases T_3 and T_4 regarding inconsistent responses (Figure 2). Both are local test cases which have the same stimuli sequences ($s1!a, s1!a \cdot s2!b$, and $s1!a \cdot s2!b \cdot s2!e$), but with a different response order. At first glance, it seems that these are valid candidates for the response inconsistency comparison, but in actuality, they are not. T_4 requires that $s1?b$ is received before the $s2!b$ stimulus is sent whereas T_3 sends both stimuli subsequently. T_3 suggests that the $s1!a$ and $s2!b$ stimuli are either independent or that the responses following $s2!b$ depend on both stimuli whereas in T_4, the stimulus $s2!b$ is follow-up behavior to the response $s1?b$, i.e., there is a causal relationship between them. The example illustrates the necessity for the additional partial order condition in the definition of the partial stimuli equivalence relation (Section 4). While the stimuli sequences match, there is no linear extension that matches for the stimuli sequence $s1!a \cdot s2!b \cdot s2!e$. The only linear extension of $PO_{stimuliseq(s1!a \cdot s2!b \cdot s2!e)}$ in T_4 is

- $s1!a \cdot s1?b \cdot s2!b \cdot s2?c \cdot s2!e$.

The two linear extensions of $PO_{stimuliseq(s1!a \cdot s2!b \cdot s2!e)}$ in T_3 are

- $s1!a \cdot s2!b \cdot s2?c \cdot s1?b \cdot s2!e$, and
- $s1!a \cdot s2!b \cdot s1?b \cdot s2?c \cdot s2!e$.

By regarding linear extensions of the responses in between the stimuli, we notice that in T_3 the $s!b$ transition would not happen without a preceding $s1?b$ transition. Since the linearization sets for the examined stimuli sequence are disjoint, T_3 and T_4 are not regarded as partially stimuli equivalent. The **reco** relation also fails for these two test cases as the possible responses after the $s1!a$ stimulus in T_4 cannot be consumed in T_3 after the same stimulus.

The situation is different for test cases T_5 and T_6. They have again the same stimulus sequences $s1!a$, $s1!a \cdot s2!b$, and $s1!a \cdot s2!b \cdot s2!e$. For $s1!a$, there are no responses and thus the stimuli sequence equals its trace with responses. The same is the case for $s1!a \cdot s2!b$. However, for the stimuli sequence $s1!a \cdot s2!b \cdot s2!e$, we have the following linear extensions for T_5:

- $s1!a \cdot s2!b \cdot s1?b \cdot s2?c \cdot s2!e$, and
- $s1!a \cdot s2!b \cdot s2?c \cdot s1?b \cdot s2!e$.

For T_6, the linear extensions are

- $s1!a \cdot s2!b \cdot s1?c \cdot s1?b \cdot s2!e$, and
- $s1!a \cdot s2!b \cdot s1?b \cdot s1?c \cdot s2!e$

The responses for $s2!b$ are ordered before the $s2!e$ action and have no order among each other. Therefore, T_4 and T_5 are in fact not only partially stimuli equivalent, but stimuli

equivalent. The actual determination of the inconsistent states is working independent from any orderings. In T_5, the states expecting responses reachable through the stimuli sequence $s1!a \cdot s2!b$ are state 3 and 4. For each transition in state 3 and 4, there are corresponding transitions in T_6 that are reachable by the same stimuli sequence (in T_6, those states are states 3 and 4). Hence, there are no response inconsistencies in T_5 and T_6. Taking a look at the **reco** relation also exhibits that T_5 and T_6 are response consistent. After the $s1!a$ stimulus, both test cases do not expect any responses. Once the $s1!a \cdot s2!b$ sequence has been sent, there are states in both test cases that expect the $s1?b, s1?f, s1?g$, and $s2?c$ responses. Finally, both test cases again do not expect any responses after $s1!a \cdot s2!b \cdot s2!e$.

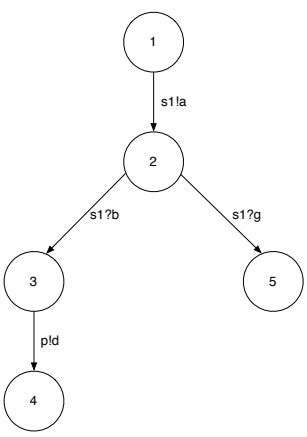
The latter example illustrates that the **reco** relation is a weak criterion in comparison to weak bisimulation due to its disregardment of structural properties within the response receival orders. This is intended as on the one hand, we cannot be sure that the test developer did not confuse orders by accident — this always needs to be validated by hand. On the other hand, this kind of freedom within the response orders is necessary to analyze test cases with concurrent behavior.

6 Response Inconsistency Analysis in Concurrent Test Cases

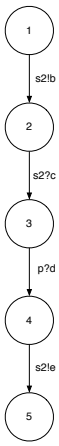
Nowadays test cases are often written with *Parallel Test Component*s (PTCs) that are executed concurrently, they have queued messages, and ports. Here, the determination of inconsistent responses is not as intuitive since the behavior is defined by composite models where non-determinisms can happen easily due to their interleaving structure.

Figure 3 illustrates an example for a concurrent test case. Figure 3a shows the *Main Test Component* (MTC) and Figure 3b shows the PTC of a test case. In this test case, there are three channels involved: $s1$ and $s2$ are channels connected to the SUT while channel p is a connection between the MTC and the PTC. A common paradigm is that a PTC is placed at each channel of the SUT and hence, the MTC communicates with the SUT via channel $s1$ and the PTC communicates with the SUT via channel $s2$. The PTC can only send the message $s2!e$ when its behavior is synchronized with the MTC through $p?d$ and $p!d$, i.e., the message must only be sent if $s1?b$ was received in the MTC. The composite model $T_7 = T_{7_a} \| T_{7_b}$ according to Definition 5 is presented in Figure 3c. To reduce the size of this model, we omitted all states and transitions that are unreachable from the start state. The order of the events depends on which events are independent from each other and on which events are dependant. For example, $s1!a$ must take place before $s1?b$ or $s1?g$, but $s2!b$ may take place any time in between. Furthermore, in the composite model, we have non-determinisms between inputs and outputs (e.g., in state $(2,1)$) and also between multiple outputs (e.g., in state $(1,1)$).

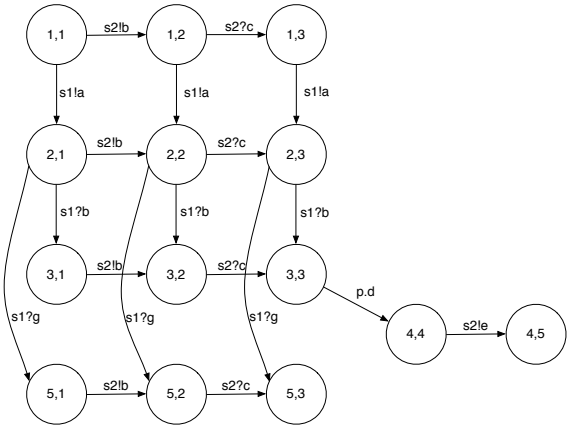
Based on the observations discussed in Section 5, we can compare such test cases against test case designs that are local, local with different response orders, and test cases that have concurrent behavior as well. There is no limitation how the response consistency and partial stimuli equivalence relations are applicable to interleaved models. We demonstrate this by example. We compare test cases T_3 and T_7. Both contain the stimuli sequences $s1!a$, $s1!a \cdot s2!b$, and $s1!a \cdot s2!b \cdot s2!e$. T_7 contains the additional stimuli sequences $s2!b$, $s2!b \cdot s1!a$, $s2!b \cdot s1!a \cdot s2!e$ that are not contained in T_3 and hence



(a) MTC T_{7_a}



(b) PTC T_{7_b}



(c) Composite Model T_7

Fig. 3: Test Case T_7 (Concurrent Test Case)

are left unregarded as the **reco** relation only regards traces that exhibit equal stimuli sequences. In every observable $s1!a$ trace in T_3, there are not corresponding responses. After the $s1!a \cdot s2!b$ stimuli sequence in T_3, there are states where every response can be consumed, i.e., $s1?b, s1?f, s1?g$, and $s2?c$. Finally, after the $s1?a \cdot s2!b \cdot s2!e$ stimuli sequence, there are again no states in which responses are consumable. In T_7, the consumable responses are quite different. After $s1!a$, there are states in which $s1?b$ and $s1?g$ are expected. The stimuli sequence $s1!a \cdot s2!b$ leads to states that can consume $s1?b, s1?g$, and $s2?c$. Finally, after $s1!a \cdot s2!b \cdot s2!e$, there may be no more responses in T_7. When evaluating the possible responses after traces with the same stimuli, we notice that the $s1!a$ sequence delivers a response mismatch where the response sets contradict each other, i.e., the response set for T_3 is empty while it is non-empty for T_7. Therefore, T_3 is not response consistent with T_7. However, T_3 and T_7 are partially stimuli

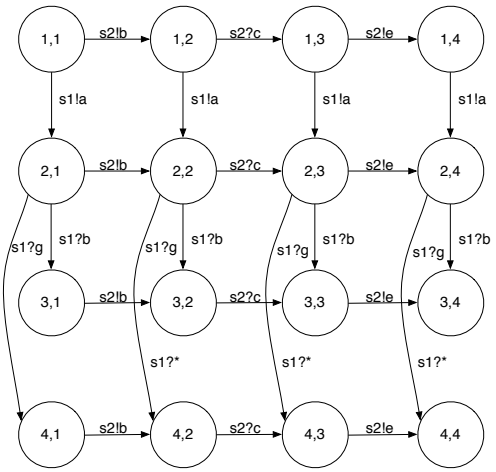


Fig. 4: Test Case T_8: Composite Test Case T_7 Without Synchronization

equivalent. They have both common stimuli sequences and for every common stimuli sequence, they have a common linear extension. Therefore, T_3 and T_7 are considered to be partially stimuli equivalent and thus classified as candidates for the response inconsistency analysis. However, due to missing responses after $s1!a$ in T_3, state $(2,1)$ is declared a response inconsistent state in IS_7 while $IS_3 = \emptyset$. Since $IS_7 \cup IS_3 \neq \emptyset$, T_3 and T_7 are response inconsistent.

The comparison between T_3 and T_7 indicates that the comparison between local and concurrent test cases will often exhibit inconsistencies when they are not necessarily considered a possible anomaly from the point of view of the test developer. Testers who write test cases with concurrent test behavior accept or even disregard the fact that the traces in subsequent executions of the same test may vary. For T_3, a possible interpretation is that the stimuli are independent from each other and the responses on $s1$ and $s2$ are independent from each other as well. However, the test case design is to send both stimuli $s1!a$ and $s2!b$ before expecting the responses for both stimuli rather than sending $s1!a$, then handling the responses for $s1!a$, and then sending $s2!b$ before handling the responses for $s2!b$. While the responses between T_3 and T_7 are strictly different (as the violated response consistency criteria suggests), it may be useful to find an even weaker criterion that declares such test cases as response consistent – after all, the local test case handles the all responses as well, just not in as many states as in the concurrent case.

We conclude the discussion with a comparison between two concurrent test cases. T_8 is essentially the composition of test cases T_{7_a} and T_{7_b} without the $p!d$ and $p?d$ transitions that synchronize the behavior. The purpose of this synchronization is to wait with the $s2!e$ transition in T_{7_b} until $s1?b$ or $p!d$ respectively took place in T_{7_a}. Removing this synchronization essentially means that $s2!e$ can take place anytime after $s2?c$ was received in T_{7_b}. As a result the $s1?b$ and $s1?g$ responses may still take place after the $s2!e$ transition happened in T_8 and thus the response set of the $s1!a \cdot s2!b \cdot s2!e$ stimuli

sequence is not empty for T_8, but there exist states which consume the responses $s1?b$ and $s1?g$ after the stimuli sequence took place. In T_7, such states do not exist after the same stimuli sequence and thus, T_7 and T_8 are not response consistent. However, there exist common linear extensions for all stimuli sequences and thus, they are stimuli equivalent and candidates for the analysis. Here, states $(1,4)$, and $(2,4)$ are considered inconsistent states in T_8.

7 Conclusion

We have presented what response consistency and inconsistency between test cases in a test suite means. For that purpose, we have defined a response consistency relation and described how to analyze test suites for response inconsistent test case pairs. For the analysis, we have discussed different test case designs (i.e., local behavior, different response orders, and concurrent test cases) and how the relations are applicable in these different scenarios.

Additional test suite based inconsistencies exist and may be subject of further studies. For example, we could analyze follow-up transitions after coinciding traces and check whether upcoming transitions are consistently responses or stimuli. Another interesting subject of analysis would be to include test case verdicts into the analysis, i.e., checking whether verdicts are consistent when the observable traces match. To handle different (or equal) message parameter values more carefully, it would be interesting to modify this approach for symbolic LTSs with constraints.

In addition, we are currently evaluating response inconsistencies on industrial-size test suites, such as the *European Telecommunications Standards Institute* (ETSI) test suite for the *Session Initiation Protocol* (SIP) protocol [17]. For this purpose, we reuse a refined version of our reverse engineering algorithm that was presented in [6] and intend to measure how often the described inconsistencies occur in practice. The reverse engineering algorithm constructs simplified models from TTCN-3 test specifications and therefore, there is the possibility for false positives when a response inconsistency is detected. This is a generic problem as the reverse engineering of semantically complete and correct transition systems from complex languages (like TTCN-3) is arguably not practically achievable with an acceptable effort. The practical impact of this problem remains to be evaluated. Furthermore, the analyses presented in this paper obviously suffer from the state explosion problem when we deal with parallel behavior. However, in comparison to system specifications, the complexity of test case behavior is (in most cases) rather low in our experience. Therefore, we suspect that there is a chance that the computation of the described comparisons is possible with todays machines without the necessity to define new heuristics or optimizations that may lower the precision of the approach.

Finally, we believe that the partial stimuli equivalence relation can also be adapted and used for test case selection, i.e., finding a smaller number of test cases by eliminating test cases with equivalent stimuli.

Acknowledgements This work is supported by Siemens AG Corporate Technology. We would like to thank Andreas Ulrich for feedback and inspiring discussions.

References

1. Parnas, D.: Software Aging. In: Proceedings of the 16th International Conference on Software Engineering (ICSE), May 16–21, 1994, Sorrento, Italy, IEEE Computer Society/ACM Press (1994) 279–287
2. Neukirchen, H., Zeiss, B., Grabowski, J., Baker, P., Evans, D.: Quality Assurance for TTCN-3 Test Specifications. Software Testing, Verification and Reliability (STVR) **18**(2) (2008)
3. Neukirchen, H., Zeiss, B., Grabowski, J.: An Approach to Quality Engineering of TTCN-3 Test Specifications. International Journal on Software Tools for Technology Transfer (STTT) **10**(4) (2008) 309–326
4. TRex Website: http://www.trex.informatik.uni-goettingen.de (Last Checked: May 15th, 2009)
5. European Telecommunications Standards Institute (ETSI): ETSI ES 201 873 V3.4.1 (2008-09): The Testing and Test Control Notation version 3; Parts 1–10. (2008)
6. Zeiss, B., Grabowski, J.: Reverse-Engineering Test Behavior Models for the Analysis of Structural Anomalies. In: TESTCOM/FATES 2008 Short Papers. (2008)
7. Din, G., Vega, D., Schieferdecker, I.: Automated Maintainability of TTCN-3 Test Suites Based on Guideline Checking. In: Software Technologies for Embedded and Ubiquitous Systems. Volume 5287 of Lecture Notes in Computer Science. (2008)
8. Vega, D., Din, G., Taranu, S., Schieferdecker, I.: Application of Clustering Methods for Analysing of TTCN-3 Test Data Quality. In: Proceedings of the 2008 The Third International Conference on Software Engineering Advances (ICSEA 2008). (2008)
9. Vega, D., Schieferdecker, I., Din, G.: Test Data Variance as a Test Quality Measure: Exemplified for TTCN-3. In: Testing of Software and Communicating Systems. Volume 4581 of Lecture Notes in Computer Science. (2007)
10. Zeiss, B., Vega, D., Schieferdecker, I., Neukirchen, H., Grabowski, J.: Applying the ISO 9126 Quality Model to Test Specifications – Exemplified for TTCN-3 Test Specifications. In: Proceedings of Software Engineering 2007 (SE 2007). Volume 105 of Lecture Notes in Informatics (LNI)., Köllen Verlag (2007)
11. Boroday, S., Petrenko, A., Ulrich, A.: Test Suite Consistency Verification. In: Proceedings of the 6th IEEE East-West Design & Test Symposium (EWDTS 2008), Ukraine. (2008)
12. Cartaxo, E.G., Neto, F.G.O., Machado, P.D.L.: Automated Test Case Selection Based on a Similarity Function. In: Proceedings of the 2nd Workshop on Model-Based Testing (MOTES 2007). (2007)
13. Alilovic-Curgus, J., Vuong, S.T.: A Metric Based Theory of Test Selection and Coverage. In: Proceedings of the IFIP TC6/WG6.1 Thirteenth International Symposium on Protocol Specification, Testing and Verification XIII, North-Holland (1993) 289–304
14. Utting, M., Legeard, B.: Practical Model-Based Testing – A Tools Approach. Morgan Kaufmann Publishers (2007)
15. Tretmans, J.: Test Generation with Inputs, Outputs, and Quiescence. In: Proceedings of Tools and Algorithms for Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science 1055. (1996)
16. Milner, R.: A Calculus of Communicating Systems. Volume 92 of Lecture Notes in Computer Science. Springer-Verlag (1980)
17. ETSI: TS 102 027-3: SIP ATS & PIXIT; Part 3: Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT). European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France (2005)