Towards a Family of Test Selection Criteria for Symbolic Models of Real-Time Systems

Diego R. Almeida[1], Alan Moraes[2,3],
Wilkerson L. Andrade[2], and Patrícia D. L. Machado[2]

[1] IFPE, Afogados da Ingazeira, PE, Brazil
diego.rodrigues@afogados.ifpe.edu.br
[2] Software Practices Laboratory (SPLab), UFCG, Campina Grande, PB, Brazil
{wilkerson,patricia}@computacao.ufcg.edu.br
[3] Informatics Center, UFPB, João Pessoa, PB, Brazil
alan@ci.ufpb.br

Abstract. In model-based testing, test cases are generated from a specification model. To avoid an exhaustive search for all possible test cases that can be obtained, usually an expensive and infeasible activity, test case generation may be guided by a test selection criterion. The objective of a test selection criterion is to produce a minimal test suite and yet effective to reveal faults. However, the choice of a criterion is not straightforward specially for real-time systems, because most criteria presented in the literature are general-purpose. Moreover, the relationship between general-purpose and specific criteria for real-time systems is not clear. In this paper, we investigate the criteria that can be applied for test case generation in the scope of model-based testing of real-time systems, specifically of Timed Input-Output Symbolic Transition Systems (TIOSTS) models. We formalize a family of 19 test selection criteria ordered by strict inclusion relation for TIOSTS models. The family combines general-purpose data-flow-oriented and transition-based criteria with specific reactive and real-time systems criteria. We also perform an empirical study to compare the effectiveness of selected criteria. Results of the empirical study indicate that failure detection capability of the generated test suite may vary, but differences are not significant for time failures. We conclude that more effective criteria for the model-based testing of real-time systems are still needed.

1 Introduction

Model-Based Testing is a testing approach that relies on the design of abstract models of an application to generate, execute and evaluate tests [10, 22, 27]. It has been applied with success in industry, with special emphasis in the avionic, railway and automotive domains [21].

Test case generation algorithms are based on test selection criteria that guide how to search for test cases and when to stop the test case generation process. Different test suites can be generated depending on the chosen test selection criterion [29]. They may vary in size, behavior coverage and failure detection

capability. While it is more likely that larger (and possibly with higher model coverage) test suites have better failure detection capability than smaller (and possibly with lower model coverage) ones, they are usually more expensive to manage and to execute. Therefore, test selection criteria need to establish how to guarantee the generation of test suites that are ultimately cost-effective.

Real-time systems are reactive systems whose behavior is constrained by time [18]. They usually combine concurrent execution of processes, consequently the nature of their failures is complex. The testing of these systems should uncover time-related faults that may require specific test cases to be exercised.

Most test selection criteria for real-time systems at model level are based on structural elements of a model behavior and its data usage [14]. Some specific test selection criteria for real-time systems have been proposed, such as covering all clock resets and all guard bounds [12]. However, the choice of a criterion is not straightforward, because the relationship between general-purpose and specific criteria for real-time systems is not clear [2].

In this paper, we investigate test selection criteria for real-time systems in the context of model-based testing. We focus on criteria that can be applied to transition systems, because they are usually the basis for conformance testing of real-time systems [17,28]. We use Timed Input-Output Symbolic Transition Systems (TIOSTS) models [5, 6], where system behavior is modeled as a transition system with data and time symbolically defined.

This paper makes two contributions. First, we formalize a family of 19 criteria partially ordered by strict inclusion relation for TIOSTS models. The family combines Transition-Based Criteria, data-flow-oriented criteria, Reactive Systems Criteria and Real-Time Systems Criteria. We prove inclusion or incompatibility whenever our family diverges from the known relationship in other models, because some relation between criteria change when applied to TIOSTS models.

Second, we conduct a controlled experiment to compare the effectiveness of selected criteria. The empirical study measures the size, the failure detection capability and the rate of failures detected by the size of the test suite of different criteria. In order to conduct the empirical study, we implemented a selection of criteria from the family using a depth-first search-based algorithm. Statistical analyses show that the criteria present different failure detection capability, although, significant differences cannot be observed for time-related failures. Furthermore, current specific criteria for real-time systems lack precision, i.e. they miss important failures, pointing to the need for further research in this area.

The paper is structured as follows. Section 2 introduces the TIOSTS model and test selection criteria for model-based testing of real-time systems. Section 3 formalizes a family criteria for TIOSTS. Section 4 presents an empirical study to compare selected criteria. Section 5 discusses related work. Finally, Section 6 presents concluding remarks along with pointers for further research.

2 Background

This section presents the symbolic model on which this work is based and introduces the concept of test selection criterion in the context of model-based testing.

2.1 Timed Input-Output Symbolic Transition System Model

Timed Input-Output Symbolic Transition System (TIOSTS) [5, 6] is a symbolic model for real-time systems that handles both data and time. The TIOSTS model was defined as an extension of two existing models: Timed Automata [3] and Input-Output Symbolic Transition Systems [15, 24]. Basically, a TIOSTS is an automaton with a finite set of locations where system data and time evolution are respectively represented by variables and a finite set of clocks. The transitions of the model are composed of a guard on variables and clocks, an action with parameters, an assignment of variables, and a set of clocks to reset.

Figure 1 shows an example of TIOSTS that models a machine for refilling a card for using the subway. Initially, the system is in the `Idle` location where it expects the `Credit` input carrying the desired `value` to refill, then this value is saved into the `refillValue` variable[4] and `balance` is initialized to zero.

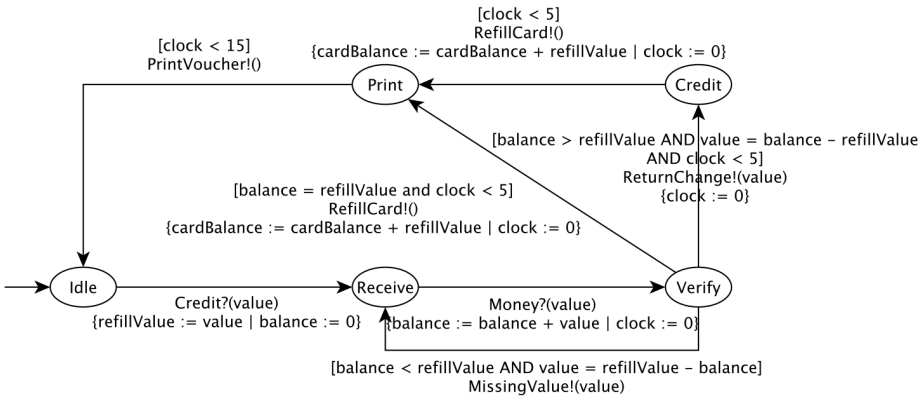

Fig. 1. TIOSTS model of a refilling machine

From the `Receive` location to `Verify` the client informs the amount to be credited to the card. This value is accumulated in the `balance` variable and the `clock` is set to zero. If the current balance is less than the desired value to refill, then the `Receive` location is reached again and the `MissingValue` output is emitted for informing the remaining value (the condition `value` =

[4] Action parameters have local scope, thus their values must be stored in variables for future references.

`refillValue − balance` contained in the guard means "choose a value for the `value` parameter that, with the values of `refillValue` and `balance` variables, satisfies the guard").

From the `Verify` location, if the `balance` is greater than `refillValue` some value must be returned to the client in less than 5 time units. After that, the `clock` is reset to zero again. Then, the `RefillCard` output action must be performed in less than 5 time units and the `cardBalance` is increased by `refillValue`. Otherwise, from `Verify`, if `balance` is exactly equals to `refillValue` the card must be refilled in less than 5 time units. Finally, from the `Print` location, the voucher must be printed in less than 15 time units and `Idle` location is reached again. A formal definition of TIOSTS models is presented in Definition 1 [5].

Definition 1 (TIOSTS). *A TIOSTS is a tuple* $W = \langle V, P, \Theta, L, l^0, \Sigma, C, \mathcal{T} \rangle$, *where:*

- V *is a finite set of typed variables;*
- P *is a finite set of parameters. For* $x \in V \cup P$, $type(x)$ *denotes the type of* x*;*
- Θ *is the initial condition, a predicate with variables in* V*;*
- L *is a finite, non-empty set of locations and* $l^0 \in L$ *is the initial location;*
- $\Sigma = \Sigma^? \cup \Sigma^!$ *is a non-empty, finite alphabet, which is the disjoint union of a set* $\Sigma^?$ *of input actions and a set* $\Sigma^!$ *of output actions. For each action* $a \in \Sigma$, *its signature* $sig(a) = \langle p_1, ..., p_n \rangle$ *is a tuple of distinct parameters, where each* $p_i \in P$ $(i = 1, ..., n)$*;*
- C *is a finite set of clocks with values in the set of non-negative real numbers, denoted by* $\mathbb{R}^{\geq 0}$*;*
- $\mathcal{T}$ *is a finite set of transitions. Each transition* $t \in \mathcal{T}$ *is a tuple* $\langle l, a, G, A, y, l' \rangle$, *where:*
  - $l \in L$ *is the origin location of the transition,*
  - $a \in \Sigma$ *is the action,*
  - $G = G^D \wedge G^C$ *is the guard, where* G^D *is a predicate over variables in* $V \cup set(sig(a))$[5,6] *and* G^C *is a clock constraint over* C *defined as a conjunction of constraints of the form* $\alpha \# c$, *where* $\alpha \in C$, $\# \in \{<, \leq, =, \geq, >\}$, *and* $c \in \mathbb{N}$,
  - $A = (A^D, A^C)$ *is the assignment of the transition. For each variable* $x \in V$ *there is exactly one assignment in* A^D, *of the form* $x := A^{D^x}$, *where* A^{D^x} *is an expression on* $V \cup set(sig(a))$. $A^C \subseteq C$ *is the set of clocks to be reset,*
  - $y \in \{$*lazy, delayable, eager*$\}$ *is the deadline of the transition,*
  - $l' \in L$ *is the destination location of the transition.* $\diamond$

The semantics of a TIOSTS is described by Andrade and Machado [5]. Next we define the concepts of *state*, *path* and *test case*.

[5] G^D is assumed to be expressed in a theory in which satisfiability is decidable.

[6] Let $set(j)$ be the function that converts the tuple j in a set.

Definition 2 (State of TIOSTS). *In TIOSTS model, a state is a tuple* $\langle l, v_1, ..., v_n, c_1, ..., c_m \rangle$*, which consists of a location* $l \in L$*, a specific valuation for all variables* $v_i \in V$*, and a valuation for all clocks* $c_i \in C$*.* $\diamond$

Definition 3 (Path). *A path is a finite sequence of transitions* $(t_1, ..., t_k)$*,* $k \geq 1$*, such that the destination location of transition* t_i *is equal to the origin location of the transition* t_{i+1} *for* $i = 1, 2, ..., k-1$*.* $\diamond$

Definition 4 (Test Case). *A test case is a deterministic TIOSTS* $TC = \langle V_{TC}, P_{TC}, \Theta_{TC}, L_{TC}, l^0_{TC}, \Sigma_{TC}, C_{TC}, \mathcal{T}_{TC} \rangle$*, where* $\Sigma^?_{TC} = \Sigma^!_{\mathcal{S}}$ *and* $\Sigma^!_{TC} = \Sigma^?_{\mathcal{S}}$ *(actions are mirrored w.r.t. specification), equipped with three disjoint sets of verdict locations* Pass, Fail*, and* Inconclusive*. Furthermore, each sequence from the initial location* l^0_{TC} *to some verdict location is a path.* $\diamond$

According to Definition 4, the execution of a test case can emit one of three possible verdicts: *Pass*, *Fail*, and *Inconclusive*. *Pass* means that some targeted behavior of the system under test has been reached, *Fail* means rejection of the SUT, and *Inconclusive* means that targeted behavior cannot be reached anymore.

Figure 2 is a test case for the TIOSTS model of the refilling machine. The test case aims to exercise the scenario where the system emits the `RefillCard` output when the amount to be credited to the card (`value_2`) is equal to desired value to refill (`value_1`). In this case, the verdict is *Pass*. If the amount to be credited to the card (`value_2`) is less than the desired value to refill (`value_1`), and the system emits the `MissingValue` output with parameter equals to `value_1` − `value_2`, then the verdict is *Inconclusive*. It is *Inconclusive* because this behavior is specified in the model, but it is not the scenario the tester would like to observe in the test case execution. The same applies to `ReturnChange` output action of the test case. All other cases lead to the implicit *Fail* verdict.

2.2 Test Selection Criteria for Real-Time Systems

In model-based testing, test cases are derived from a model which specifies the expected behavior of a system under test. A *Test Selection Criterion* defines which parts of the system are going to be tested, how often and under what circumstances they will be tested [29]. Test selection criteria are used for two main purposes: to measure the adequacy of the test suite with respect to the level of quality required by the context, and to stop the test generation process after the criterion is reached [29].

We conducted a systematic literature review to identify studies that address test selection criteria for real-time systems at model level [2]. We considered studies that a criterion was used at least as part of a test case generation process in the scope of transition and state-based systems [1,7,9,12–14,16,17,20,26,31].

The results of the review show that most general-purpose test selection criteria may be applied to models of real-time systems. There are also specific criteria for real-time systems proposed in the literature. However, there is a lack

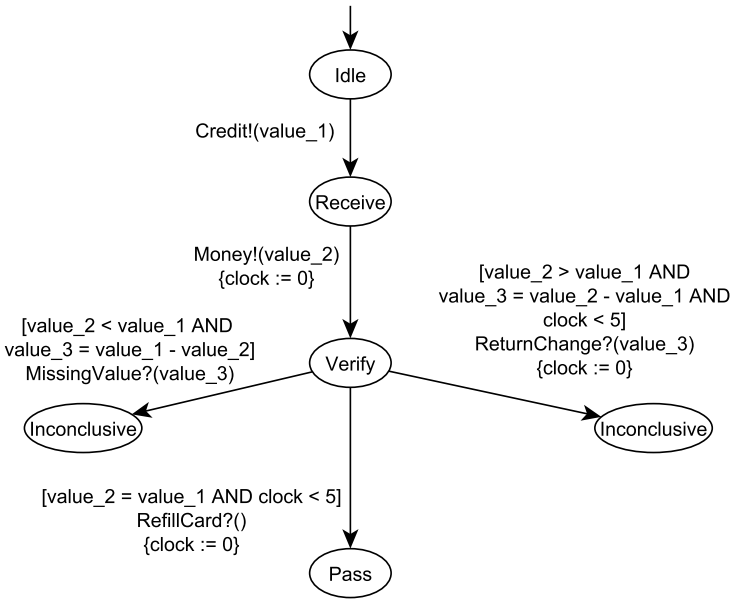

Fig. 2. A test case for the refilling machine.

of studies that investigate the theoretical and empirical relationship between criteria. The theoretical relationship could indicate the relative effort to satisfy a criterion, while the empirical evaluation could compare criteria effectiveness with respect to failure detection capability.

En-Nouaary [12] proposes a family of test selection criteria ordered by strict inclusion relation for Timed Input-Output Automata (TIOA). His family combines Transition-Based Criteria, Reactive Systems Criteria, and Real-Time Systems Criteria. But data-related criteria are not included because the TIOA model does not support data abstraction. Conversely, the TIOSTS model symbolically abstracts both time and data, thus data-related criteria can be applied to it. Furthermore, to the best of our knowledge, there is no work on test selection criteria for real-time systems at model level that evaluate the ability to reveal faults of selected criteria.

3 Towards a Family of Test Selection Criteria for TIOSTS

In this section, we propose a family of test selection criteria for TIOSTS models. We extend En-Nouaary's family [12] to include data-related criteria. We choose to include Data-Flow-Oriented Criteria, because they can be empirically evaluated with the same failure model employed to compare Transition-Based

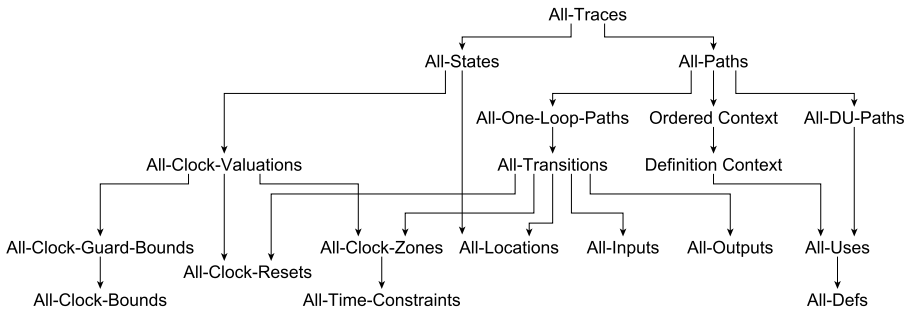

Fig. 3. Family of test selection criteria ordered by strict inclusion relation for TIOSTS models.

Criteria and Real-Time Systems Criteria in the next section. Thus our proposed family of criteria combines Transition-Based Criteria, Reactive Systems Criteria, Real-Time Systems Criteria and Data-Flow-Oriented Criteria. Table 1 describes the criteria we considered in this work.

Test selection criteria are often theoretically compared to each other by three relations: *strict inclusion*, *equivalence*, or *incompatibility* [23]. The relations are formalized in Definitions 6, 7 and 8 respectively.

Definition 5 (Inclusion Relation). *A criterion* c_1 includes *a criterion* c_2 *if any set of test cases that satisfies* c_1 *also satisfies* c_2 *[23].* ◇

Definition 6 (Strict Inclusion Relation). *A criterion* c_1 strictly includes c_2*, denoted by* $c_1 \Rightarrow c_2$*, if* c_1 *includes* c_2 *but there is a set of test cases that satisfies* c_2 *but does not satisfy* c_1*. Note that this is a transitive relation [23].* ◇

Definition 7 (Equivalence Relation). *A criterion* c_1 *is* equivalent *to a criterion* c_2 *if* c_1 *includes* c_2 *and* c_2 *includes* c_1*.* ◇

Definition 8 (Incompatible Relation). *A criterion* c_1 *is* incompatible *with a criterion* c_2 *if* c_1 *does not include* c_2 *and* c_2 *does not include* c_1*.* ◇

Our goal is to produce a sound family of test selection criteria partially ordered by strict inclusion relation. We do not intend to prove all equivalences or incompatibilities between criteria. To accomplish this, our strategy is i) to reuse the proofs of strict inclusion relations from other formalisms if they are also valid for TIOSTS; ii) to prove new strict inclusion relations resulting from the combination of classes of criteria; iii) to prove the exclusion of strict inclusion relations valid for other formalisms but not valid for TIOSTS. The proposed family is formalized in Theorem 1.

Theorem 1. *The family of criteria for TIOSTS is partially ordered by strict inclusion as shown in Figure 3. Furthermore,* $c_1 \Rightarrow c_2$ *iff it is explicitly shown to be so in Figure 3 or follows from the transitivity of the relationship.*

Table 1. Test Selection Criteria for TIOSTS models.

Criterion	Description
Transition-Based Criteria	
All-Locations [12, 16]	Every location of the model must be exercised by at least one test case.
All-Paths [12, 14]	Every path of the model must be exercised by at least one test case.
All-One-Loop-Paths [29]	Every loop-free paths through the model must be exercised, plus all the paths that loop at least once.
All-Transitions [12, 16]	Every transition of the model must be exercised by at least one test case.
All-States [1, 12, 31]	Every state of the model must be exercised by at least one test case.
All-Traces [12]	Every trace of the model must be included in the test suite.
Data-Flow-Oriented Criteria	
All-Defs [29]	At least one def-use pair(d_v, u_v) for every definition d_v must be exercised by at least one test case, i.e. at least one path from every definition to one of its use must be covered.
All-DU-Paths [29]	Every path for all def-use pairs(d_v, u_v) must be exercised by at least one test case, i.e. all paths from every definition d_v to every use u_v must be covered.
All-Uses [29]	Every def-use pairs(d_v, u_v) must be exercised by at least one test case, i.e. at least one path from every definition d_v to every use u_v must be covered.
Definition Context [14]	All paths from every context of definition of variable x to the definition of variable x must be exercised by at least one test case. The context of definition of the variable x are the transitions where the variables used to define the value of x are defined.
Ordered Context [14]	Similar to Definition Context, but the transitions context are listed in the order of their definitions.
Reactive Systems Criteria	
All-Inputs [9, 12]	Every input action of the model must be exercised by at least one test case.
All-Outputs [9, 12]	Every output action of the model must be exercised by at least one test case.
Real-Time Systems Criteria	
All-Clock-Bounds [12]	Every clock bound of the model must be exercised by at least one test case. The bound of a clock is the highest value that a clock can assume.
All-Clock-Guard-Bounds [12]	Every clock guard bound of the model must be exercised by at least one test case. This criterion is similar to All-Clock-Bounds but considering only the time guards.
All-Clock-Valuations [12]	Every clock valuation of the model must be exercised by at least one test case.
All-Clock-Resets [12]	Every clock reset of the model must be exercised by at least one test.
All-Clock-Zones [12, 26]	Every clock zone of the model must be visited through at least one test case, i.e. all transitions with clock resets or time guards must be covered.
All-Time-Constraints [12]	Every time guard of the model must be exercised by at least one test case.

Note: The criteria in this table are defined in terms of satisfiable paths, i.e. all data and time guards in a path must be satisfiable.

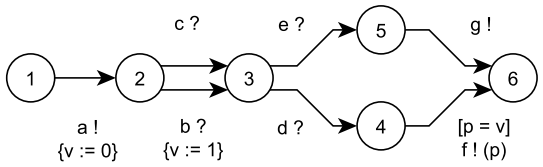

Fig. 4. A TIOSTS model to assist in the proof of All-DU-Paths $\not\Rightarrow$ All-Transitions.

Proof. We need to prove the relations All-States $\Rightarrow$ All-Locations, All-One-Loop-Paths $\Rightarrow$ All-Transitions, All-Transitions $\Rightarrow$ All-Clock-Resets, and All-DU-Paths $\not\Rightarrow$ All-Transitions. All other relations can be easily checked based on proofs already presented in the literature [12, 23, 29, 32].

1. All-States $\Rightarrow$ All-Locations. Proof follows directly from the definitions of the criteria. We recap that a state of a TIOSTS consists of a location, a specific valuation for all variables, and a valuation for all clocks. Since the All-States criterion demands the all states to be covered, thus All-States $\Rightarrow$ All-Locations.
2. All-One-Loop-Paths $\Rightarrow$ All-Transitions. Proof follows directly from the definitions of the criteria. The All-One-Loop-Paths criterion demands that all loop-free paths to be covered plus all loops at least one lap. Since all transitions must be either in a loop-free path or in a loop, thus All-One-Loop-Paths $\Rightarrow$ All-Transitions.
3. All-Transitions $\Rightarrow$ All-Clock-Resets. Proof follows directly from the definitions of the criteria. A clock reset happens within the assignment of a transition. The All-Transitions criterion demand that all transitions to be covered. Since all transitions with clock resets are a subset of all transitions, thus All-Transitions $\Rightarrow$ All-Clock-Resets.
4. All-DU-Paths $\not\Rightarrow$ All-Transitions. Proof by contradiction. Let's assume that All-DU-Paths $\Rightarrow$ All-Transitions. Consider the TIOSTS model in the Figure 4. The model has two def-use pairs: $\{(q_1, [true], a!, \{v := 0\}, \emptyset, q_2), (q_4, [p = v], f!(p), \emptyset, \emptyset, q_6)\}$ and $\{(q_2, [true], b?, \{v := 1\}, \emptyset, q_3), (q_4, [p = v], f!(p), \emptyset, \emptyset, q_6)\}$. The test cases[7] $\{\{a! \rightarrow c? \rightarrow d? \rightarrow f!(p)\}, \{a! \rightarrow b? \rightarrow d? \rightarrow f!(p)\}\}$ satisfy the All-DU-Paths criterion for this model, but the transitions $(q_3, true, e?, \emptyset, \emptyset, q_5)$ and $(q_5, true, g!, \emptyset, \emptyset, q_6)$ are not covered. Thus our assumption is incorrect, and All-DU-Paths $\not\Rightarrow$ All-Transitions. □

It is important to remark that the relation All-Uses $\Rightarrow$ All-Transitions does not hold for TIOSTS as it does for other models [23]. In fact, even All-DU-Paths $\not\Rightarrow$ All-Transitions for TIOSTS. This happens because a transition in TIOSTS may have neither a definition nor a use of a variable. Thus not all transitions will be covered by the All-DU-Paths criterion.

[7] The last transition in the test case leads to the *Accept* location.

4 Empirical Study

In this section we present a controlled experiment to compare the effectiveness of selected criteria. We follow the guidelines given by Wohlin, Runeson, Höst and Ohlsson [30]. The main goal of the empirical study is to investigate test selection criteria for real-time systems by observing the test suite generated from TIOSTS models according to a given criterion with respect to their size and failure detection capability from the point of view of the tester in the context of model-based testing. The research hypothesis is that different criteria may generate different suites of different sizes that may reveal a number of different failures.

Planning. We conducted this experiment in a research laboratory — an offline study with a specific context. As independent variable, we have the test selection criterion. The treatments are: All-One-Loop-Paths (AOLP), All-Transitions (AT), All-Locations (AL), All-Clock-Zones (ACZ), All-Clock-Resets (ACR), All-DU-Paths (ADUP), All-Uses (AU), and All-Defs (AD). Instead of evaluating all criteria of the family, we choose to evaluate the most used criteria found in our literature review. The selected criteria are representative of transition, time and data-related criteria.

The dependent variables are: i) size of the generated test suites ($Size$); and ii) failure detection capability, measured as the number of different failures that can be detected ($Failure$). From these dependent variables, for each treatment and object, we computed two values: i) the *percentage of failure*, defined as the relation between the $Failure$ value and the total of possible failures; ii) the *density of failure* as the relation between the $Failure$ and the $Size$ values. For the sake of simplicity, the hypotheses of the study are formulated based on these measures only as follows. Let $\%failure_i = \frac{Failure_i}{TotalFailures}$ and $density_i = \frac{Failure_i}{Size_i}$, where i is a test criterion and $Failure_i$, $Size_i$ are the average value of the correspondent dependent variables for each of the considered objects. Based on statistical testing, the null hypothesis is defined as the equality of all criteria, whereas the alternative hypothesis is defined as the difference between all criteria.

Regarding experimental design, this study consists of one factor and eight levels (eight test criteria) with six repetitions corresponding to six different models from three applications of real-time systems presented in the literature. We considered a confidence of 95% when deciding on hypothesis rejection. As input, for each criterion, only TIOSTS models are required. Dependent variables are computed automatically. Therefore, there is no human intervention and no subjects to be considered. Since there are no random choices involved, there is no need to compute the number of replications required.

The objects (TIOSTS models) were obtained from 3 different applications: i) Alarm System — Monitoring and actuation system that can detect invasion and also the presence of intruders in a building through door, window and movement sensors [25]; ii) Aircraft Attack System — System that controls attacks to specific land targets and also threat detection from a missile or another aircraft [19]; and

iii) Philips Audio Protocol — Protocol that defines control message exchanging for audio and video devices [8]. Moreover, collisions detection and delivery failure are handled. From these applications, we created six models and used them as input to the test case generator we implemented using a depth-first search-based algorithm. Table 2 presents the metrics of number of locations, transitions, transitions with time constraints, and transitions with data constraints of the considered models.

Table 2. Metrics of real-time system models used in the empirical study.

Model	Locations	Transitions	Trans. w/ time constraints	Trans. w/ data constraints
Alarm1	7	9	6	7
Alarm2	10	23	13	19
Aircraft1	11	13	8	6
Aircraft2	14	35	20	28
Protocol1	17	29	10	25
Protocol2	17	37	18	25

Notes. Alarm1: Alarm System without power failure. Alarm2: Simplified version of Alarm1 with power failure treatment. Aircraft1: Aircraft Attack System functionality only. Aircraft2: Simplified version of Aircraft1 with threat detection functionality. Protocol1: System without failure recovery. Protocol2: Simplified version of Protocol1 with failure recovery.

It is often difficult to associate a failure with a single fault at code level, because a failure may be caused by one or more faults. Therefore, for the purpose of this study and also to avoid undesired effects in the results, instead of the number of faults, we opt to measure failures — the number of different failures that can be detected by at least one test case in a given test suite. To allow for a reasonable sample of failures, we defined a failure model that contains potential failures which can be detected in a real-time system, particularly as a result of violation of time constraints. This model was based on previous studies such as the one performed by En-Nouaary, Khendek and Dssouli [11], and by Andrade and Machado [4]. Two basic types of failures were considered: time and behavior. The former is necessarily connected to non-conformity with time constraints, whereas the latter are more related to behavior non-conformity. For the sake of space, Table 3 presents only considered failures for the *Alarm2* model. Note that there is a different distribution of faults of the two types. The reason is that we do not aim to control this factor so that the distribution achieved is mostly a consequence of potential failures identified by considering each model.

Study execution was conducted according to the following process: 1) For each input model, a test suite was generated for each of the criteria; 2) For each test suite, each test case was analysed to determine whether it can fail according to the failure model; 3) For each test suite, failures from the failure model were marked when covered by the suite; 4) Data on study variables was collected; 5) *%failure* and *density* values were computed and analysis of results conducted.

Table 3. Failure Model for *Alarm2* model.

Failure	Type	Description
F04	Time	When power failure occurs, sensor status does not change.
F05	Time	When power failure is detected, the system does not change power supply on time.
F06	Behavior	After handling power failure, system does not resume execution as expected.
F07	Time	When power failure occurs, status change of movement sensor is not detected.
F08	Time	When power failure occurs, status change of window sensor is not detected.
F09	Behavior	After power failure handling, system does not detect an invaded room.
F10	Behavior	After power failure handling, alarm starts without invasion detection.

Threats to Validity. Measures were rigorously taken regarding data treatment and assumption with a confidence level of 95% that is usually applied in comparing studies. Also, to avoid the influence on the kind of applications in the obtained results, we have chosen specifications constructed by different authors — the models have different structural elements as illustrated in Table 2. Moreover, correctness of the implementation of the algorithms is critical to assess whether the results are reliable. Therefore, validation was throughly performed and, to avoid an inconsistent generation of suites, all algorithms are based on the same basic strategy — a depth-first search — where each criterion is applied as a stop condition. Furthermore, models used in the study may not be representative of all kinds of real-time systems, therefore, results can only be interpreted as specific. However, it is important to remark that they may be considered as an evidence since results confirm properties already known, particularly for the general criteria.

Results and Analysis. Data collected in the study as well as test cases generated can be downloaded from the study web site[8]. Figure 5 shows the box plots for the percentage of failure values and Figure 6 shows the box plot for the density of failures values. As the values do not follow a normal distribution, the Kruskal-Wallis test was performed and we obtained a *p-value* of 0.0388 for the percentage of failures. This means that we can reject the null hypotheses: when compared together the criteria present a different failure detection capability. However, if we consider only "Time" failures, the *p-value* would be 0.1487. Therefore, we can observe that, for the considered criteria, significant differences of capability for this kind of failure cannot be observed.

On the other hand, for the density of failure values, by applying the Kruskal-Wallis test we obtained a *p-value* of 0.0670. This means that we cannot reject the null hypotheses: we cannot observe a significant difference on the failure density for the considered criteria. It is also important to mention that no significant correlation between the values of size and failure has been observed for any of the considered criteria.

[8] https://sites.google.com/a/computacao.ufcg.edu.br/rtscoverage/

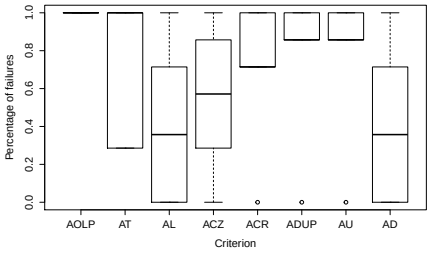

Fig. 5. Boxplot of percentage of failure detected for each criterion.

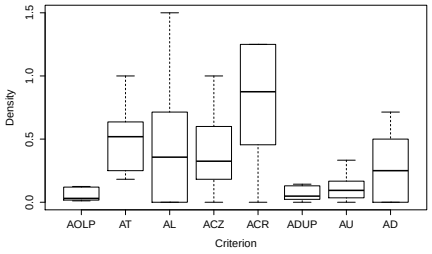

Fig. 6. Boxplot of density of failure detected for each criterion.

General Remarks. From this study, we can observe that the more general criteria such as All-One-Loop-Paths and All-Transitions as well as All-DU-Paths and All-Uses present a better failure coverage even when only time failures are considered. The reason is that more test cases are generated when these criteria are considered. However, they do not always present the best failure density capacity. Which means that a number of test cases may be either useless or redundant for the purpose of detecting the considered failures. From the general criteria, All-Uses (followed by All-DU-Paths) seems to present more consistently the best relation between size and failure detection capability. The reason is that they can most effectively explore the relation between events that are related to a given variable, whereas the structural criteria such as All-Transitions and All-Locations can miss certain combinations. The clock related criteria All-Clock-Zones and All-Clock-Resets present considerably smaller test suites and good density failure capacity, particularly the second one. However, not all failures are covered, even time related ones. Consequently, these criteria may only be considered under severe project constraints. Otherwise, one might consider using both of them together in order to improve failure detection capability and still keep a reasonable failure density.

5 Related Work

Test selection criteria for different kinds of models of real-time systems have already been investigated in the literature. But most of works just describe a criterion or a set of criteria without proper theoretical and empirical evaluation.

En-Nouaary [12] proposes a family of test selection criteria ordered by strict inclusion relation criteria for TIOA models. Our proposal is an extension to his family including data-related criteria for TIOSTS models. We refine the relation between All-Clock-Resets and the class of Transition-Based Coverage criteria. In his family, All-Paths $\Rightarrow$ All-Clock-Resets, but we prove that the narrow relation All-Transitions $\Rightarrow$ All-Clock-Resets is true too. We introduce the relation All-States $\Rightarrow$ All-Locations that was missing. En-Nouaary's family has neither the All-One-Loop-Paths criterion nor the class

of Data-Flow-Oriented Coverage criteria. We introduce them below the All-Paths criterion. Conversely, our family does not have the All-Clock-Regions criterion, because TIOSTS uses zones instead of regions. Finally, only we evaluate empirically the failure detection capability of eight criteria.

Zhu, Hall and May [32] surveys the literature for test selection criteria at source code level. They present several criteria applicable to unit testing, compare them using the strict inclusion relation and provide an axiomatic study of the properties of criteria. Our work is close to theirs because we also compare test selection criteria using the strict inclusion relation. But we work at model level instead of source code level, and we also perform an empirical study to compare selected criteria.

6 Concluding Remarks

In this paper we presented test selection criteria that can be applied to symbolic transition models of real-time systems, particularly, the TIOSTS model.

We investigated the literature for test selection criteria applicable to models of real-time systems. Next we selected the ones applicable to TIOSTS and formalized a family of 19 test selection criteria partially ordered by the strict inclusion relation.

We evaluated 8 criteria in an empirical study with six TIOSTS models. Our results showed that, even though there are differences on the criteria related to size and failure detection capability, the differences were not significant, particularly when considering time-related failures and cost-effectiveness measured as the rate of size by the number of failures.

In general, we can observe that current specific available criteria are still imprecise, because a number of failures were missed. General criteria were precise, but test suites were large, with a high percentage of test cases that did not fail. Therefore, we can conclude that more effective criteria for the model-based testing of real-time systems are still needed, particularly for symbolic models such as TIOSTS.

As future works, we plan to extend this study to include more test selection criteria, specially the Control-Flow-Oriented Criteria which exercise data and time guards thoroughly. Based on the analysis of advantages and weakness of the criteria in a new empirical study, we intend to propose more precise and effective criteria for TIOSTS.

Acknowledgements. This work was supported by the National Council for Scientific and Technological Development (CNPq) under grants 475710/2013-4, 484643/2011-8, and 560014/2010-4. This work was partially supported by the National Institute of Science and Technology for Software Engineering[9] of CNPq under grant 573964/2008-4. First author was also supported by CNPq. Finally, we thank the anonymous reviewers for their constructive comments.

[9] www.ines.org.br

References

1. Alagar, V.S., Ormandjieva, O., Zheng, M.: Specification-based testing for real-time reactive systems. In: Proceedings of the 34th International Conference on Technology of Object-Oriented Languages and Systems. pp. 25–36 (2000)
2. Almeida, D.R.: Critérios de Geração de Casos de Teste de Sistemas de Tempo Real. Master's thesis, Federal University of Campina Grande, Campina Grande, PB, Brazil (2012)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
4. Andrade, W.L., Machado, P.D.L.: Testing interruptions in reactive systems. Formal Aspects of Computing 24, 331–353 (2012)
5. Andrade, W.L., Machado, P.D.L.: Generating test cases for real-time systems based on symbolic models. IEEE Transactions on Software Engineering 39(9), 1216–1229 (2013)
6. Andrade, W.L., Machado, P.D.L., Jéron, T., Marchand, H.: Abstracting time and data for conformance testing of real-time systems. In: Proceedings of the 8th Workshop on Advances in Model Based Testing. pp. 9–17 (2011)
7. Arcuri, A., Iqbal, M.Z., Briand, L.: Black-box system testing of real-time embedded systems using random and search-based testing. In: Proceedings of the 22nd International Conference on Testing Software and Systems. pp. 95–110 (2010)
8. Bengtsson, J., Griffioen, W.O.D., Kristoffersen, K.J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Verification of an audio protocol with bus collision using UPPAAL. In: Proceedings of the 8th International Conference on Computer Aided Verification. pp. 244–256 (1996)
9. Clarke, D., Lee, I.: Automatic test generation for the analysis of a real-time system: Case study. In: Proceedings of the 3rd IEEE Real-Time Technology and Applications Symposium. pp. 112–124 (1997)
10. El-Far, I.K., Whittaker, J.A.: Model-based software testing. In: Marciniak, J.J. (ed.) Encyclopedia of Software Engineering, vol. 1, pp. 825–837. John Wiley & Sons, Inc. (2002)
11. En-Nouaary, A., Khendek, F., Dssouli, R.: Fault coverage in testing real-time systems. In: Proceedings of the 6th Real-Time Computing Systems and Applications. pp. 150–157 (1999)
12. En-Nouaary, A.: Test selection criteria for real-time systems modeled as timed input-output automata. International Journal of Web Information Systems 3(4), 279–292 (2007)
13. En-Nouaary, A., Hamou-Lhadj, A.: A boundary checking technique for testing real-time systems modeled as timed input output automata. In: Proceedings of the 8th International Conference on Quality Software. pp. 209–215 (2008)
14. Hessel, A.: Model-Based Test Case Selection and Generation for Real-Time Systems. Ph.D. thesis, Uppsala University, Uppsala, Sweden (2007)
15. Jeannet, B., Jéron, T., Rusu, V., Zinovieva, E.: Symbolic test selection based on approximate analysis. In: Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 349–364 (2005)
16. Krichen, M., Tripakis, S.: Black-box conformance testing for real-time systems. In: Proceedings of the 11th International SPIN Workshop on Model Checking of Software. pp. 109–126 (2004)

17. Krichen, M., Tripakis, S.: Conformance testing for real-time systems. Formal Methods in System Design 34(3), 238–304 (2009)
18. Laplante, P.A.: Real-Time System Design and Analysis. John Wiley & Sons (2004)
19. Locke, C.D., Vogel, D.R., Lucas, L., Goodenough, J.B.: Generic avionics software specification. Tech. rep., Software Engineering Institute, Carnegie Mellon University (1990)
20. Nielsen, B., Skou, A.: Test generation for time critical systems: Tool and case study. In: Proceedings of the 13th Euromicro Conference on Real-Time Systems. pp. 155–162 (2001)
21. Peleska, J.: Industrial-strength model-based testing - state of the art and current challenges. In: Proceedings of the 8th Workshop on Model-Based Testing. pp. 3–28 (2013)
22. Pretschner, A., Slotosch, O., Aiglstorfer, E., Kriebel, S.: Model-based testing for real. International Journal on Software Tools for Technology Transfer 5(2), 140–157 (2004)
23. Rapps, S., Weyuker, E.J.: Selecting software test data using data flow information. IEEE Transactions on Software Engineering 11(4), 367–375 (1985)
24. Rusu, V., du Bousquet, L., Jéron, T.: An approach to symbolic test generation. In: Proceedings of the 2nd International Conference on Integrated Formal Methods. pp. 338–357 (2000)
25. Sommerville, I.: Software Engineering. International Computer Science Series, Addison-Wesley, Boston, MA, USA, 9 edn. (2010)
26. Trab, M.S.A., Alrouh, B., Counsell, S., Hierons, R.M., Ghinea, G.: A multi-criteria decision making framework for real time model-based testing. In: Proceedings of the 5th International Academic and Industrial Conference on Testing - Practice and Research Techniques. pp. 194–197 (2010)
27. Tretmans, J.: Model-based testing and some steps towards test-based modelling. In: Proceedings of 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems. pp. 297–326 (2011)
28. Tretmans, J.: Testing concurrent systems: A formal approach. In: Proceedings of the the 10th International Conference on Concurrency Theory. pp. 46–65 (1999)
29. Utting, M., Legeard, B.: Practical Model Based Testing: A Tools Approach. Elsevier, San Francisco, CA, USA (2007)
30. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, New York, NY, USA (2012)
31. Zheng, M., Alagar, V., Ormandjieva, O.: Automated generation of test suites from formal specifications of real-time reactive systems. Journal of Systems and Software 81(2), 286–304 (2008)
32. Zhu, H., Hall, P.A.V., May, J.H.R.: Software unit test coverage and adequacy. ACM Computing Surveys 29(4), 366–427 (1997)