Exhaustive Model-Based Equivalence Class Testing

Wen-ling Huang and Jan Peleska*

Department of Mathematics and Computer Science
University of Bremen, Germany
{huang,jp}@informatik.uni-bremen.de
http://informatik.uni-bremen.de/agbs

Abstract. In this article we present a formal justification of model-based equivalence partition testing applied to black box tests of reactive systems with large input data types (floating point types or large integer ranges) and small internal and output data ranges. Systems of this variant typically perform control tasks, where a small number of control commands is issued, depending on analogue or discretised input data (e.g., sensors) and internal control states. We prove that a finite collection of input traces whose elements have been selected from a specific set of input equivalence classes suffices to prove a conformance relation between specification model and system under test. This proof holds under certain practically feasible fault hypotheses. The proof is performed on systems whose operational semantics may be encoded by means of Kripke Structures. It is shown how the semantics of SysML state machines can be represented in Kripke Structures, so that the theorem induces an equivalence class testing strategy for this formalism in a straightforward way. To our best knowledge, this is the first formal justification of the well-known equivalence class testing principle for systems with potentially infinite input data types.

Keywords: Model-based testing, Equivalence class partition testing, Kripke Structures, UML/SysML state machines.

1 Introduction

Motivation. Equivalence class testing is a well-known heuristic approach to testing software or systems whose state spaces, inputs and/or outputs have value ranges of a cardinality inhibiting exhaustive enumeration of all possible values within a test suite. The heuristic suggests to create *equivalence class partitions* structuring the input or output domain into disjoint subsets for which *"the behavior of a component or system is assumed to be the same, based on the specification"* [11, p. 228]. If this assumption is justified it suffices to test "just a few" values from each class, instead of exploring the behavior of the system under

* The authors' research is funded by the EU FP7 COMPASS project under grant agreement no.287829.

H. Yenigün, C. Yilmaz, and A. Ulrich (Eds.): ICTSS 2013, LNCS 8254, pp. 49–64, 2013.
© IFIP International Federation for Information Processing 2013

test (SUT) for each possible value. In order to investigate that the SUT respects the boundaries between different equivalence class partitions *boundary values* are selected for each class, so that equivalence class and boundary value testing are typically applied in combination. As an alternative to deriving equivalence class partitions from the specification, the structure of the SUT or its model can be analyzed: classes are then defined as sets of data leading to the same execution paths [13, B.19].

For testing safety-critical systems the justification of the equivalence class partitions selected is a major challenge. It has to be reasoned why the behaviour of the SUT can really be expected to be equivalent for all values of a class, and why the number of representatives selected from each class for the test suite is adequate. While being quite explicit about the code coverage to be achieved when testing safety-critical systems, standards like [13,10,7] do not provide any well-defined acceptance conditions for equivalence class partitions to be sufficient.

Main Contributions. In this paper we present rules for generating input equivalence class partitions, whose justification is given by the fact that they lead to an *exhaustive* test suite: under certain hypotheses the generated classes and the test data selected from them *prove* conformance between a specification model and its implementation, if the latter passes all tests of this suite. The algorithm is applicable in a model-based testing context, provided that the behavioural semantics of the modelling formalism can be expressed using Kripke Structures. The equivalence class partitioning strategy is elaborated and proven to be exhaustive on Kripke Structures. As an example of a concrete formalism, we illustrate how the strategy applies to SysML state machine models [8]. To our best knowledge, this is the first formal justification of the well-known equivalence class testing principle for systems with potentially infinite input data types (see Section 6 for a discussion of related work).

Example 1. The following example describes a typical system of the class covered by our input equivalence class partition testing strategy. It will be used throughout the paper for illustrating the different concepts and results described in this paper. The example is taken – in simplified form, in order to comply with the space limitations of this publication – from the specification of the European Train Control System ETCS and describes the required behaviour of the *ceiling speed monitoring* which protects trains from overspeeding, as specified in [15, 3.13.10.3]. The interface is shown in Figure 1. The I/O variables have the following meaning.

Interface	**Description**
V_{est}	Current speed estimation [km/h]
V_{MRSP}	Applicable speed restriction [km/h] (MRSP = Most Restrictive Speed Profile)
W	Warning to train engine driver at driver machine interface (DMI) (1 = displayed, 0 = not displayed)
EB	Emergency brake (1 = active, 0 = inactive)

Fig. 1. Interface of the ETCS ceiling speed monitoring function (simplified)

The behaviour of the ceiling speed monitoring function is specified by the UML (or SysML) state machine shown in Figure 2. The function gives a warning to the train engine driver if the currently applicable speed limit V_{MRSP} is not observed, but the actual estimated speed V_{est} does not exceed the limit too far. If the upper threshold for the warning status is violated (this limit is specified by guard conditions g_{ebi_1} or g_{ebi_2}), the emergency brake is activated. After such an *emergency brake intervention* has occurred, the brakes are only released after the train has come to a standstill. While the specification model requires guard condition $g_{ebi_1} \vee g_{ebi_2}$, we assume for the purpose of this example that the implementation has an erroneous guard implementation $\overline{g}_{ebi_1} \vee g_{ebi_2}$. □

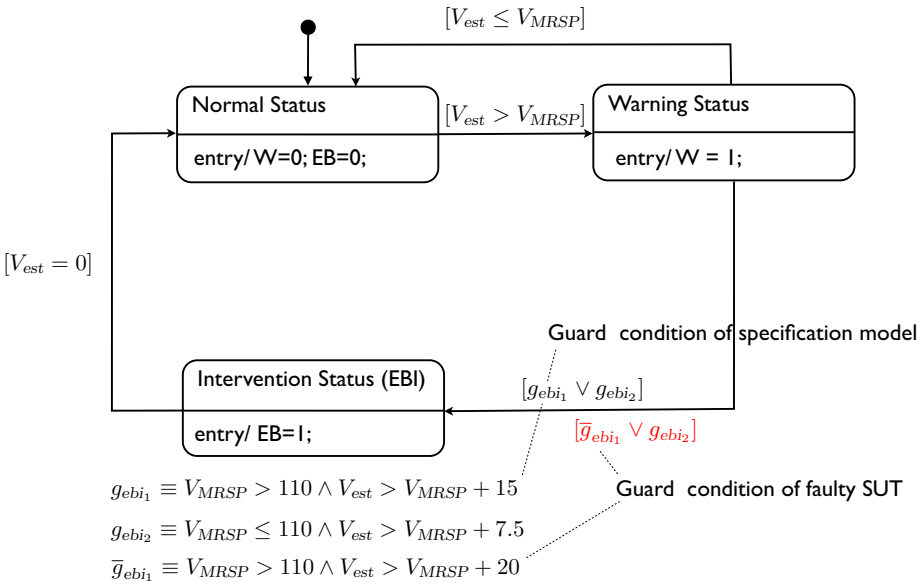

Fig. 2. State machine of the ETCS ceiling speed monitoring function

Overview. In Section 2 the basic concepts about *Reactive Kripke Structures* – a specialisation of general Kripke Structures which is suitable for application in a reactive systems context – are introduced. In Section 3 it is shown how input equivalence class partitionings for Reactive Kripke Structures are constructed. In Section 4, two test hypotheses are presented, whose validity allows us to prove that our equivalence class partitioning and test data selection principle leads to an exhaustive test suite (Theorem 1). While this theorem states that I/O equivalence can be established using a finite input alphabet only (though the input data types may be infinite), it does not state whether the number of input traces needed is finite. In Section 5 we therefore show by means of this theorem, that Reactive Kripke Structures associated with input equivalence partitionings can be abstracted to deterministic finite state machines. Then the well-known W-Method can be applied to establish a *finite* exhaustive test suite proving I/O equivalence between specification model and SUT. Section 6 discusses related work, and we conclude with a discussion of the results obtained and a conjecture about an extension of the main theorem's validity in Section 7.

2 Reactive Kripke Structures

2.1 Notation and Definitions

Let $K = (S, S_0, R, L, AP)$ be a Kripke Structure (KS) with state space S, initial states $S_0 \subseteq S$, transition relation $R \subseteq S \times S$ and labelling function $L : S \to \mathbb{P}(AP)$, where AP is a set of atomic propositions. We specialise on state spaces over variable valuations: let V be a set of variable symbols for variables $v \in V$ with values in some domain $D = \bigcup_{v \in V} D_v$. The state space S of K is the set of all variable valuations $s : V \to D$ which are properly typed in the sense that $s(v) \in D_v$ for all $v \in V$.[1] It is required throughout the paper that the labelling function shall be consistent with, and determined by these variable valuations, in the sense that AP contains propositions with free variable in V and[2]

$$\forall s \in S : L(s) = \{p \in AP \mid s(p)\}$$

Since s satisfies exactly the propositions contained in $L(s)$, it satisfies the negation of all propositions in the complement, that is, $\forall p \in AP - L(s) : \neg s(p)$.

K is called a *Reactive Kripke Structure (RKS)* if it satisfies the following additional properties.

[1] The state space is always total in the sense that all $s : V \to D$ are elements of S. This allows us to assume that specification models K and implementations K' operate on the same state space S, possibly with differing subsets of *reachable* states. The types D_v are always assumed to be maximal, like $\mathbb{R}$, `float` or `int` with a given bit width. Therefore we can assume that a faulty SUT may produce erroneous variable values $s'(v)$, but will never violate the type D_v of its interface variables v.

[2] We use notation $s(p)$ for the Boolean expression p, where every free variable $v \in var(p)$ has been replaced by its current value $s(v)$ in state s. For example, $s(x < y)$ is true if and only if $s(x) < s(y)$ holds. Observe that this can be alternatively written as $s \models p$, or $p[s(v)/v \mid v \in V]$.

1. V can be partitioned into disjoint sets $V = I \cup M \cup O$ called input variables, (internal) model variables, and output variables, respectively.[3]
2. The state space can be partitioned into states from where only input changing transitions are possible, and those from where only internal and output changing transitions are possible. The former states are called *quiescent*, the latter *transient*.

$$\begin{aligned}\exists S_Q, S_T \subseteq S : S = S_Q \cup S_T \wedge S_Q \cap S_T = \varnothing \wedge \\ \forall (s, s') \in R : s \in S_Q \Rightarrow s'|_{M \cup O} = s|_{M \cup O} \wedge \\ \forall (s, s') \in R : s \in S_T \Rightarrow s'|_I = s|_I\end{aligned}$$

3. All initial states have the same internal and output variable valuations, and all possible inputs are allowed in initial states.[4]

$$\exists \overline{s} : M \cup O \to D : S_0 = \{\{\boldsymbol{x} \mapsto \boldsymbol{c}\} \oplus \overline{s} \mid \boldsymbol{c} \in D_I\}$$

4. The input vector may change without any restrictions.

$$\forall s \in S_Q, s' \in S : s'_{M \cup O} = s|_{M \cup O} \Rightarrow (s, s') \in R$$

5. Transient states are always followed by quiescent ones. Internal and output state changes are deterministic in the sense that they only depend on the current state valuation.

$$\exists T : S_T \to S_Q : \forall s \in S_T, s' \in S : (s, s') \in R \Rightarrow s' = T(s)$$

   Function T can be extended to the complete state space by defining $\forall s \in S_Q : T(s) = s$.
6. The transition relation is total on S.[5]

The rules above imply that the transition relation of an RKS can be written as $R = \{(s, s') \mid s \in S_Q \wedge s'|_{M \cup O} = s|_{M \cup O}\} \cup \{(s, T(s)) \mid s \in S_T\}$. While transient states always have quiescent ones as post-states (this is stated in rule 5), quiescent states may have both transient and quiescent ones as post-states.

Example 2. Consider the UML/SysML state machine described in Example 1. Its behavioural semantics can be described by an RKS $K = (S, S_0, R, L, AP)$ with variable symbols from $V = I \cup M \cup O$, $I = \{V_{est}, V_{MRSP}\}$, $M = \{\ell\}$, and $O = \{\mathsf{W}, \mathsf{EB}\}$. Sets I and O contain the interface variable symbols with domains $D_{V_{est}} = D_{V_{MRSP}} = [0, 350] \subseteq \mathbb{R}$ (maximum speed of ETCS trains under consideration is 350km/h). Symbol ℓ ("location") has values in $D_\ell = \{\mathsf{NS}, \mathsf{WS}, \mathsf{IS}\}$ and its valuation signifies the current control state 'Normal Status', 'Warning Status', or 'Intervention Status', respectively. The output symbols have values

[3] Frequently we use input vectors $\boldsymbol{c}$ to the system, where $\boldsymbol{c}$ is an element of $D_I = D_{x_1} \times \cdots \times D_{x_{|I|}}$, and $x_1, \ldots, x_{|I|}$ are the input variables. Changing the valuation of all input variables of a state s_0 to $\boldsymbol{c} = (c_1, \ldots, c_{|I|})$ is written as $s_1 = s_0 \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}\}$. State s_1 coincides with s_0 for all but the the input variables, and $s_1(x_i) = c_i, i = 1, \ldots, |I|$.

[4] Observe that initial states may be quiescent or transient.

[5] Though not all states in S may be reachable from S_0.

in $D_{\mathsf{W}} = D_{\mathsf{EB}} = \mathbb{B} = \{0, 1\}$. The state space S contains all valuations of these symbols, $S = V \to D$, with $D = [0, 350] \cup D_\ell$. Setting $D_I = [0, 350] \times [0, 350]$, the initial states are elements of $S_0 = \{s_0 \in S \mid \exists (c_0, c_1) \in D_I : s_0 = \{V_{est} \mapsto c_0, V_{MRSP} \mapsto c_1, \ell \mapsto \mathsf{NS}, \mathsf{W} \mapsto 0, \mathsf{EB} \mapsto 0\}\}$. Fixing the variable order to vector $(V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB})$, we will from now on describe states s by their value vector $(s(V_{est}), s(V_{MRSP}), s(\ell), s(\mathsf{W}), s(\mathsf{EB}))$, so that an initial state s_0 is written as $(c_0, c_1, \mathsf{NS}, 0, 0)$. The transition relation R is specified by the predicate (see [3] about how to express transition relations as first order predicates)

$$
\begin{array}{l}
R((V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}), (V'_{est}, V'_{MRSP}, \ell', \mathsf{W}', \mathsf{EB}')) \equiv \\
\quad \bigvee_{i=0}^{7} \varphi_i((V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}), (V'_{est}, V'_{MRSP}, \ell', \mathsf{W}', \mathsf{EB}')) \\
\varphi_0 \equiv (\ell = \mathsf{NS} \wedge V_{est} \leq V_{MRSP} \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = \mathsf{W} \wedge \mathsf{EB}' = \mathsf{EB}) \\
\varphi_1 \equiv (\ell = \mathsf{NS} \wedge V_{est} > V_{MRSP} \wedge \ell' = \mathsf{WS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = \mathsf{EB} \wedge V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP}) \\
\varphi_2 \equiv (\ell = \mathsf{NS} \wedge (g_{ebi_1} \vee g_{ebi_2}) \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = 1 \wedge V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP}) \\
\varphi_3 \equiv (\ell = \mathsf{WS} \wedge V_{est} > V_{MRSP} \wedge \neg(g_{ebi_1} \vee g_{ebi_2}) \wedge \ell' = \mathsf{WS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = \mathsf{EB}) \\
\varphi_4 \equiv (\ell = \mathsf{WS} \wedge V_{est} \leq V_{MRSP} \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = 0 \wedge \mathsf{EB}' = 0 \wedge V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP}) \\
\varphi_5 \equiv (\ell = \mathsf{WS} \wedge (g_{ebi_1} \vee g_{ebi_2}) \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = \mathsf{W} \wedge \mathsf{EB}' = 1 \wedge V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP}) \\
\varphi_6 \equiv (\ell = \mathsf{IS} \wedge V_{est} > 0 \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = \mathsf{W} \wedge \mathsf{EB}' = 1) \\
\varphi_7 \equiv (\ell = \mathsf{IS} \wedge V_{est} = 0 \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = 0 \wedge \mathsf{EB}' = 0 \wedge V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP})
\end{array}
$$

The quiescent states are characterised by the pre-conditions ("unprimed conjuncts") in $\varphi_0, \varphi_3, \varphi_6$, the transient states by the pre-conditions in $\varphi_1, \varphi_2, \varphi_4, \varphi_5, \varphi_7$. Observe that in order to enforce the RKS rule 5 (transient states are followed by quiescent states), φ_2 specifies the direct transitions from control state NS to IS. Initial state $s_0 = (0, 90, \mathsf{NS}, 0, 0)$, for example, is quiescent; this follows from φ_0. In contrast to this, $s_1 = (95, 90, \mathsf{NS}, 0, 0) \in S_0$ is transient (φ_1 applies). The latter initial state applies in a situation where the ceiling speed monitoring controller is re-booted while the train is driving ($V_{est} = 95$), and the state is immediately left, since V_{est} exceeds the allowed speed V_{MRSP}. The atomic propositions AP and the labelling function L will be discussed in the examples below.

□

2.2 Quiescent Reduction

The notion of transient states in RKSs is semantically redundant. They only help to facilitate the mapping of concrete modelling formalisms (such finite state machines or UML/SysML state machines) into RKSs. The redundancy of transient states is captured in the following definition.

Definition 1. *Given a Reactive Kripke Structure $K = (S, S_0, R, L)$ the Kripke structure $Q(K)$ defined by*

$$
\begin{array}{l}
Q(K) = (Q(S), Q(S_0), Q(R), Q(L)), \quad Q(S) = S_Q \\
Q(L) = L|_{S_Q} : S_Q \to \mathbb{P}(AP), \quad Q(S_0) = \{T(s_0) \mid s_0 \in S_0\} \\
Q(R) = \{(s, s') \mid s, s' \in S_Q \wedge (R(s, s') \vee (\exists s'' \in S_T : R(s, s'') \wedge s' = T(s'')))\}
\end{array}
$$

is called the quiescent reduction of K. □

The state space of $Q(K)$ consists of quiescent K-states only, and its labelling function is the restriction of L to quiescent states. The initial states of $Q(K)$

consist of the union of the quiescent initial K-states and the quiescent post-states of transient initial K-states (recall that T maps quiescent states to themselves and transient states to their quiescent post-states). The transition relation $Q(R)$ relates quiescent states already related in K, and those pairs of quiescent states that are related indirectly in K by means of an intermediate transient state.

Example 3. For the RKS described in Example 2, the quiescent reduction $Q(K)$ has initial states $Q(S_0) = \{(V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}) \mid V_{est} \leq V_{MRSP} \wedge \ell = \mathsf{NS} \wedge \mathsf{W} = 0 \wedge \mathsf{EB} = 0\} \cup \{(V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}) \mid V_{est} > V_{MRSP} \wedge \neg(g_{ebi_1} \vee g_{ebi_2}) \wedge \ell = \mathsf{WS} \wedge \mathsf{W} = 1 \wedge \mathsf{EB} = 0\} \cup \{(V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}) \mid (g_{ebi_1} \vee g_{ebi_2}) \wedge \ell = \mathsf{IS} \wedge \mathsf{W} = 1 \wedge \mathsf{EB} = 1\}$. The transition relation is given by

$$
\begin{aligned}
&Q(R)((V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}), (V'_{est}, V'_{MRSP}, \ell', \mathsf{W}', \mathsf{EB}')) \equiv \\
&\qquad \textstyle\bigvee_{i=0}^{7} \psi_i((V_{est}, V_{MRSP}, \ell, \mathsf{W}, \mathsf{EB}), (V'_{est}, V'_{MRSP}, \ell', \mathsf{W}', \mathsf{EB}')) \\
&\psi_0 \equiv (\ell = \mathsf{NS} \wedge V'_{est} \leq V'_{MRSP} \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = 0 \wedge \mathsf{EB}' = 0) \\
&\psi_1 \equiv (\ell = \mathsf{NS} \wedge V'_{est} > V'_{MRSP} \wedge \neg(g'_{ebi_1} \vee g'_{ebi_2}) \wedge \ell' = \mathsf{WS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = \mathsf{EB}) \\
&\psi_2 \equiv (\ell = \mathsf{NS} \wedge (g'_{ebi_1} \vee g'_{ebi_2}) \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = 1) \\
&\psi_3 \equiv (\ell = \mathsf{WS} \wedge V'_{est} \leq V'_{MRSP} \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = 0 \wedge \mathsf{EB}' = 0) \\
&\psi_4 \equiv (\ell = \mathsf{WS} \wedge V'_{est} > V'_{MRSP} \wedge \neg(g'_{ebi_1} \vee g'_{ebi_2}) \wedge \ell' = \mathsf{WS} \wedge \mathsf{W}' = 1 \wedge \mathsf{EB}' = \mathsf{EB}) \\
&\psi_5 \equiv (\ell = \mathsf{WS} \wedge (g'_{ebi_1} \vee g'_{ebi_2}) \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = \mathsf{W} \wedge \mathsf{EB}' = 1) \\
&\psi_6 \equiv (\ell = \mathsf{IS} \wedge V'_{est} > 0 \wedge \ell' = \mathsf{IS} \wedge \mathsf{W}' = \mathsf{W} \wedge \mathsf{EB}' = 1) \\
&\psi_7 \equiv (\ell = \mathsf{IS} \wedge V'_{est} = 0 \wedge \ell' = \mathsf{NS} \wedge \mathsf{W}' = 0 \wedge \mathsf{EB}' = 0)
\end{aligned}
$$

□

2.3 Traces

Traces of K are finite sequences of states related by R, including the empty sequence ε, $\mathrm{Traces}(K) = \{\varepsilon\} \cup \{s_0 \ldots s_n \in S^* \mid n \in \mathbb{N}_0 \wedge s_0 \in S_0 \wedge \bigwedge_{i=0}^{n-1} R(s_i, s_{i+1})\}$. The last state of a finite sequence of states is denoted by $last(s_0 \ldots s_n) = s_n$, and $tail(s_0 \ldots s_n) = (s_1 \ldots s_n)$, $tail(s_0) = \varepsilon$. Given trace $s_0 \ldots s_n$ we define its *restriction* to symbols from $X \subseteq V$ by $(s_0 \ldots s_n)|_X = (s_0|_X) \ldots (s_n|_X)$.

Given an RKS $K = (S, S_0, R, L)$, we consider input traces on states in K's quiescent reduction $Q(K)$: an *input trace* $\iota = \boldsymbol{c}_0.\boldsymbol{c}_1 \ldots$ is a finite sequence of input vectors $\boldsymbol{c}_i \in D_I$, that is, $\iota \in (D_I)^*$. The application of an input trace ι to a quiescent state $s \in Q(S)$ is written as s/ι and yields a trace of $Q(K)$ which is recursively defined by $s/\varepsilon = s,\ s/(\boldsymbol{c}_0.\iota) = s.(T(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_0\})/\iota)$.

As in the definitions above, T denotes the function mapping quiescent K-states to themselves and transient ones to their quiescent post-states. Obviously each pair of consecutive states in trace s/ι is related by transition relation $Q(R)$. We will be frequently interested in the last element of an input trace application to a state; therefore the abbreviation $s//\iota = last(s/\iota)$ is used. Since RKSs are deterministic with respect to their reactions on input changes, $s//\iota$ is uniquely determined.

2.4 I/O Equivalence

Model-based testing always investigates some notion of I/O conformance: stimulating the SUT with an input trace ι, the observable behaviour should be consistent with the behaviour expected for ι according to the model. The following definitions specify aspects of I/O equivalence, as they are relevant in the context of Reactive Kripke Structures.

Definition 2. *Given the quiescent reduction $Q(K)$ of some RKS K, and quiescent states $s_0, s_1 \in Q(S)$.*

1. *States s_0 and s_1 are called* I/O equivalent*, written as $s_0 \sim s_1$, if and only if $(s_0/\iota)|_O = (s_1/\iota)|_O$ holds for all input traces ι.*
2. *States s_0 and s_1 are called* ι-equivalent*, written as $s_0 \overset{\iota}{\sim} s_1$, if and only if ι is an input trace satisfying $(s_0/\iota)|_O = (s_1/\iota)|_O$.*

□

Definition 3. *Two RKSs K, K' over the same variable symbols V are called* I/O equivalent *(written $K \sim K'$) if their quiescent reductions are equivalent in the sense that $\forall (s_0, s_0') \in Q(S_0) \times Q(S_0') : (s_0|_I = s_0'|_I \Rightarrow s_0 \sim s_0')$.* □

3 Input Equivalence Class Partitionings over Reactive Kripke Structures with Finite Outputs

In the remainder of this paper we study the special case where our specification models K and implementations K' may have infinite input domains, but have output and internal variables with finite domains only. The term "finite" is to be interpreted here in the sense that these values can be enumerated with reasonable effort. This contrasts with the domains of input variables, which we allow to have infinite range (such as real values) or to have "very large" finite cardinality (such as floating point or large integer types), where an enumeration would be impossible, due to time and memory restrictions. As a consequence, it is possible to further restrict the sets AP of atomic propositions under consideration. Since all possible values of internal states and output variables can be explicitly enumerated, AP can be structured into disjoint sets

$$\begin{aligned} AP &= AP_I \cup AP_M \cup AP_O \\ AP_I &\subseteq \{p \mid p \text{ is atomic and } var(p) \subseteq I\} \\ AP_M &= \{m = \alpha \mid m \in M \wedge \alpha \in D_m\} \\ AP_O &= \{y = \beta \mid y \in O \wedge \beta \in D_y\} \end{aligned}$$

Example 4. Consider a SysML state machine transition

$$\boxed{C_0} \overset{[x<m+y]}{\longrightarrow} \boxed{C_1}$$

with $x \in I, m \in M, y \in O$, where $D_x = \mathbb{R}, D_y = \{0, 1\}, D_m = \{10, 11\}$. When transforming this machine into an RKS, the atomic propositions AP can be strictly separated according to their free variables being from I, M, or O, respectively. For example, $AP = \{\ell = C_0, m = 10, y = 0, x < 10, x < 11, x < 12\}$.

□

Definition 4. *Given RKS $K = (S, S_0, R, L, AP)$ with finite outputs and internal states, and AP partitioned into $AP = AP_I \cup AP_M \cup AP_O$ as described above. If and only if*

$$\forall s_0, s_1 \in S : (L(s_0) = L(s_1) \Rightarrow L(T(s_0)) = L(T(s_1)))$$

then AP is called an input equivalence class partitioning (IECP) *of K, and its* input classes *are specified by*

$$\mathcal{I} = \big\{\{\ \boldsymbol{c} \in D_I \mid \bigwedge_{p \in M} p[\boldsymbol{c}/\boldsymbol{x}] \wedge \bigwedge_{p \in AP_I - M} \neg p[\boldsymbol{c}/\boldsymbol{x}]\ \} \mid M \subseteq AP_I\big\} \setminus \{\varnothing\}$$

□

In order to understand the essence of Definition 4, consider an IECP AP and alternative input changes $\{\boldsymbol{x} \mapsto \boldsymbol{c}\}$ and $\{\boldsymbol{x} \mapsto \boldsymbol{d}\}$ applied to some state s. The input vectors $\boldsymbol{c}, \boldsymbol{d}$ belong to the same input class from $\mathcal{I}$, if and only if they satisfy the same input-related propositions, that is, if

$$\{p \in AP_I \mid p[\boldsymbol{c}/\boldsymbol{x}]\} = \{p \in AP_I \mid p[\boldsymbol{d}/\boldsymbol{x}]\}$$

As a consequence, s, when changed by either $\boldsymbol{c}$ or $\boldsymbol{d}$, will satisfy the same propositions from AP; this means

$$L(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}\}) = L(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}\})$$

Now, since AP is an IECP, Definition 4 implies that

$$L(T(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}\})) = L(T(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}\}))$$

Therefore the post states $T(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}\})$ and $T(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}\})$ will be in the same internal state and produce the same outputs. Recall that T maps quiescent states to themselves, so the IECP property is only non-trivial for the transient states of an RKS.

Example 5. For the ceiling speed monitoring function, whose RKS K has been constructed in Example 2, atomic propositions

$$AP = \{V_{est} = 0, V_{est} > V_{MRSP}, V_{MRSP} > 110, V_{est} > V_{MRSP} + 7.5, V_{est} > V_{MRSP} + 15, \\ \ell = \mathsf{NS}, \ell = \mathsf{WS}, \ell = \mathsf{IS}, \mathsf{W}, \mathsf{EB}\}$$

introduce an IECP for K. Consider, for example, the states s_0 labelled by $L(s_0) = \{V_{est} > V_{MRSP}, \ell = \mathsf{NS}\}$. Each of these s_0 is transient and has a post state s_1 satisfying $V'_{est} = V_{est} \wedge V'_{MRSP} = V_{MRSP} \wedge \ell' = \mathsf{WS} \wedge \mathsf{W}$. As a consequence, all of these post-states are labelled by $L(s_1) = \{V_{est} > V_{MRSP}, \ell = \mathsf{WS}, \mathsf{W}\}$.

□

The following Lemma shows that input traces applied to the same state and passing through the same sequences of input equivalence classes produce identical outputs.

Lemma 1. *Given RKS $K = (S, S_0, R, L, AP)$ with finite outputs and internal state as described above, so that AP is an IECP for K with input classes $\mathcal{I}$. Let $\iota = \boldsymbol{c}_1 \ldots \boldsymbol{c}_k$, $\tau = \boldsymbol{d}_1 \ldots \boldsymbol{d}_k$, $\boldsymbol{c}_i, \boldsymbol{d}_i \in D_I, i = 1, \ldots, k$, such that*

$$\forall i = 1, \ldots, k, \exists X_i \in \mathcal{I} : \{\boldsymbol{c}_i, \boldsymbol{d}_i\} \subseteq X_i$$

Then $\forall s \in S_Q : (s/\iota)|_{M \cup O} = (s/\tau)|_{M \cup O}$.

Proof. Let $X_i \in \mathcal{I}$ satisfying $\{\boldsymbol{c}_i, \boldsymbol{d}_i\} \subseteq X_i, \forall i = 1, \ldots, k$. Let $s \in S_Q$. Denote $s/\iota = s_0.s_1 \ldots s_k$, $s/\tau = r_0.r_1 \ldots r_k$, where $s = s_0 = r_0$. We prove by induction over $i = 0, \ldots, k$ that $s_i|_{M \cup O} = r_i|_{M \cup O}$. For $i = 0$ it is trivial, since $s = s_0 = r_0$. Suppose that the induction hypothesis holds for $i < k$, $s_i|_{M \cup O} = r_i|_{M \cup O}$. Since $s_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_{i+1}\}|_{M \cup O} = r_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}_{i+1}\}|_{M \cup O}$ and, according to the assumptions of the lemma, $\{\boldsymbol{c}_{i+1}, \boldsymbol{d}_{i+1}\} \subseteq X_{i+1}$, we conclude that $L(s_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_{i+1}\}) = L(r_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}_{i+1}\})$.The IECP property of AP now implies that also $L(T(s_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_{i+1}\})) = L(T(r_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}_{i+1}\}))$, and by definition $T(s_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_{i+1}\}) = s_{i+1}, T(r_i \oplus \{\boldsymbol{x} \mapsto \boldsymbol{d}_{i+1}\}) = r_{i+1}$, therefore $s_{i+1}|_{M \cup O} = r_{i+1}|_{M \cup O}$. This proves the lemma. □

Lemma 2. *Given RKS K with finite outputs and internal state as described above, and AP an IECP. Let $p_1, \ldots, p_n$ be a set of fresh atomic propositions not contained in AP, with $var(p_i) \subseteq I, i = 1, \ldots, n$. Then $AP_2 = AP \cup \{p_1, \ldots, p_n\}$ is another IECP, called the* refinement of AP. *The input classes of AP_2, constructed according to Definition 4, are denoted by $\mathcal{I}_2$.* □

Observe that IECP refinement according to Lemma 2 introduces new propositions in AP_I only, while AP_M and AP_O remain unchanged.

4 Test Hypotheses and Proof of Exhaustiveness

The input equivalence class testing strategy to be introduced in this section yield exhaustive tests, provided that the following two test hypotheses are met.

(TH1) Testability Hypothesis. There exists an RKS $K' = (S, S'_0, R', L', AP')$ with finite internal states and output as introduced in Section 3 describing the true behaviour of the SUT, and its state space S consists of valuation functions $s : V \to D$ for variables from V as specified for the reference model $K = (S, S_0, R, L, AP)$.

(TH2) Existence of Refined Equivalence Class Partitioning. For specification model $K = (S, S_0, R, L, AP)$ and SUT $K' = (S, S'_0, R', L', AP')$, both atomic proposition sets AP, AP' are IECP of K and K' with input classes $\mathcal{I}, \mathcal{I}'$, respectively, and $AP_M = AP'_M$, $AP_O = AP'_O$. Moreover, there exists an input partition refinement $AP_2 = AP_{2I} \cup AP_M \cup AP_O$, in the sense of Lemma 2, such that

$$\forall X \in \mathcal{I}, X' \in \mathcal{I}' : \exists X_2 \in \mathcal{I}_2 : (X \cap X' \neq \varnothing \Rightarrow X_2 \subseteq X \cap X')$$

Validity of (TH2) induces a finite input alphabet to K and K' which will be shown below to suffice for uncovering any violation of I/O equivalence between K and K'.

Definition 5. *Given RKSs K, K' with finite internal state and outputs, and input equivalence class partitionings AP, AP' and AP_2 according to test hypothesis (TH2). Let $\mathcal{A}_I$ denote a finite subset of input vectors $\boldsymbol{c} \in D_I$ satisfying $\forall X \in \mathcal{I}_2 : \exists \boldsymbol{c} \in \mathcal{A}_I : \boldsymbol{c} \in X$. Then $\mathcal{A}_I$ is called an* input alphabet *for equivalence class partition testing of K' against K. For any nonnegative integer k, ${\mathcal{A}_I}^k$ is the set of all $\mathcal{A}_I$-sequences of length less than or equal to k (including the empty trace ε).* □

Example 6. Let K be the RKS of the ceiling speed monitor model constructed in Example 2, with IECP AP as given in Example 5. Now suppose that the SUT implementing the monitor model has an error, as indicated in Figure 2: it uses a faulty guard condition $\overline{g}_{ebi_1} \vee g_{ebi_1}$ instead of $g_{ebi_1} \vee g_{ebi_1}$. Its IECP (which, of course, would be unknown in a black box test) is $AP' = \{V_{est} = 0, V_{est} > V_{MRSP}, V_{MRSP} > 110, V_{est} > V_{MRSP} + 7.5, V_{est} > V_{MRSP} + 20, \ell = \mathsf{NS}, \ell = \mathsf{WS}, \ell = \mathsf{IS}, \mathsf{W}, \mathsf{EB}\}$. The IECP refinement of AP, $AP_2 = \{V_{est} = 0, V_{est} > V_{MRSP}, V_{MRSP} > 110, V_{est} > V_{MRSP} + 7.5, V_{est} > V_{MRSP} + 15, V_{est} > V_{MRSP} + 18.75, V_{est} > V_{MRSP} + 22.5, \ell = \mathsf{NS}, \ell = \mathsf{WS}, \ell = \mathsf{IS}, \mathsf{W}, \mathsf{EB}\}$ fulfils test hypothesis (TH2). Consider, for example the intersection of K input class $X = \{(V_{est}, V_{MRSP}) \mid V_{MRSP} > 110 \wedge V_{est} > V_{MRSP} + 15\}$ and the K' input class $X' = \{(V_{est}, V_{MRSP}) \mid V_{MRSP} > 110 \wedge V_{est} > V_{MRSP} + 7.5 \wedge \neg(V_{est} > V_{MRSP} + 20)\}$. Then the input class $X_2 = \{(V_{est}, V_{MRSP}) \mid V_{MRSP} > 110 \wedge V_{est} > V_{MRSP} + 15 \wedge \neg(V_{est} > V_{MRSP} + 18.75)\}$ of the refined IECP AP_2 is contained in the intersection $X \cap X'$. Indeed, any input from X_2 applied to the SUT in control state WS would reveal the erroneous guard condition, because K transits into IS, while K' remains in WS.

For practical application (since the IECP of the SUT is unknown), the input space D_I is systematically partitioned by intersecting the input-related propositions from AP with interval vectors, partitioning D_I into $|I|$-dimensional cubes. □

Theorem 1. *Given RKSs $K = (S, S_0, R, L, AP)$, $K' = (S, S'_0, R', L', AP')$, such that AP, AP' are IECP of K and K' with input classes $\mathcal{I}, \mathcal{I}'$, respectively, and AP_2 is a refinement of AP according to test hypothesis (TH2). $\mathcal{I}_2$ contains the input classes associated with AP_2. Let $\mathcal{A}_I$ be an input alphabet derived from $\mathcal{I}_2$ according to Definition 5. Then for any quiescent states $s \in S_Q$, $s' \in S'_Q$ and any input trace ι, there exists an input trace $\tau \in {\mathcal{A}_I}^*$ with the same length, such that $s/\iota|_O = s/\tau|_O$ and $s'/\iota|_O = s'/\tau|_O$. Hence, $s \overset{\iota}{\sim} s'$ if and only if $s \overset{\tau}{\sim} s'$.*

Proof. If ι is empty, there is nothing to prove, since $\varepsilon \in \mathcal{A}_I$. Suppose therefore, that $\iota = \boldsymbol{c}_1 \ldots \boldsymbol{c}_k$ with $k \geq 1$ and let $s/\iota = s_0.s_1 \ldots s_k$, and $s'/\iota = s'_0.s'_1 \ldots s'_k$, where $s_0 = s, s'_0 = s'$.

Consider the associated sequences of input classes $X_1 \ldots X_k \in \mathcal{I}$ and $X'_1 \ldots X'_k \in \mathcal{I}'$, where $\boldsymbol{c}_i \in X_i$ and $\boldsymbol{c}_i \in X'_i$, for all $i = 1, \ldots, k$. Since $\boldsymbol{c}_i \in X_i \cap X'_i \neq \varnothing$, $i = 1, \ldots, k$, (TH2) implies the existence of $X_{21}, \ldots, X_{2k} \in \mathcal{I}_2$ such that

$$X_{2i} \subseteq X_i \cap X'_i, \quad i = 1, \ldots, k \quad (*)$$

According to Definition 5, we can select $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k \in \mathcal{A}_I$, such that $\boldsymbol{d}_i \in X_{2i}$ for all $i = 1, \ldots, k$. (*) implies $\boldsymbol{d}_i \in X_i \cap X'_i \quad i = 1, \ldots, k$. Therefore, setting $\tau = \boldsymbol{d}_1 \ldots \boldsymbol{d}_k$, Lemma 1 may be applied to conclude that $(s/\iota)|_O = (s/\tau)|_O$ and $(s'/\iota)|_O = (s'/\tau)|_O$. Therefore $s \overset{\iota}{\sim} s' \Leftrightarrow s \overset{\tau}{\sim} s'$, and this completes the proof. □

5 Test Strategy

5.1 Application of the W-Method

Given specification model $K = (S, S_0, R, L, AP)$ and SUT $K' = (S, S'_0, R', L', AP')$, and the refined IECP AP_2 with input classes $\mathcal{I}_2$ according to test hypothesis (TH2). Let $\mathcal{A}_I$ be the input alphabet constructed from $\mathcal{I}_2$ as specified in Definition 5. Then AP, $\mathcal{A}_I$ and each $s_0 \in Q(S_0)$ induce a deterministic finite state machine (DFSM) abstraction $M(K, s_0) = (\mathcal{Q}, q_0, \mathcal{A}_I, D_O, \delta, \omega)$ of K with state space $\mathcal{Q} = \{[s] \mid s \in S_Q\}$, initial state $q_0 = [s_0]$, and input alphabet $\mathcal{A}_I$, where $[s] = \{r \in S_Q \mid r \sim s\}$. Let O be the set of output variables of K. The output alphabet of $M(K, s_0)$ is defined by $D_O = D_{y_1} \times \ldots \times D_{y_{|O|}}$. The state transition function $\delta : \mathcal{Q} \times \mathcal{A}_I \to \mathcal{Q}$ of $M(K, s_0)$ is defined by

$$\delta(q, \boldsymbol{c}) = q_1 \text{ if and only if } \exists s \in S_Q : q = [s] \wedge q_1 = [s//\boldsymbol{c}]$$

The output function $\omega : \mathcal{Q} \times \mathcal{A}_I \to D_O$ of $M(K, s_0)$ is defined by

$$\omega(q, \boldsymbol{c}) = \boldsymbol{e} \text{ if and only if } \exists s \in S_Q : q = [s] \wedge (s//\boldsymbol{c})|_O = \{\boldsymbol{y} \mapsto \boldsymbol{e}\}$$

We extend the domain of the state transition function to input traces, $\overline{\delta} : \mathcal{Q} \times \mathcal{A}_I^* \to \mathcal{Q}^*$ by setting recursively $\overline{\delta}(q, \varepsilon) = q$, $\overline{\delta}(q, \boldsymbol{c}.\iota) = q.\overline{\delta}(\delta(q, \boldsymbol{c}), \iota)$. The output function can be extended to $\overline{\omega} : \mathcal{Q} \times \mathcal{A}_I^* \to D_O^*$ by setting $\overline{\omega}(q, \iota) = \boldsymbol{e}_0 \ldots \boldsymbol{e}_k$, if and only if $\overline{\delta}(q, \iota) = [s_0] \ldots [s_k]$ and $s_i|_O = \{\boldsymbol{y} \mapsto \boldsymbol{e}_i\}, i = 0, \ldots, k$.

Lemma 3. *The DFSMs $M(K, s_0) = (\mathcal{Q}, q_0, \mathcal{A}_I, D_O, \delta, \omega)$ introduced above are well-defined.*

Proof. Let $q = [s]$ and $[r] = [s]$ for some $s, r \in S_Q$. Then $r \sim s$, and therefore $s//\boldsymbol{c} \sim r//\boldsymbol{c}$, and this shows that $\delta(q, \boldsymbol{c})$ is well-defined. Since all members of $[s//\boldsymbol{c}]$ coincide on O, this also shows that ω is well-defined. □

By construction, the DFSMs are minimal, because each pair of different states $[s_0] \neq [s_1]$ can be distinguished by an input trace resulting in different outputs when applied to $[s_0]$ or $[s_1]$, respectively. Since AP is an IECP, all K-states s_0, s_1 carrying the same label $L(s_0) = L(s_1)$ are I/O-equivalent, so $\{s_1 \mid L(s_1) = L(s)\} \subseteq [s]$ for all quiescent states of K. It may be the case, however, that some states carrying different labels are still I/O-equivalent, that is, $L(s_0) \neq L(s_1)$, but $\{s \mid L(s) = L(s_0)\} \cup \{s \mid L(s) = L(s_1)\} \subseteq [s_0] = [s_1]$. In analogy to $M(K, s_0)$, DFSMs $M(K', s'_0)$ can be constructed from $K', AP', s_0 \in Q(S'_0)$, and the same input alphabet $\mathcal{A}_I$ as has been used for the DFSMs $M(K, s_0)$.

We write $M(K, s_0) \sim M(K', s'_0)$ and $q_0 \sim q'_0$, if and only if $\overline{\omega}(q_0, \iota) = \overline{\omega}'(q'_0, \iota)$ for every $\iota \in \mathcal{A}_I^*$. Note that this differs from I/O equivalence between K and K', where $s_0 \sim s'_0$ if and only if $(s_0/\iota)|_O = (s'_0/\iota)_O$ for every $\iota \in D_I^*$. The following theorem states that I/O equivalence between specification model K and an implementation K' can be established by investigating the equivalence of their associated DFSM, that is, using $\iota \in \mathcal{A}_I^*$ only.

Theorem 2. *With the notation above, the following statements are equivalent.*

- K *and* K' *are I/O equivalent,* $K \sim K'$.
- $\forall s_0 \in Q(S_0), s_0' \in Q(S_0') : (s_0|_I = s_0'|_I \Rightarrow M(K, s_0) \sim M(K', s_0'))$.

Proof. Obviously, $M(K, s_0) \sim M(K', s_0') \Leftrightarrow q_0 \sim q_0' \Leftrightarrow (\forall \tau \in \mathcal{A}_I^* : s_0 \overset{\tau}{\sim} s_0')$. By Theorem 1, we have $(\forall \iota \in D_I^* : s_0 \overset{\iota}{\sim} s_0') \Leftrightarrow (\forall \tau \in \mathcal{A}_I^* : s_0 \overset{\tau}{\sim} s_0')$. Hence $s_0 \sim s_0' \Leftrightarrow M(K, s_0) \sim M(K', s_0')$. Now the assertion follows directly from the definition of $K \sim K'$ (Definition 3). □

Definition 6. *With the terms introduced above, a* transition cover *of* $M(K, s_0)$ *is a set of input traces* $\iota \in \mathcal{A}_I^*$ *satisfying the following condition: for any reachable state* $q \in \mathcal{Q}$ *and any* $\boldsymbol{c} \in \mathcal{A}_I$*, there is an input trace* $\iota \in TC$ *such that* $\overline{\delta}(q_0, \iota) = q$ *and* $\iota.\boldsymbol{c} \in TC$. □

Definition 7. *With the terms introduced above and minimal* $M(K, s_0)$*, define a* characterisation set W *of* $M(K, s_0)$ *as a set of traces* $\iota \in \mathcal{A}_I^*$*, such that for all* $q_1, q_2 \in \mathcal{Q}$*, there exists an input trace* $\iota \in W$ *such that* $\overline{\omega}(q_1, \iota) \neq \overline{\omega}(q_2, \iota)$. □

On DFSM $M(K, s_0)$ we can apply Chow's W-method [2] to conclude that the following finite test suite is exhaustive for testing I/O equivalence between K and K'.

Theorem 3. *Let* $s_0 \in Q(S_0), s_0' \in Q(S_0')$ *with* $s_0|_I = s_0'|_I$*, and* $TC(s_0), W(s_0)$ *the transition cover and characterisation set of* $M(K, s_0)$ *as introduced above. Assume that* $M(K, s_0)$ *has* n *states and that* $M(K', s_0')$ *has at most* m *states and* $m_0 = \max(n, m)$*. Then*

$$\mathcal{W}(K) = \bigcup_{[s_0] \in Q(S_0)/\sim} \left(TC(s_0).\mathcal{A}_I^{m_0 - n}.W(s_0) \right)$$

is an exhaustive test suite for testing SUT K' *against specification model* K.

Proof. $M(K, s_0)$ and $M(K', s_0')$ are two minimal DFSMs with the same input alphabet $\mathcal{A}_I$. Applying Chow's W-method [2] to $M(K, s_0)$ and $M(K', s_0')$, $M(K, s_0)$ and $M(K', s_0')$ are I/O equivalent if and only if they are $TC(s_0).\mathcal{A}_I^{m_0 - n}.W(s_0)$ equivalent.[6] Hence the assertion follows directly from Theorem 2. □

For the example of the ceiling speed monitor introduced in this paper, a detailed description of the test cases resulting from application of Theorem 3 can be found in [6, Section 7.5].

[6] Observe that in [2], the author uses a slightly different notation, where $\mathcal{A}_I^i$ denotes the set of input traces with length i, while we use this term to denote the traces of length less or equal i.

5.2 Complexity Considerations

Definition 5 determines the size of the input alphabet $\mathcal{A}_I$ as the number $k_2 = |\mathcal{A}_I| \leq 2^{|AP_{2I}|}$ of input classes in the refined equivalence partitioning AP_2 according to test hypothesis (TH2).

The number n of states in the DFSM associated with K is less or equal to the number $\overline{n}$ of labels $L(s), s \in S_Q$ (we get $n < \overline{n}$, if different labels $L(s_0) \neq L(s_1)$ are associated with I/O equivalent states). Let $m_0 = \max(n, m)$, where n is the number of states in $M(K, s_0)$, and m the number of states in $M(K', s'_0)$. Then according to [2,16], the number of input traces contained in $TC(s_0).\mathcal{A}_I^{m_0-n}.W(s_0)$ is bounded by $n^2 \cdot k_2^{m_0-n+1}$. We have to execute several test suites of this type, their number is equal to $k = |Q(S_0)/\sim|$, the number of equivalence classes derived from initial states of the quiescent reduction of K. In the worst case, all classes of K can be reached from some transient initial state, so $k \leq n$. This results in an upper bound of $k \cdot n^2 \cdot k_2^{m_0-n+1} \leq n^3 \cdot k_2^{m_0-n+1}$ test cases (that is, input traces) to be performed.

5.3 Summary of SUT-Related Estimates

While parameters n, k are calculated from the known representation of K, the following hypotheses about the SUT influence the complexity parameters m, m_0, k_2 introduced above. (1) The size k_2 of the input alphabet relies on (TH2) (Section 4); we assume that $\mathcal{A}_I$ is sufficiently fine-grained, such that one $\boldsymbol{c} \in \mathcal{A}_I$ can be found in every intersection of input classes X, X' associated with K and K', respectively. (2) The number m of $M(K', s'_0)$-states is bounded by $\overline{m} = |\{L'(s) \mid s \in S \text{ and } s \text{ reachable in } K'\}|$. The finite number of reachable internal states and output states is bounded by $\prod_{v \in M \cup O} |D_v|$, the product of finite value ranges for internal state variables and output variables. The number of different proposition sets $M' \subset AP'_I$ fulfilled by reachable states of K' is bounded by the number k_2 of elements in $\mathcal{A}_I$, because this set contains one element per input equivalence class of AP_2 refining AP'. As a consequence, $m \leq \overline{m} \leq k_2 \cdot \left(\prod_{v \in M \cup O} |D_v|\right)$. (3) This also determines $m_0 = \max(n, m)$.

6 Related Work

Notable examples for exhaustive test methods have been given in [2,12,9,14]. There exists a large variety of research results related to testing against hierarchic state machines similar to Harel's Statecharts or to UML state machines. We mention [4] as one representative and refer to the references given there. These contributions, however, mainly deal with the state machine hierarchy and do not tackle the problem of attributes from large input domains, which is the main motivation for the results presented here. In [1, pp. 205] large data domains in the context of state machine testing are addressed, but no formal justification of the heuristics presented there are given.

In model-based testing, the idea to use data abstraction for the purpose of equivalence class definition has been originally introduced in [5], where the classes

are denoted as *hyperstates*, and the concept is applied to testing against abstract state machine models. Our results presented here surpass the findings described in [5] in the following ways: (1) while the authors of [5] introduce the equivalence class partitioning technique for abstract state machines only, our approach extracts partitions from the models' semantic representation. Therefore an exhaustive equivalence class testing strategy can be elaborated for any formalism whose semantics can be expressed by Kripke Structures. (2) The authors sketch for white box tests only how an exhaustive test suite could be created [5, Section 4]: the transition cover approach discussed there is only applicable for SUT where the internal state (respectively, its abstraction) can be monitored during test execution. (3) The authors only consider finite input sets whose values have been fixed *a priori* [5, Section 2], whereas our approach allows for inputs from arbitrary domains.

Our notion of I/O equivalence (Definition 2) corresponds to the well-known **ioco** relation, when translating the Reactive Kripke Structures into input/output transition systems (IOTS) as defined in [14]. To this end, however, the requirement [14, Definition 1] that LTL should only have countably many states and labels has to be dropped, since RKS deal with potentially uncountable input data types. IOTS traces restricted to input actions correspond to our input traces, and IOTS suspension traces to our traces revealing inputs and outputs. Our test strategy is based on quiescent reduction, that is, only outputs in quiescent states are visible. The resulting suspension traces are therefore of the form $\sigma = \boldsymbol{c}_1.\boldsymbol{y}_1.\delta.\boldsymbol{c}_2.\boldsymbol{y}_2.\delta\ldots$, where $\boldsymbol{c}_i$ are input actions, $\boldsymbol{y}_i$ are outputs, and δ is the special output action denoting quiescence. This restricted type of suspension traces occurs naturally in test applications where test data is exchanged between test environment and SUT via shared variables, and not via events.

7 Conclusion and Future Work

In this paper, a novel exhaustive test strategy for input equivalence class testing has been established. The main result (Theorem 1) shows that even in presence of infinite input data domains, a finite input alphabet can be identified, so that for every trace performed by specification model or implementation, there exists a trace using inputs from this finite alphabet only, but producing the same outputs as the original one. This result holds for arbitrary modelling formalisms, whose semantics may be expressed by Reactive Kripke Structures with input domains that may be infinite (or too large to be explicitly enumerated), but with internal states and outputs having a sufficiently small range to be enumerated in an explicit way. With the main theorem at hand, the well-known W-Method can be applied to identify a finite and at the same time exhaustive test suite. Using an abstraction of the Kripke Structures under consideration to deterministic finite state machines, we have proven that this method is applicable.

Further research will focus on the generalisation of the test strategy to Reactive Kripke Structures with arbitrary data domains for internal states and outputs. According to our conjecture, a result similar to Theorem 1 should hold

in the general case. The equivalence classes under consideration, however, will no longer refer to system inputs only, but will be characterised by more general atomic propositions with inputs, internal state and outputs as free variables.

References

1. Binder, R.V.: Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley (2000)
2. Chow, T.S.: Testing software design modeled by finite-state machines. IEEE Transactions on Software Engineering SE-4(3), 178–186 (1978)
3. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)
4. Gnesi, S., Latella, D., Massink, M.: Formal test-case generation for uml statecharts. In: Ninth IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2004), pp. 75–84. ICECCS (2004)
5. Grieskamp, W., Gurevich, Y., Schulte, W., Veanes, M.: Generating finite state machines from abstract state machines. ACM SIGSOFT Software Engineering Notes 27(4), 112–122 (2002)
6. Huang, W., Peleska, J., Schulze, U.: Comprehensive modelling for advanced systems of systems – specialised test strategies. Public Document D34.2, COMPASS (October 2013), `http://www.compass-research.eu/deliverables.html`
7. ISO/DIS 26262-4: Road vehicles – functional safety – part 4: Product development: system level. Tech. rep., International Organization for Standardization (2009)
8. Object Management Group: OMG Systems Modeling Language (OMG SysML^{TM}). Tech. rep., Object Management Group (2010), OMG Document Number: formal/2010-06-02
9. Peleska, J., Siegel, M.: Test automation of safety-critical reactive systems. South African Computer Jounal 19, 53–77 (1997)
10. RTCA,SC-167: Software Considerations in Airborne Systems and Equipment Certification, RTCA/DO-178B. RTCA (1992)
11. Spillner, A., Linz, T., Schaefer, H.: Software Testing Foundations. Dpunkt Verlag, Heidelberg (2006)
12. Springintveld, J., Vaandrager, F., D'Argenio, P.: Testing timed automata. Theoretical Computer Science 254(1-2), 225–257 (2001)
13. European Committee for Electrotechnical Standardization: EN 50128 – Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems. CENELEC, Brussels (2001)
14. Tretmans, J.: Model based testing with labelled transition systems. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST. LNCS, vol. 4949, pp. 1–38. Springer, Heidelberg (2008)
15. UNISIG: ERTMS/ETCS SystemRequirements Specification, ch. 3, Principles, vol. Subset-026-3 (2012), issue 3.3.0
16. Vasilevskii, M.P.: Failure diagnosis of automata. Kibernetika (Transl.) 4, 98–108 (1973)