A Coverage Metric For The Verification Of Discrete-Time Dynamic Systems

Pawel Skruch, AGH University of Science and Technology, Department of Automatics

Abstract
The paper defines a measure that can be used to evaluate tests for software systems described by discrete-time dynamic models. The measure is based on a state space model format of the system under tests. The model is mathematically formulated by difference equations. The proposed solution gives the test engineer assurance that a given test set is sufficient and indicates where additional testing is needed. The classical coverage metrics, such as statement, branch, and path coverage, are not appropriate for the dynamic systems because such systems have usually infinitely many states. An example is presented to illustrate theoretical analysis and mathematical formulation.

1. Introduction
In recent years, the extensive use of electronics has revolutionized the implementation and scope of many control systems. Control algorithms in the form of mathematical and logical formulas are first modelled and then simulated using sophisticated computational tools. They are verified, calibrated and optimized against system performance requirements and expectations. Once the model of a control algorithm is ready it can be either manually transformed into the code or auto-generation can be used to get code automatically generated from the model. In the next step, the code is downloaded to the hardware. Testing process allows verifying that the software behaviour in the final hardware is identical to that observed during computer simulations (see Fig. 1). As the complexity of control systems grows, testing becomes more crucial and time consuming [11].

Dynamic systems can be found in all major engineering disciplines and include mechanical, electrical, fluid and thermal systems. Dynamic behaviour is also implemented in a digital machine to realize specific functions. It should be noted that testing of dynamic aspects of software systems (both continuous and discrete) creates a number of challenges and still is not sufficiently supported by tools and methods. One of these challenges is to create a set of tests that adequately exercises the behaviour of the system under test (SUT). The adequacy of such set is inferred by examining different coverage metrics on the SUT. Currently, there is no objective standard for directly determining the adequacy of a black-box test set for dynamic systems. The classical coverage metrics such as statement, branch, and path coverage are unsuitable for the behaviour of the dynamic system, because such system has usually infinitely many states.

![Fig.1. Flowchart for the analysis and design of dynamic systems.](image)

The behaviour of the software systems can be categorized into three different types [3], [5]: discrete combinational (logical operation defined) behaviour, discrete sequential (state defined) behaviour, and continuous behaviour. The combinational behaviour is typically modelled using Boolean algebra [2], [9], the state behaviour is modelled using graphs [2], [3], [6], and the continuous behaviour is modelled using...
The objective of this paper is to study the system (1), (2) under the condition that the discrete state spaces $U$, $X$ and $Y$ are bounded sets of $\mathbb{R}^r$, $\mathbb{R}^m$, $\mathbb{R}^n$, respectively. In practice, many software systems have multiple resource constraints such as energy, memory, and implementation constraints. These constraints can be somehow reflected by the above assumption.

3. System Testing With A Model As An Oracle

Testing dynamic aspects of software systems requires test cases that utilize time continuous input signals and time continuous output signals (even when the system is digitally processed). The execution of a test case consists of exciting the system using actuators to simulate its working conditions, and measuring the system’s response in terms of electrical signals, motion, force, strain, etc. The signals can be both physical and virtual. In the concept of model-based testing (see [13], [15]), the mathematical model of the SUT can be used as a mathematical description describing the system behaviour, $\mathbb{R}^r \times \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ determines the output, $U$ is the input space, $X$ is the state space, $Y$ is the output space, $\mathbb{R}^r$, $\mathbb{R}^m$, $\mathbb{R}^n$ are real vector spaces of column vectors, $r$, $m$, $n$ are positive integers.
predefined tolerance range relative to the expected output \( y \) (then the test is qualified as passed), and \( z(y(k), y_s(k)) = 1 \) in other cases (then the test is qualified as failed).

![Mathematical Model](image)

**Fig. 3. Model-based testing concept.**

A test case is a set of input values, execution preconditions, expected results and execution postconditions, developed to verify compliance with a specific requirement. Using the state space modelling concept of the SUT (Fig. 2), a single test case can be defined as:

\[
T_{\text{case}} : (K, x_0, u, y),
\]

where \( K \) is a final step, \( x_0 \) is an initial condition, \( u \) is an input sequence applied to the system, \( y \) is a sequence expected on the system output.

### 4. Test Coverage For The Output Space

Test coverage is the degree to which a specified coverage item has been exercised by a test set. Given a mathematical model of the SUT, the most obvious quantification of the model output space coverage exercised by a test set is computed by dividing the number of the output states explored by the test set by the cardinality of the entire output space. However, this method has limited usefulness since the output space for dynamic systems has usually the infinite number of states.

The idea presented in this paper is to transform the output space \( Y \) into another space that contains countable number of elements. Introducing the definition of a hyperelement \( G_h \) in the space \( \mathbb{R}^m \) plays an important role in this transformation.

**Definition 1 (Hyperelement).** A hyperelement \( G_h (i_1, i_2, \ldots, i_n) \) with the edge length \( h > 0 \) in the space \( \mathbb{R}^m \) is a set of points defined as:

\[
G_h (i_1, i_2, \ldots, i_m) = \left\{ y \in \mathbb{R}^m : \frac{y_j}{h} = i_j, j = 1, 2, \ldots, m \right\},
\]

where \( i_j \in \mathbb{Z}, j = 1, 2, \ldots, m \), \( \frac{y_j}{h} \) is the largest integer not greater than \( \frac{y_j}{h} \), \( \mathbb{Z} \) stands for the set of integers.

**Definition 2 (Transformed output space).** The transformed output space \( Y_h \) with the edge \( h > 0 \) is defined as:

\[
Y_h = \left\{ y \in \mathbb{Z}^m : \exists x \in Y \ y \in G_h (i_1, i_2, \ldots, i_m) \right\},
\]

The number of elements in the transformed space \( Y_h \) depends on the parameter \( h > 0 \) and can be chosen accordingly based on the size of the system that is being tested and the resources that are available in test environment.

**Definition 3 (Output space coverage).** Let \( T_{\text{set}} = \left\{ T_{\text{case}}^{(1)}, T_{\text{case}}^{(2)}, \ldots, T_{\text{case}}^{(N)} \right\} \) be a test set, where:

\[
T_{\text{case}}^{(j)} : (K^{(j)}, x_0^{(j)}, u^{(j)}, y^{(j)}),
\]

\( j = 1, 2, \ldots, N \). Let \( V_h \left( T_{\text{case}}^{(j)} \right) \) be a set of states in the transformed space \( Y_h \) covered by the test case \( T_{\text{case}}^{(j)} \):

\[
V_h \left( T_{\text{case}}^{(j)} \right) = \left\{ y \in Y_h : \exists \ y \in G_h (i_1, i_2, \ldots, i_m) \right\},
\]

The output space coverage \( C_h \) of the test set \( T_{\text{set}} \) is defined as:

\[
C_h (T_{\text{set}}) = \frac{\left| \bigcup_{j=1}^{N} V_h \left( T_{\text{case}}^{(j)} \right) \right|}{\left| Y_h \right|}.
\]

### 5. Example

Consider a discrete-time dynamic system defined by the equations (1), (2) with two-dimensional output space \( Y \). The state space \( Y \) is induced by the output variables \( y(k) = [y_1(k) \ y_2(k)]^T \) (see Fig. 4).

![Illustration of the output space and the transformed space](image)
Let $T_{\text{set}} = \{ T_{\text{case}}^{(1)}, T_{\text{case}}^{(2)} \}$ be a test set that consists of two test cases. Fig. 4 shows how these test cases are reflected in the output space $Y$. The figure illustrates also the transformed output space $Y_h$. Elements of this space are depicted by squares with dotted edges.

According to Fig. 4, the elements of the space $Y_h$ covered by the test cases $T_{\text{case}}^{(1)}$ and $T_{\text{case}}^{(2)}$ can be expressed in the following form:

$$V_h(T_{\text{case}}^{(1)}) = \{ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}, \begin{array}{c} 2 \\ 2 \\ 2 \\ 2 \\ 3 \\ 3 \\ 2 \end{array} \} \quad (9)$$

$$V_h(T_{\text{case}}^{(2)}) = \{ \begin{array}{c} 2 \\ 3 \\ 3 \\ 4 \\ 4 \\ 4 \\ 3 \end{array}, \begin{array}{c} 4 \\ 2 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \end{array} \} \quad (10)$$

The coverage for the set $T_{\text{set}}$ is equal:

$$C_h(T_{\text{set}}) = \frac{V_h(T_{\text{case}}^{(1)}) \cup V_h(T_{\text{case}}^{(2)})}{|Y_h|} = \frac{17}{38}. \quad (11)$$

6. Conclusions

In the paper, the test coverage metric has been defined. The metric is a measure that can help the test engineer in testing of discrete-time dynamic systems, because traditional measures used in software testing are unsuitable. The proposed solution is easy in implementation, platform and language independent and can handle systems with growing size in a graceful manner. Using this metric, various test strategies and test generation algorithms can be developed.

Acknowledgements

This work was supported by Ministry of Science and Higher Education in Poland in the years 2008-2011 as a research project No N N514 414034.

Bibliography


Sam K. authors...

Paweł Skruch, Ph.D.
Department of Automation
AGH University of Science and Technology
Al. Mickiewicza 30/B1
30-059 Kraków
tel. (012) 617 28 34
tax (012) 634 15 68

e-mail: pawel.skruch@agh.edu.pl