Model Based Regression Test Reduction Using Dependence Analysis

Bogdan Korel
Computer Science Department
Illinois Institute of Technology
Chicago, IL 60616
korel@iit.edu

Luay H. Tahat
Lucent Technologies
Bell Labs Innovations
Naperville, IL 60566
ltahat@lucent.com

Boris Vaysburg
Computer Science Department
Illinois Institute of Technology
Chicago, IL 60616
vaysbor@iit.edu

Abstract

Model based testing is a system testing technique used to test software systems modeled by formal description languages, e.g., an Extended Finite State Machine (EFSM). System models are frequently changed because of specification changes. Selective test generation techniques are used to test the modified parts of the model. However, the size of regression test suites still may be very large. In this paper, we present a model-based regression testing approach that uses EFSM model dependence analysis to reduce regression test suites. The approach automatically identifies the difference between the original model and the modified model as a set of elementary model modifications. For each elementary modification, regression test reduction strategies are used to reduce the regression test suite based on EFSM dependence analysis. Our initial experience shows that the approach may significantly reduce the size of regression test suites.

1. Introduction

During software maintenance of large and evolving software systems, their specification and implementation are changed to fix defects, to enhance or change functionality, to add new functionality, or to delete the existing functionality. Regression testing is the process of validating that the changes introduced in a system are correct and do not adversely affect the unchanged portion of the system and to provide confidence that changes are correct. Regression testing tends to consume a large amount of time and computing resources, especially for large software systems. To minimize time and effort spent on testing and re-testing a software product, many companies invest into tools to automate the regression testing process. During regression testing, previously developed test cases are deployed for revalidating a modified system, but also new test cases are frequently generated. There has been a significant amount of research on the design of effective regression testing techniques to reduce the cost of regression testing [18].

There exist two types of regression testing: code-based and specification-based regression testing. It has been shown that code-based testing and specification-based testing complement each other [4]. Most regression testing techniques are code-based, i.e., these techniques select test cases using the source code of the original and modified programs [1, 11, 12, 13, 15, 17, 18, 20, 25]. There exists limited research on specification-based regression testing techniques [2, 7, 16, 22]. Most of these techniques select regression tests using only the modified system specification.

Model based testing is one of the techniques that can be used on the system level. In recent years, several model-based test generation techniques [3, 4, 5, 8, 19, 21, 23, 24] have been developed. These techniques are appropriate for state-based systems that can be modeled using formal description languages like Extended Finite State Machine (EFSM), Specification Description Language (SDL) [9], or ESTELLE [14]. These techniques are used to automatically generate system-level test suites even for large software systems. Several tools have been developed to support these model-based automated test generation techniques [6, 19]. These tools automatically generate test suites from system models.

System models are frequently modified to reflect changes in specifications. When the system model is changed, selective regression test generation techniques [21] are used to generate regression test suites to test the modified parts of the model. The size of these regression test suites may be very large even for relatively small systems. Therefore, regression test suite reduction is important, especially, in situations where executing regression tests is very time and resource consuming.

In this paper, we present a novel approach of model-based regression test reduction that uses EFSM model dependence analysis to reduce a given regression test suite. The approach automatically identifies the difference between the original and modified system models by identifying a set of elementary modifications: elementary addition of a transition and elementary deletion of a transition. For each elementary modification, regression test reduction strategies that use EFSM dependence analysis are used to reduce the regression test suite by eliminating repetitive tests. Our initial experience shows that this approach may significantly reduce the size of regression test suites.
The paper is organized as following: Section 2 provides an overview of the model-based testing, Section 3 presents an approach of model-based regression test generation, Section 4 introduces EFSM dependencies, and Section 5 presents an approach of model-based regression test suite reduction based on EFSM dependence analysis. In Conclusions, future research is discussed.

2. Model based testing

Model based testing is one of the techniques that can be used on the system level. Model-based testing [3, 4, 5, 8, 19, 21, 23, 24] is used to test state based systems that are modeled using formal description languages like EFSM, SDL [9], ESTELLE [14], etc. In this paper, we concentrate on the EFSM system models. However, our approach can be extended to models described in other modeling languages, e.g., SDL [21, 23]. EFSM [9, 19] is very popular for modeling state-based systems like computer communications, telecommunications, and industrial control systems. An EFSM consists of states (including an initial state and an exit state) and transitions between states. A transition is triggered when an event occurs and a condition associated with the transition is satisfied. When a transition is triggered, an action(s) may be performed. The action may manipulate variables, read input or produce output. The condition associated with a transition is a Boolean predicate that must evaluate to TRUE in order for the transition to be taken.

EFSM models are graphically represented as graphs where states are represented as nodes and transitions as directed edges between states. The following elements are associated with each transition: (a) an event, (b) a condition, and (c) a sequence of actions. Figure 1 shows graphical representation of an EFSM transition.

![Figure 1. EFSM transition](image1)

A simplified EFSM model of an ATM system is shown in Figure 2. This ATM system supports two types of transactions: withdrawal and deposit represented by transitions. Before ATM transactions can be performed, user must enter a valid PIN that is matched against the PIN stored in the ATM card. A user is allowed a maximum of three attempts to enter the valid PIN. For example, the transition labeled $T_2$ is triggered when the system is in state $S_1$, event PIN is received, the value of parameter $p$ of the event does not equal to variable $pin$, and the value of variable $attempts$ is less than 3. When the transition is triggered, an error message is displayed, the value of variable $attempts$ is incremented, and the user is prompted to enter PIN.

![Figure 2. EFSM model of the ATM system](image2)

An EFSM model of an ATM system becomes an input to an EFSM test generator that may support a variety of the existing EFSM model-based test generation strategies [19, 21, 23, 24]. Depending on the selected testing strategy, the test generator automatically generates a set of tests (paths from an initial state to the final state) in the EFSM model that satisfies the selected strategy. For each path, appropriate test values (inputs) that lead to the traversal of the selected path are identified. Clearly, a test case consists of a sequence of events (transitions) with appropriate input values. The following is an example of a test case for the ATM system shown of Figure 2:

- Card(1234, 100.00); PIN(1234); Deposit(20); Receipt; Withdrawal(50); Receipt; Exit.

In this paper, we concentrate mainly on tests as sequences of transitions (events) and we do not consider input values. Therefore, the test shown above is represented as the following sequence of transitions: $T_1, T_4, T_6, T_7, T_5, T_7, T_8$.

Most of the existing EFSM model-based test generation strategies are mainly used to test the whole system, referred to as complete system testing. Several testing strategies exist, e.g., transition coverage, path coverage, and constrained path coverage. Transition coverage requires that every transition in the model be traversed at least once. Path coverage requires that every path in the model be traversed at least once; this strategy is frequently not practical because of an unacceptable number of test cases generated in the presence of cycles in the model. A modified path strategy, referred to as constrained path coverage, limits the test explosion by limiting a number of times each transition can be traversed. This strategy requires that every path in the model be traversed at least once where each path can contain at most $n$ "occurrences" of the same transition (any transition can be traversed at most $n$ times in a path). This is a frequently used testing.
strategy in model-based testing. For example, consider the
EFSM system model shown in Figure 2. A tester decides
to generate a complete system test suite using a constrained
path coverage testing strategy. The resulting complete
system test suite contains 64 tests for \( n = 3 \); for \( n = 4 \), the
complete system test suite size is 160.

Complete system testing is very expensive because of a
large number of test cases that are generated. In the early
stages of a testing process, a frequently used type of testing
is selective system testing [21, 23, 24] where testers want
to partially test the system with respect to a set of selected
transitions of the model (these transitions frequently
represent individual requirements). Several selective
testing strategies exist, e.g., transition coverage, path
coverage, and constrained path coverage. Transition
coverage requires that every selected transition in the
model be traversed at least once. A selective constrained
path coverage strategy is identical to the system
coverage selective testing (described in the previous
sections) with respect to the added transition, e.g.,
constrained path selective testing.

Testing addition of a transition
The goal is to test whether the added transition performs
as expected. Regression testing of an added transition is
equivalent to selective testing (described in the previous
section) with respect to the added transition, e.g.,
constrained path selective testing.

Testing deletion of a transition
The goal is to test whether the deleted transition does
not cause unintended effects and the system performs
correctly in the situations when the deleted transition was
involved. Since the deleted transition does not exist
anymore, it cannot be traversed in the modified model.
However, assuming that during traversal of a test the
system is in the state from which the deleted transition was
outgoing, it is possible to imitate traversal of the deleted
transition when the event associated with the deleted
transition is generated and the enabling condition of the
deleted transition evaluates to TRUE. This is considered as
a traversal of the deleted transition. As a result, selective
testing techniques, described in the previous section, may
be used for regression testing of a deleted transition
(imitation of a traversal of the deleted transition), e.g.,
constrained path selective testing with respect to the
deleted transition.

The union of regression test suites generated for every
elementary modification constitutes the regression test
suite. Such test suite contains a set of tests such that the
chosen selective testing strategy is satisfied for every
elementary modification. For example, when constrained
path selective testing strategy is chosen to test the model
modification, the resulting regression test suite contains
tests that satisfy the constrained path selective testing
strategy for every elementary modification (for every
added and deleted transition). Notice that one test may
cover (test) multiple elementary modifications. Therefore,
one may try to identify a minimal regression test suite such
that the chosen selective testing strategy is satisfied for
every elementary modification.
This model-based regression testing approach generates high quality test suites. However, a relatively large number of tests may be generated. In the next sections, we present a novel approach that uses EFSM dependence analysis to achieve significant reduction of a model based regression test suite.

4. EFMS dependencies

Before we present the approach of regression test suite reduction using EFSM dependence analysis, we introduce dependencies that may exist in the EFSM model. We define two types of dependencies between transitions (“active” elements of an EFSM model): data dependence and control dependence. Note that states are “passive” elements of the EFSM model. These dependencies capture the notion of potential “interactions” between transitions in the model.

Let T be a transition. The following notation related to transition T is introduced:

- \( S_i(T) \) is a state from which T is incoming.
- \( S_o(T) \) is a state to which T is outgoing.
- \( U(T) \) is a set of variables used in transition T, i.e., variables used in a condition or an action of T.
- \( D(T) \) is a set of variables defined by transition T, i.e., variables defined by an action or defined in an event of T and not redefined by the action of T.
- \( C(T) \) is an enabling condition associated with transition T.
- \( E(T) \) is an event associated with transition T.

For example, in the EFSM model of Figure 2 for transition \( T_2 \), \( U(T_2) = \{p, \text{pin}, \text{attempts}\} \) and \( D(T_2) = \{p, \text{attempts}\} \).

Data dependence

Data dependence captures the notion that one transition defines a value to a variable and another transition may potentially use this value. More formally, there exists data dependence between transitions \( T_i \) and \( T_k \) if there exists a variable \( v \) such that: (1) \( v \in D(T_i) \), (2) \( v \in U(T_k) \), and (3) there exists a path (transition sequence) in the EFSM model from \( T_i \) to \( T_k \) along which \( v \) is not modified; such a path is referred to as definition-clear path. For example, there exists data dependence between transitions \( T_1 \) and \( T_3 \) because transition \( T_1 \) assigns a value to variable \( b \) (in the event \( \text{Card}^{\text{pin}}(\text{b}) \)), transition \( T_3 \) uses variable \( b \) (in action “\( b = b - w \)”), and there exists a path (sequence of transitions \( T_1, T_4, T_3 \)) from \( T_1 \) to \( T_3 \) along which \( b \) is not modified.

Control dependence

Control dependence was originally defined for a program control graph [10]. Control dependence captures the notion that one node in the control graph may affect the execution of another node. In this paper, we extend the concept of control dependence to the EFSM model. Control dependence in the EFSM exists between transitions, and it captures the notion that one transition may affect traversal of another transition. Control dependence between transitions is defined similarly as control dependence between nodes of a program control graph [10], i.e., in terms of the concept of post-dominance. Let \( Y \) and \( Z \) be two states (nodes) and \( T \) be an outgoing transition (edge) from \( Y \). State \( Z \) post-dominates state \( Y \) if \( Z \) is on every path from \( Y \) to the exit state. State \( Z \) post-dominates transition \( T \) if \( Z \) is on every path from \( Y \) to the exit state through transition \( T \).

Transition \( T_i \) has control dependence on transition \( T_k \) (transition \( T_k \) is control dependent on transition \( T_i \)) if (1) state \( S_i(T_k) \) does not post-dominate state \( S_i(T_i) \), and (2) state \( S_i(T_k) \) post-dominates transition \( T_i \). For example, transition \( T_4 \) has control dependence on transition \( T_5 \) in the EFSM model of Figure 2 because state \( S_i(T_5) \) does not post-dominate state \( S_i(T_4) \) and state \( S_i(T_5) \) post-dominates transition \( T_4 \).

EFSM dependence graph

Data and control dependence in the EFSM system can be graphically represented by a graph where nodes represent EFSM transitions and directed edges represent EFSM data and control dependencies. For example, Figure 3 shows static EFSM dependence graph of the EFSM model of Figure 2. Note that data dependencies are shown as solid edges and control dependencies are shown as dashed edges.

Figure 3. EFSM dependence graph

5. Regression test suite reduction

In this paper, we present a model-based regression test suite reduction approach that uses EFSM model dependence analysis to reduce regression test suites. The approach accepts as inputs: the original model, the modified model, and a given regression test suite. The regression test suite may be generated using model-based regression test generation technique described in Section 3 or any other regression test generation technique. The
presented regression test suite reduction approach is independent of the way the regression test suite was originally created. The goal of the approach is to reduce the given regression test suite using EFSM dependence analysis.

The approach automatically identifies the difference between the original model and the modified model as a set of elementary model modifications. For each elementary modification, regression test reduction strategies that use EFSM dependence analysis are used to reduce the test suite. In general, when a software system is modified three types of testing are performed: testing the affect of the system on the modification (modified part of the system), testing the affect of the modification on the remaining part of the system, and testing side effects introduced by the modification. As a result, when a model is modified, three types of model-based regression testing need to be performed: (1) test the affect of the model on the modification (modified part of the model), (2) test the affect of the modification on the remaining part of the system model, and (3) test side affects caused by the modification.

Our approach is based on the observation that not the whole system model affects the modified part of the model and that not the whole system is affected by the modified part of the model. Frequently, only a relatively small part of the model affects the modified part of the model, and only a relatively small part of the model is affected by the modified part of the model.

During test execution, different elements of a system interact with each other, and different tests may exercise different interactions. Therefore, the goal is to test different interactions or patterns of interactions between functional elements of the system with respect to a transition that represents an elementary modification. There is no need to test the system with tests that exercise the same interactions with respect to the transition under test (such tests are repetitive). Two tests are equivalent with respect to the elementary modification if, during traversal of the EFSM model, the tests exhibit the same interactions or the same pattern of interactions with respect to the transition under test. Our goal is to reduce a test suite by identifying equivalent test cases with respect to the transition under test and removing these tests from the selective test suite. For example, a regression test suite for the EFSM model of Figure 2 contains the following two tests, among others:

- **Test #1**: \(T_1, T_4, T_6, T_7, T_6, T_7, T_5, T_7, T_8\)
  - Valid PIN entered, two deposit transactions performed followed by a withdrawal transaction.

- **Test #2**: \(T_1, T_5, T_4, T_6, T_7, T_6, T_7, T_5, T_7, T_8\)
  - Invalid PIN entered, followed by a valid PIN, two deposit transactions performed followed by a withdrawal transaction.

Suppose, a tester is interested in testing transition \(T_3\) because it corresponds to an elementary modification (e.g., addition of a transition). Notice that these two tests are identical except for the number of times PIN was entered. However, entering the PIN (number of times PIN is entered) does not interact with transitions \(T_4\). Therefore, the interaction pattern with respect to transition \(T_3\) must be the same for both tests.

Since there are three types of regression testing, we introduce three types of interaction patterns related to the modification: (1) an affecting interaction pattern, (2) an affected interaction pattern, and (3) a side-effect interaction pattern. The affecting interaction pattern captures interactions between model elements that affect the modification. The affected interaction pattern captures interactions that are affected by the modification. Finally, the side-effect interaction pattern captures interactions that occur because of side effects introduced by the modification.

In this paper, we assume that interactions between EFSM transitions are represented as EFSM dependencies between transitions. In order to identify equivalent tests, we use EFSM dependencies that were defined in the previous section and new dependencies that will be defined in the next subsections. During traversal of a test (traversal of a transition sequence), dependencies (represented as a dependence sub-graph) related to the different types of interaction patterns for the transition under test are identified. Clearly during traversal of a test three interaction patterns (dependence sub-graphs) are computed for each modification: the affecting interaction pattern, the affected interaction pattern, and the side-effect interaction pattern. If the same interaction pattern of a certain type is computed for two different tests for an elementary modification, these tests are considered equivalent, with respect to the elementary modification and the interaction pattern.

In the next subsections, we discuss computation of interaction patterns for two types of elementary model modifications: the addition of a transition and the deletion of a transition.

### 5.1. Testing the addition of a transition

When a transition is added, three types of testing needs to be performed: (1) testing the affect of the model on the added transition, (2) testing the affect of the added transition on the model, and (3) testing the side effects introduced by the added transition.

#### 5.1.1. Testing the affect of the model on the added transition

The goal is to test interactions that affect the added transition. The approach works as follows: During the traversal of a test, data and control dependencies that
occur during the traversal of the test are identified and marked in the EFSM dependence graph. In the next step, all unmarked dependencies are removed from the EFSM dependence graph. Then, all the dependencies, in the dependence sub-graph that "affect" the added transitions are identified by traversing backwards from the added transition and marking all traversed dependencies. All unmarked dependencies are removed from the sub-graph. The resulting dependence sub-graph is referred to as an Affecting Interaction Pattern, where data and control dependencies represent interactions between transitions. For example, consider the model shown in Figure 2. The Balance Inquiry transaction has been added to the ATM system. The transaction is represented by transition T9 that has been added to the ATM system of Figure 4.

![Figure 4. ATM system with added balance transaction](image)

Suppose the regression test suite contains the following two tests:

- **Test #1**: T1, T4, T6, T7, T5, T7, T9, T7, T8
- **Test #2**: T1, T2, T4, T6, T7, T5, T7, T9, T7, T8

Both of these tests result in the same affecting interaction pattern shown in Figure 5, therefore, these two tests are equivalent with respect to the affecting interaction pattern.

![Figure 5. Affecting interaction pattern](image)

5.1.2. Testing the affect of the added transition on the model. The goal is to test the affect of the added transition on the model. During traversal of a test, data and control dependencies that occur during the traversal are identified and marked in the EFSM dependence graph. In the next step, all unmarked dependencies are removed from the EFSM dependence graph. Then, all the dependencies, in the dependence sub-graph that are "affected" by the added transition are identified by traversing forward from the added transition and marking all traversed dependencies. All unmarked dependencies are removed from the sub-graph. The resulting dependence sub-graph is referred to as an Affecting Interaction Pattern, where data and control dependencies represent interactions between transitions. Two tests are equivalent, with respect to the affected interaction pattern, if both tests result in the same interaction pattern. For example, the two tests discussed in Section 5.1.1 result in the same affected interaction pattern shown in Figure 6.

![Figure 6. Affected interaction pattern](image)

5.1.3. Testing side-effects introduced by the added transition. An addition of a transition may introduce new dependencies between the existing parts of the system that do not exist in the original model (note that addition of a new transition cannot cause deletion of existing dependencies between parts of the model). The goal of testing side effects is to test the newly introduced dependencies between the existing parts of the system.

The new dependencies introduced by the addition of a transition may not directly relate to the added transition. For example, adding the balance transaction to the ATM system introduces a new data dependence between transitions T1 and T3 that does not exist in the original model. Therefore, we introduce a new type of dependence between the added transition and the new dependence.

This new dependence is referred to as an activation dependence and is defined as follows: Suppose a new transition Tn has been added to the system model. Due to introduction of transition Tn, there exists a new data dependence between T1 and Tn with respect to variable v in the modified model that does not exist in the original model. More formally: there exists an activation dependence between T1 and the data dependence (T1, Tn) if follows: (1) there exists a definition clear path from T1 to Tn, (2) Tn is triggered, and (3) there exists a definition clear path from T0 to Tn with respect to v. The activation dependence becomes a part of the EFSM dependence graph. Its graphical representation is shown in Figure 7.

During the traversal of a test, data, control and activation dependencies that occur during the traversal are
identified and marked in the EFSM dependence graph. In the next step, all unmarked dependencies are removed from the EFSM dependence graph. Then, all the dependencies, in the dependence sub-graph that are "affected" by the added transition are identified by traversing forward from the added transition through the marked activation dependencies and marking all traversed dependencies. All unmarked dependencies are removed from the sub-graph. The resulting dependence sub-graph is referred to as a Side-Effect Interaction Pattern, where data, control, and activation dependencies represent interactions between transitions. Two tests are equivalent, with respect to the side-effect interaction pattern, if both tests result in the same side-effect interaction pattern. For example, the two tests discussed in Section 5.1.1 result in the same side-effect interaction pattern, shown in Figure 7.

![Figure 7. Side-effect interaction pattern](image)

**5.2. Testing the deletion of a transition**

Consider the ATM model of Figure 4. The Deposit transition is deleted from the original model, resulting in the model shown in Figure 8. For the purpose of this presentation, transition $T_8$ (represented by dashed arrow) is shown in the model of Figure 8, however, it does not appear in the modified model. Notice that transition $T_{10}$ is added to the model to indicate that the Deposit event is handled in state $S_2$ and represents an empty operation. Testing added transition $T_{10}$ will result in a single test because all interaction (affecting, affected, and side effects) patterns for testing $T_{10}$ are empty. This example clearly shows that testing only added transitions is not sufficient in regression testing, and additional tests are needed that test a deleted transition.

When a transition is deleted, three types of testing need to be performed: (1) testing the affect of the model on the deleted transition, (2) testing the affect of the deleted transition on the model, and (3) testing the side effects introduced by the deleted transition.

**5.2.1. Testing the affect of the model on the deleted transition.** The goal of this type of testing is to test interactions that used to affect the deleted transition. Deletion of a transition can cause elimination of dependencies associated with the deleted transition where the deleted transition was dependent on another transition (deletion of a transition cannot introduce new dependencies). For example, the original model of the ATM system of Figure 4 contains data dependence between transitions $T_1$ and $T_6$. This data dependence does not exist in the modified model of Figure 8.

The eliminated dependence is referred to as the affecting ghost dependence. We define ghost dependence as follows: Suppose $T_j$ is a deleted transition, there exists an affecting ghost data dependence between $T_j$ and $T_i$ in the modified model, if: (1) there is a data dependence between $T_j$ and $T_i$ with respect to variable $v$, in the original model, (2) there exists a definition clear path from $T_j$ to $S_a(T_i)$ with respect to $v$, in the modified model, (3) event $E(T_i)$ occurs in state $S_b(T_i)$, and (4) condition $C(T_i)$ evaluates to TRUE. For example, there exists a ghost data dependence between transitions $T_1$ and $T_6$, in the modified model of Figure 8.

During traversal of a test, data, control, and affecting ghost dependencies that occur during the traversal of the test are identified and marked in the EFSM dependence graph. In the next step, all unmarked dependencies are removed from the EFSM dependence graph. Then, all the dependencies, in the dependence sub-graph, that "affect" the deleted transition are identified by traversing backwards from the deleted transition through the affecting ghost dependencies and marking all traversed dependencies. All unmarked dependencies are removed from the sub-graph. The resulting dependence sub-graph is referred to as an Affecting Interaction Pattern, where data, control, and affecting ghost dependencies represent interactions between transitions. Two tests are equivalent, with respect to the affecting interaction pattern, if both tests result in the same interaction pattern. For example, consider the model of Figure 8. The regression test suite contains the following two tests:

- **Test #1:** $T_1, T_4, T_5, T_7, T_{10}, T_9, T_8$
- **Test #2:** $T_1, T_2, T_4, T_5, T_7, T_{10}, T_8$

Both of these tests result in the same affecting interaction pattern shown in Figure 9, therefore, these two tests are equivalent with respect to the affecting interaction pattern. Notice, the ghost dependencies are included in the dependence graph and, for the purpose of presentation, are...
shown as light lines whereas data and control dependencies are shown in bold.

![Figure 9. Affecting interaction pattern](image)

5.2.2. Testing the affect of the deleted transition on the model. The goal of this type of testing is to test the affect of the deleted transition on the model. Deletion of a transition can cause elimination of the dependence associated with the deleted transition where some transition was dependent on the deleted transition. The eliminated dependence is referred to as an affected ghost data dependence. We define this dependence as follows: Suppose Ti is a deleted transition. There exists an affected ghost data dependence between Ti and Tj iff: (1) in state Sb(Ti) event E(Ti) occurs, (2) condition \( C(T_i) \) evaluates to \( \text{TRUE} \), (3) suppose transition \( T_k \) is triggered (\( T_k \) must be a newly added transition), (4) there exists a definition clear path from \( T_k \) to \( T_j \) with respect to variable \( v \), and (5) \( T_k \) is not a definition of \( v \) (a case when \( T_k \) is a definition of \( v \) is covered by testing the addition of \( T_k \) as this is a 'new' dependence). For example, there exists a ghost data dependence between transitions \( T_6 \) and \( T_5 \), in the modified model of Figure 8.

Suppose transition Ti has been deleted from the original model. There exists a data dependence between \( T_j \) and \( T_k \) with respect to variable \( v \) in the original model. Due to deletion of transition \( T_i \), the data dependence \( (T_j, T_k) \) ceases to exist in the modified model. This dependence is referred to as a ghost activation dependence. More formally: there exists a ghost activation dependence between \( T_i \) and \( (T_j, T_k) \) iff: (1) there exists a definition clear path from \( T_i \) to \( S_b(T_i) \) with respect to \( v \), (2) event \( E(T_i) \) occurs in \( S_b(T_i) \) and condition \( C(T_i) \) evaluates to \( \text{TRUE} \), (3) suppose transition \( T_n \) is triggered \( (T_n \) is a newly added transition), and (4) there exists a definition clear path from \( T_n \) to \( T_k \) with respect to \( v \).

During traversal of a test, data, control and ghost activation dependencies that occur during the traversal are identified and marked in the EFSM dependence graph. In the next step, all unmarked dependencies are removed from the EFSM dependence graph. Then, all the dependencies, in the dependence sub-graph that are "affected" by the deleted transition are identified by traversing forward from the deleted transition and marking all traversed dependencies. All unmarked dependencies are removed from the sub-graph. The resulting dependence sub-graph is referred to as an Affected Interaction Pattern, where data, control, and ghost activation dependencies represent interactions between transitions. Two tests are equivalent, with respect to the side-effect interaction pattern, if both tests result in the same interaction pattern.

5.3. Reducing regression test suite

Any change to a model consists of a set of elementary modifications: elementary additions and elementary deletions. For example, consider the model of Figure 11. In this model, transition \( T_5 \) was replaced with the transitions \( T_{10} \) and \( T_{11} \).
The problem of test reduction is to find a test suite that satisfies the properties of the reduced test suite. There may be many test suites that satisfy the properties of the reduced test suite. One may try to identify a test suite of a minimum size that satisfies the properties of the reduced test suite.

Based on the initial experimentation with the presented approach, we observed significant reduction of regression test suites. For example, for the modified EFSM models presented in Figure 4, Figure 8, and Figure 11, we have observed 83% to 99% reduction in the regression test suite size.

Notice that the number of interaction patterns associated with modifications is bounded by the number of possible dependence sub-graphs. Therefore, the maximum size of the reduced test suite is bounded by the number of possible interaction patterns, regardless of a test strategy used in the selective test suite generation. We believe that our approach may lead to significant reduction of the regression test suite regardless of the size of the EFSM model and the number and complexity of modifications.

6. Conclusion

In this paper, we have presented a novel approach of model-based regression test reduction. The approach automatically identifies the difference between the original model and the modified model as a set of elementary model modifications. For each elementary modification, for each test in the regression test suite, interaction patterns are identified based on the EFSM dependence analysis. These patterns are used to reduce the regression test suite. Our initial experience shows that the approach may significantly reduce the size of regression test suites. The presented approach is currently under development. In the future, we plan to perform an experimental study to investigate the presented approach of regression test reduction for different types of system models, including industrial models, to determine the effectiveness of the presented approach and the quality (fault detection capability) of reduced test suites. In addition, we plan to develop efficient algorithms of test suite reduction and investigate the issue of minimization of reduced test suites.

7. References


