NOTE TO USERS

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Panitee RITTHIRUANGDECH

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Test Suite Reduction using SDL and EFSM Dependency Analysis

H. Ural

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

J.P. Corriveau

R. Probert

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Test Suite Reduction using SDL and EFSM Dependency Analysis

Panitee Ritthiruangdech

A thesis
*Submitted to the Faculty of Graduate and PostDoctoral Studies of the University of Ottawa in Partial Fulfillment of the Requirements for the Degree of Masters in Computer Science.**

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario

---

* The Masters program in Computer Science is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

Your file Votre référence
ISBN: 0-494-01593-4
Our file Notre référence
ISBN: 0-494-01593-4

Abstract

Requirement-based test generation is a model-based technique for generating suites of test cases related to individual requirements. Requirement-based test suites can be constructed manually or automatically using a coverage criterion; however, the constructed test suite's size in general is very large and the cost of executing considerably large number of test cases is very expensive. Hence, the problem of test suite reduction arises.

A reduction of a requirement-based test suite can be achieved without significantly reducing the fault-detection capabilities of the original test suite. This is done by eliminating all but one of the equivalent test cases from each class of equivalent test cases of the original test suite. A requirement-based reduction technique proposed in [1] uses EFSM dependency analysis to define classes of equivalent test cases. Two types of dependencies, namely control and data dependencies, are identified in an EFSM/SDL model. Analysis of these dependencies yields patterns of interaction among the elements of the EFSM/SDL model that affect a requirement under test. The patterns of interaction are in turn used to identify equivalent test cases w.r.t. the requirement under test, i.e., two tests are considered equivalent w.r.t. the requirement under test if both exhibit the same interaction pattern; hence, one of them can be discarded from the test suite.

In this thesis, based on [1], we have proposed algorithms to generate interaction patterns of a test case w.r.t. a requirement under test, algorithms to compare interaction patterns and determine whether or not they are equivalent, and an algorithm to identify a set of interaction patterns (w.r.t. the requirement under test) that are not covered by any

test case from a given test suite. Also, *Test Suite Reduction* (TSR) program has been developed based on these algorithms, which contributes towards object oriented testing.

Acknowledgements

I would like to acknowledge my supervisor, Dr. Hasan Ural, for introducing me to the area of software testing. My thanks to him for his guidance, patience and support, especially for giving me feedback at amazing speed. I am also thankful to Drs. Jean-Pierre Corriveau and Robert L. Probert for their useful comments on my thesis.

My studies would not have been possible without the financial support of the Royal Thai Government, CITO and the University of Ottawa. Especially, I gratefully thank the Royal Thai Government for providing me with an opportunity to study abroad.

I would like to thank my fellow ASERT members: Tuong, Olfa, Bo and Yan, for giving me very useful discussions and support as well as technical help. Also, I especially wish to thank Stephane for his invaluable advice and his great help.

I must finally acknowledge the support I received from my family, Dunn's family and my friends: Sunida, Dr. Suwannee and Abdel. Their moral support and continuous encouragement have been tremendously helpful in the moments of difficult time every international graduate student encounters.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Software testing is widely used to ensure software quality. It is one of the most commonly used techniques to reveal the existence of faults in a computer program. However, software testing is a very labor-intensive and expensive process. It accounts for approximately 50% to 70% of the cost of software development [9]. If testing could be done automatically, this could significantly reduce the cost of developing software and improve time-to-market. Therefore, in recent years, a considerable amount of work has been done on the subject of automated test generation techniques with respect to the related economic challenge. In general, there are three types of automated test generation techniques: "white-box" (code-based) test generation [22, 24, 32], "grey-box" (design-based) test generation [36] and "black-box" (specification-based) test generation [25, 27], depending on the artifact used for the construction of test cases. Test cases are derived from specification, design and source code of the program under test in black-, grey- and white-box testing, respectively [9, 17]. This thesis deals with black-box testing, in particular, test suite reduction based on requirements.

Model-based testing [26, 34, 35] is a black-box testing technique that generates a suite of test cases from a system model. In fact, this technique is well-suited for state-based systems that can be modeled using formal models and description languages such as Extended Finite State Machine (EFSM) [27], Specification and Description Language (SDL) [8, 29, 30, 33], ESTELLE [23, 28, 31] and LOTOS [7]. However, test cases

generated from model-based testing are not necessarily associated with individual requirements [21]; therefore, testers do not know which test case covers which individual requirement. As a consequence, it is hard to ascertain whether individual requirements are sufficiently tested or not.

Requirement-based test generation is a model-based technique for generating suites of test cases related to individual requirements [21]. In this technique, individual requirements are described in SDL. This technique may be supported by automated generation of an EFSM model from individual SDL requirements. Then the EFSM may be used to automatically generate test cases related to individual requirements. Several requirement-based selective test generation strategies were proposed to support partial system testing with respect to a set of selected requirements [21]. Test suites generated by requirement-based selective testing are considerably smaller than those generated by complete system testing; however, the number of test cases may still be very large. Since the cost of executing a large number of test cases and analyzing their results is very expensive, the problem of test suite reduction arises.

In [1], an approach of reduction of requirement-based test suites using EFSM dependency analysis was proposed. Two types of dependencies, namely control dependency and data dependency are identified in an EFSM model representing the system requirements. Those dependencies are analyzed to yield patterns of interaction among the elements of the EFSM model. Subsequently, equivalent tests are identified: two tests are equivalent if both exhibit the same pattern of interaction w.r.t. the requirement under test. Therefore, one of them can be removed from the test suite.

This thesis places the approach in [1] in a formal setting, expands the applicability of the approach in [1] by extending the subset of SDL and that of EFSM considered for the representation of the requirements, then implementing the concept of test suite reduction and finally identifying patterns of interactions that are not covered by any test case in the test suite.

1.2 **Contribution of the Thesis**

Reduction of requirement-based test suites using EFSM dependency analysis is studied in this thesis, which contributes towards object oriented testing. Given a single EFSM (or a single SDL process), we provide two methods to reduce the size of a given test suite based on two types of dependency analysis in the given EFSM: static dependency analysis and dynamic dependency analysis, which respectively yield static and dynamic interaction patterns. In this work, the approach in [1] has been placed in a formal setting, and expanded to cover larger subsets of SDL and EFSM based representations of requirements. Test suite reduction algorithms based on those two methods have been developed. Then, the approach in [1] has been extended to identify patterns of interactions that are not covered by any test case in the test suite (lack of coverage of such patterns may affect the fault detection capability of the test suite). Finally, a test suite reduction tool has been implemented using C++ and case studies have been performed to confirm the expected reduction in the size of test suites.

The differences between the approach in [1] and this thesis are as follows: this thesis

- places the approach in [1] in a formal setting,

- expands the applicability of the approach in [1] by extending the subset of SDL and that of EFSM considered for the representation of the requirements i.e., set, reset and procedure calls are allowed in this thesis,
- extends the approach in [1] to evaluate the adequacy of the reduced test suite based on the coverage of interaction patterns, i.e., identifies patterns of interactions that are not covered by any test case in the given test suite; therefore, enhances the capability of the reduced test suite in its interaction coverage,
- proposes algorithms for test suite reduction,
- implements the algorithms proposed in the thesis,
- performs case studies.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 reviews formal description languages and test derivation methods, and introduces the concepts related to dependency analysis in EFSM. Algorithms for the reduction of requirement-based test suites using EFSM dependency analysis are presented in Chapter 3. Based on two types of dependency analysis (static and dynamic), two test suite reduction methods are presented. Chapter 4 presents the Test Suite Reduction program, which has been developed based on algorithms introduced in Chapter 3. Our implementation has been evaluated through four case studies, whose results are presented and analyzed in Chapter 5. Chapter 6 presents our conclusions, with a summary of contributions and directions for future research.

Chapter 2

Formal Description Languages, Testing Methods and Dependency Analysis

2.1 Formal Description Languages

Generally, software system specifications consist of individual requirements, which are expressed informally in a textual format - e.g., English - that may be ambiguous, inconsistent and incomplete. Hence, models and formal description techniques such as EFSM [18], SDL [5], LOTOS [7] and Estelle [6] are used to describe requirements in order to eliminate problems associated with informal specifications.

2.1.1 An overview of SDL

The Specification and Description Language (SDL) [19] is a formal description technique developed by ITU (International Telecommunications Union) for the specification of telecommunication systems. In SDL, a system can be specified in terms of the external behavior of a collection of *blocks* and *processes* that communicate with each other and with the environment by sending and receiving *signals* via *channels* or *signal routes*. Such specifications can be expressed either in textual or in graphical representations.

An SDL specification consists of:

- a system identifier;
- pre-defined data types;
- blocks (used to describe the static structure of a system);
- channels (used to describe the communication links between a block to other blocks and/or to the environment of the system). Each channel is specified by a

channel identifier, its end points (block or environment), and the set of signals exchanged over it;

- and signals (packets of information exchanged through the channels). Each signal is specified by a signal identifier and the data type of each parameter.

In turn, a block is made up of:

- a block identifier;
- the definition of each process existing in the block;
- the signal routes (the communication links between a process and the other processes in the block and/or to the environment of the block);
- the connection between the signal routes and the channels external to the block; and the signals (local to the block) exchanged over signal routes.

Finally, a process definition consists of:

- a process identifier;
- formal parameters;
- procedure descriptions;
- declarations of local variables;
- timers;
- and the specification of the behavior of the process.

The behavior of each SDL process is described in the form of an extended finite state machine (EFSM), defined in the next section. Each transition is characterized by the conditions (that must be evaluated to be true for the transition to be executed) and the actions (that must be performed during an execution). The actions may assign or modify the values of variables, initiate output signals, start or stop timers, execute decision

statements, perform procedure calls, create new process instances and terminate its own execution. In our work, we do not consider decision statements and process creation in an SDL process.

Figure 2.1 summarizes the most important elements of an EFSM with their graphical and textual representation and Figure 2.2 shows a fragment of an example SDL process.

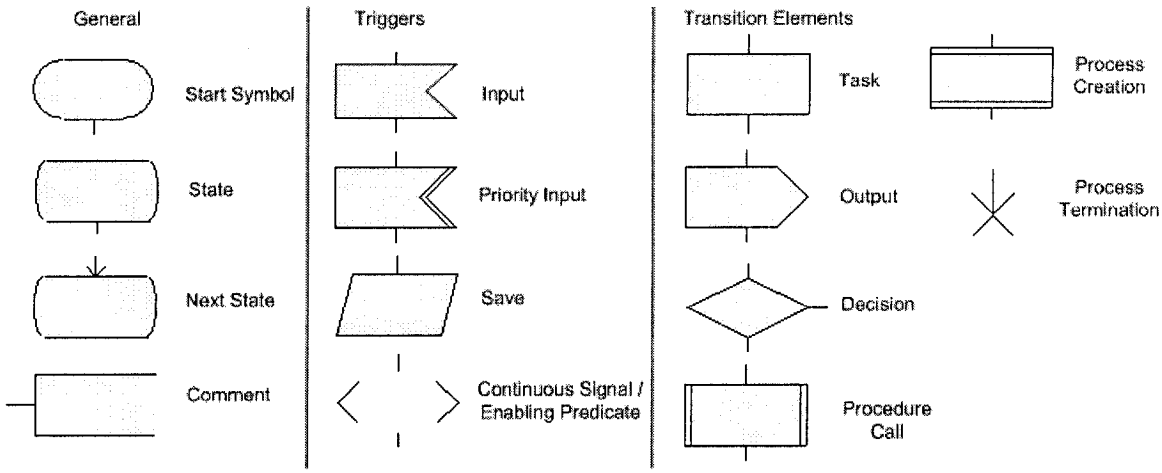


Figure 2.1 SDL-2000 EFSM elements

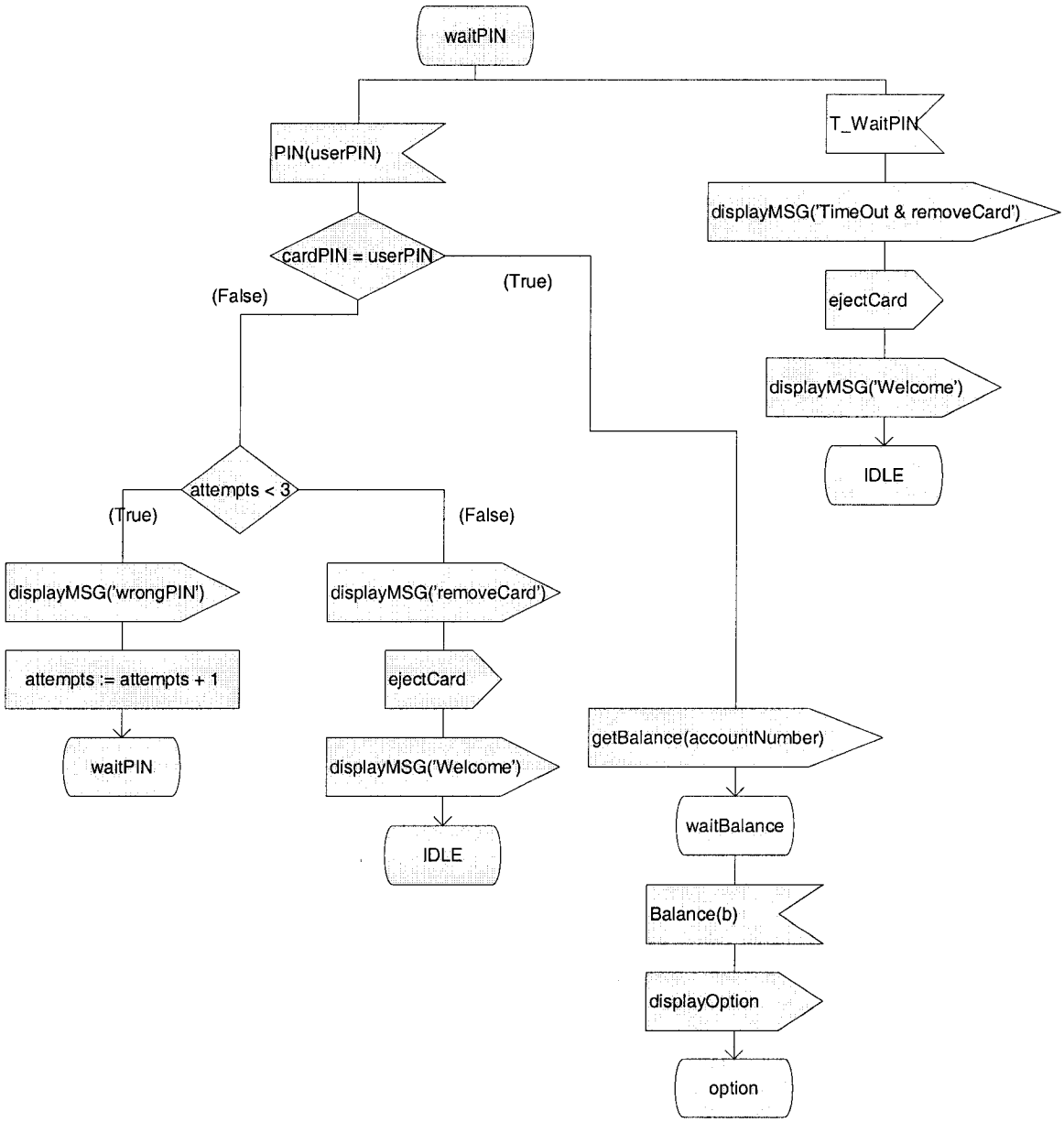


Figure 2.2 Fragment of an SDL process for verifying a PIN in an ATM System

2.1.2 EFSM

Extended **F**inite **S**tate **M**achines (EFSM) [18] have been widely used to model many types of systems, especially state-based systems found in telecommunications and computer communication networks. EFSM is an extension of the classical Finite State Machine (FSM), which adds variables, enabling predicate, and actions into a transition.

An EFSM can be formally represented as a seven-tuple (S, s_{en}, s_{ex}, I, O, V, T), with the following definitions:

S finite set of states

s_{en} entry state

s_{ex} exit state

I finite set of input interactions

O finite set of output interactions

V finite set of variables

T finite set of transitions

Each element of T is a 5-tuple $t = (s_s, s_d, i, g, a)$ where s_s and s_d are states in S representing the state from which t is outgoing and the state to which t is incoming, respectively. i is an input interaction in I that triggers t, g is a Boolean condition that must be evaluated to be true for t to be executable, and a is a sequence of actions that takes place when t is executed. An action may be an assignment, output, set, reset or procedure call statement. Note that, in general, an EFSM model may include, in actions part of a transition, conditional statements such as *if-then-else* or *case* statements as well as repetitive statements such as *for* or *while* statements. In our work, we do not consider EFSMs where transitions have conditional and repetitive statements. Interested reader may refer to [19] for handling such statements within the context of the EFSM model.

An EFSM can also be graphically represented by a digraph G = (V, E) where V is a set of nodes, each representing a state in S and E is a set of edges, each representing a transition in T. For instance, Figure 2.3 shows an EFSM describing the requirements for a simplified ATM System. "Requirements" are traditionally non-executable. They are

usually expressed in a natural language, declarative (non-imperative) form. In this thesis, we assume that requirements can be represented as a single EFSM and each requirement can be adequately represented by a single transition in this EFSM. For example, the set of requirements of the ATM system is $R = \{r1, r2, r3, r4, r5\}$: where each r is represented by a single transition in the corresponding EFSM:

r1: User fails to enter a correct pin that matches with the PIN stored in the ATM card in less than four attempts. If pins don't match, and less than four attempts were performed, the system displays an error message, increments the number of attempts and prompts for pin. At the fourth attempt, if user still fails to enter a correct pin, the system prints an error message and ejects the user's ATM card.

r1 = T2 T2 T2 **T3**

r2: After entering a correct pin in less than four attempts, user selects withdrawal function. The system adjusts the balance, and displays a menu with withdrawal, deposit, balance inquiry, and exit functions.

r2 = x T4 **T5**

r3: After entering a correct pin in less than four attempts, user selects deposit function. The system adjusts the balance, and displays a menu with withdrawal, deposit, balance inquiry, and exit functions.

r3 = x T4 **T6**

r4: After entering a correct pin in less than four attempts, user selects balance function. The system displays the balance, and then a menu with withdrawal, deposit, balance inquiry, and exit functions.

r4 = x T4 **T7**

r5: After entering a correct pin in less than four attempts, user selects exit function and the system ejects the user's ATM card.

r5 = *x* T4 **T9**

x: T2 may be inserted 0, 1, 2 or 3 times

The transition in bold in each sequence of transitions given above is the *transition under test* (*TUT*) which stands for the requirement *r*.

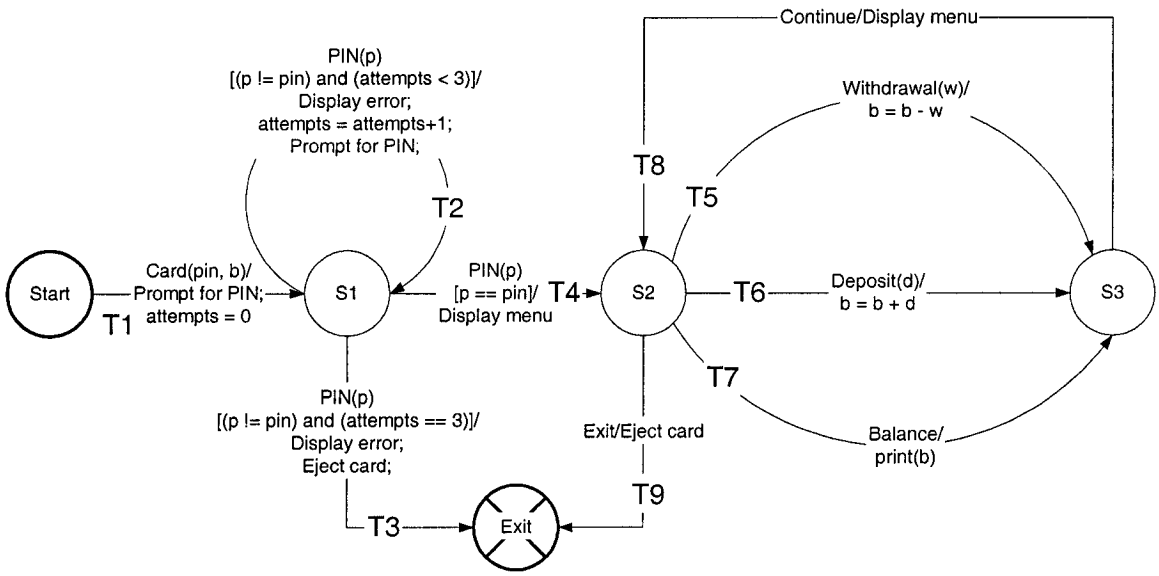


Figure 2.3 An EFSM for a Simplified ATM System [1]

2.1.3 Conversion from SDL Models into EFSM Models

The behavior of a process in SDL can be rewritten as an EFSM. During this conversion, all states in the process are preserved in the EFSM. A path between two states of the process (not including other states) is mapped into a transition between two corresponding EFSM states such that the signal and conditions are mapped into an input and a predicate associated with this transition. A sequence of tasks, output signals emission, timer settings, and procedure calls on the path is mapped into actions of the EFSM transition. In particular, an SDL timer is represented by a variable in an EFSM;

therefore, *start* and *stop* timers in SDL are mapped into *set* and *reset* in an action of EFSM transition, respectively.

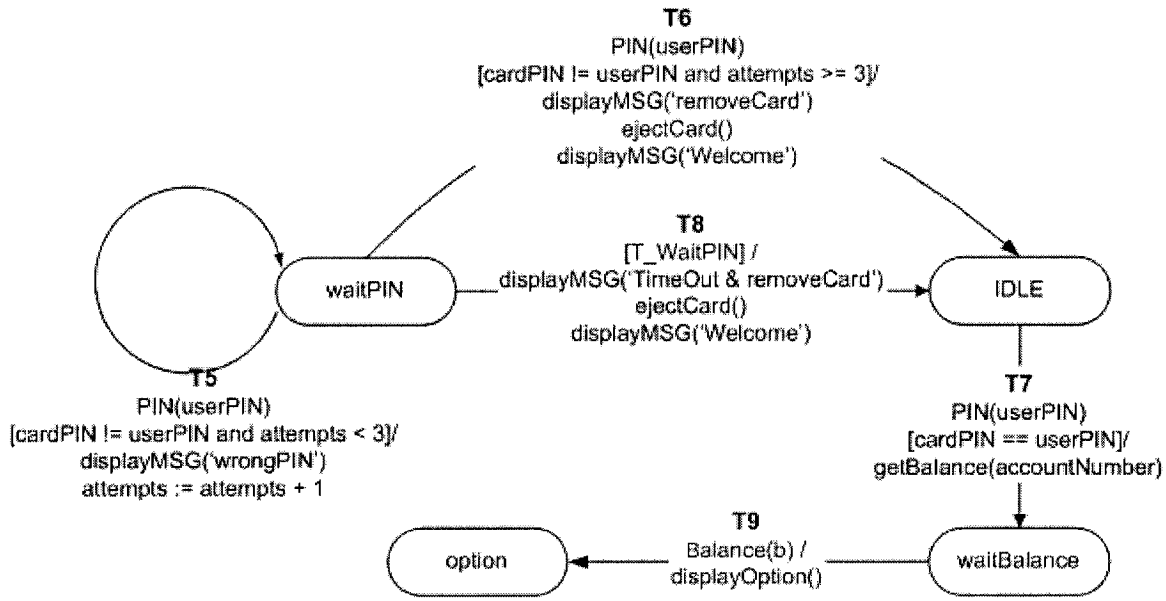


Figure 2.4 Fragment of the EFSM System Model

Figure 2.4 shows a partial representation of the EFSM model that corresponds to the fragment of an SDL process shown in Figure 2.2. In the EFSM, each state transition has an optional boolean predicate that must be evaluated to be true for this transition to be executed. For example, in Figure 2.4, suppose that the current state is *waitPIN* and the wrong userPIN is entered. If the maximum number of attempts (in this case, 3) has not been exceeded, transition T5 is taken; otherwise transition T6 is taken instead.

2.2. Test Derivation Methods

Test derivation methods are broadly classified into black-box, grey-box and white box testing methods depending on whether the specification, design, or source code of a

system is used as the resource to derive tests. White box testing methods are traditionally divided into two categories: control flow and data flow oriented testing. In general, determining whether an implementation of a system establishes the desired flow of control and data expressed in its specification can be referred to as testing the control and data flow aspects of the implementation. Control flow oriented testing is based on the analysis of the control flow of the implementation, while data flow oriented testing is based on the analysis of the data flow of the implementation. The tests generated by the data flow oriented test coverage criteria are complementary to those constructed by control flow oriented test coverage criteria [17]. The white box test coverage criteria have been adapted to black-box (specification-based) testing for systems specified in Estelle [6], LOTOS [7], and SDL [5]. Since our work aims to produce tests from requirements of systems for both control flow oriented testing and data flow oriented testing, the commonly used control and data flow oriented test coverage criteria are discussed in the following sections. These criteria are based on directed graphs (digraphs) representing the source code or the specification of the system under test. As far as applicable, we transfer these methods to graphs representing EFSM models.

A *path* $(n_1, n_2, \ldots, n_m)$ in a directed graph G is a sequence of nodes in G, such that for all i, $1 \leq i \leq m\text{-}1$, $m \geq 2$, $(n_i, n_{i+1}) \in E$. A *sub-path* of a path $(n_1, \ldots, n_m)$ is a path $(i_1, \ldots, i_k)$ if there exists a ζ, $0 \leq \zeta \leq m\text{-}k$, such that for all j, $1 \leq j \leq k$, $i_j = n_{j+\zeta}$. A *loop-free path* is a path in which all nodes are distinct. A *complete path* in a single entry, single exit directed graph G is a path whose first node is the entry node and whose last node is the exit node. A complete path is *executable* or *feasible* if a set of input data which causes its execution exists, and *un-executable* (*infeasible*) otherwise. A path is executable if it is a sub-path of

an executable complete path. Whether a path is executable depends on the semantics of the system itself, not just on the underlying graph structure. It is recognized that guaranteeing feasibility of selected paths is an undecidable problem, and this is a fundamental problem in test case design.

Let p (n_1, ..., n_m) be a complete path in a directed graph G. We say that a node i is *covered* by p if $i = n_j$ for some j, $1 \leq j \leq m$. Similarly, an edge (i_1, i_2) is covered by p if $i_1 = n_j$ and $i_2 = n_{j+1}$ for some j, $1 \leq j \leq m$-1. A path (i_1, ..., i_k) is covered by p if path (i_1, ..., i_k) is sub-path of p. A node, edge, or path is covered by a set Π of complete paths of G if the node, edge, or path, respectively is covered by a complete path in Π.

Consider as an example, Figure 2.3 depicting a directed graph $G = (V, E)$ representing the EFSM of a simplified ATM system where V is a set of nodes, each representing a state and E is a set of directed edges, each representing a transition. In this graph, let Π be a set of complete paths starting at the start state and terminating at the exit state. For example, a complete path (T1, T4, T5, T8, T9) is in Π and covers nodes: *start*, *S*1, *S*2, *S*3 and *exit* states, covers edges: T1, T4, T5, T8 and T9 and covers a path: T1, T4, T5, T8, T9.

2.2.1 Control Flow Oriented Testing

Each control flow oriented test coverage criterion aims at generating a set of test cases that covers (i.e. causes execution of) all occurrences of a control flow oriented test unit (e.g. node, edge, etc.). There are various control flow oriented test coverage criteria: all-nodes [4,9,10], all-edges [4,9,10], all-paths [4,9,10], decision coverage (branch coverage), condition coverage, decision/condition coverage and multiple condition

coverage [9]. Let Π be a set of complete paths of a directed graph G.

- All-nodes (sometimes called statement coverage)

Π satisfies the all-nodes coverage criterion in G if every node of G is covered by Π at least once. This is the easiest and weakest of all the control flow oriented test coverage criteria.

- All-edges

Π satisfies the all-edges coverage criterion in G if every edge of G is covered by Π at least once.

- All-paths

Π satisfies the all-paths coverage criterion in G if every complete path of G is included in Π. This is the strongest criterion among the control flow oriented test coverage criteria and is not generally practical or feasible.

- Decision Coverage (sometimes called branch coverage)

A decision is a boolean expression [9]:

<decision> := <component> AND <component>

| <component> OR <component>

| NOT <component>

A component of a decision can be a decision or a condition, which is defined as follows:

<condition> := <var1> relation <var2>

<relation> := = | > | < | ≥ | ≤ | ≠

where var1, var2 may be optionally replaced by function names or constants.

Π satisfies the decision coverage criterion [9] (or branch coverage criterion) in G if every possible outcome of each decision (i.e. branch) in G is covered by Π at least once.

However, the criterion does not imply that all combinations of conditions in each decision are covered; thus some faults present in a particular condition may not be exposed. For example, for testing the decision (A and B), the decision coverage criterion can be satisfied by two test cases: (A = true, B = true) and (A = true, B = false); however, neither test case causes A to be false. This could mask a possible fault in condition A. Hence, a shortcoming of decision coverage is that it does not guarantee coverage of each condition within each decision.

- Condition Coverage

Π satisfies the condition coverage criterion [9] in G if each possible outcome of every condition within each decision in G is covered by Π at least once. Even though the criterion requires that each possible outcome of every condition within each decision will be executed, it does not guarantee that every possible outcome of each decision will be taken; therefore, it does not guarantee the coverage of all statements or all branches in the graph. For example, for testing the decision (A or B), two test cases: (A = true, B = false) and (A = false, B = true) would satisfy the condition coverage criteria, but it would not fulfill the decision coverage criterion because it would fail to test the false outcome of the decision.

- Decision/Condition Coverage

Π satisfies the decision/condition coverage criterion [9] in G, if each outcome of each decision and each outcome of each condition in G is covered by Π at least once. Developed to overcome the deficiencies of decision coverage and condition coverage

alone, decision/condition coverage ensures that each outcome of each decision and each outcome of each condition will be executed. However, the fault of using "and" instead of "or" in a decision may still not be covered. For example, for testing the decision (*A* and *B*), two test cases: (*A* = true, *B* = true) and (*A* = false, *B* = false) would actually satisfy decision, condition and decision/condition coverage, but it would be possible to replace the "and" with an "or" without affecting the result of the tests.

- Multiple Condition Coverage

Π satisfies the multiple condition coverage criterion [9] in *G*, if all possible combinations of condition outcomes in each decision in *G* is covered by Π at least once. Multiple condition coverage eliminates the deficiencies of the previous criteria, since test cases must be generated to exercise all possible combinations of all condition outcomes in each decision.

It is widely accepted that all-nodes coverage is rather weak, and that the branch coverage criterion is a minimal standard of coverage in white box testing [17]. Other criteria (multiple condition coverage, decision/condition coverage, condition coverage) are difficult or even impossible to achieve in software of more than moderate complexity [17].

2.2.2 Data Flow Oriented Testing

Basically, data flow oriented testing is based on data flow analysis. Data flow analysis establishes certain associations between definitions and uses of variables. Such associations are identified by tracking variables as they are defined and modified until they are ultimately used to compute values for other variables or generate output values. Data flow oriented test coverage criteria require each association between the definition

of a variable and its uses to be examined at least once during testing. The motivation behind the selection of tests based on the coverage of data flow associations is that faults in a system may lead to incorrect values and as the result of propagation through computations, an erroneous result may show up at the system's output.

2.2.2.1 Data Flow Related Concepts

Data flow analysis was originally used for compiler optimization [3]. It aims to trace the use of a program's variables as they are initialized and modified during program execution. Data flow analysis classifies an occurrence of a variable as part of a statement associated with a node or as part of a Boolean expression associated with an edge in a graph representing the software as a *definition* or a *use* [10,11]. Data flow analysis techniques can be applied to the selection of test sequences from an EFSM by transforming the EFSM into a special flow graph F [13,14,15] which is a digraph (N, *en*, *ex*, E) where $N = \{n \mid n$ is an s-node, i-node or t-node$\}$; *en* and *ex* $\in N$ are the entry node and the exit node, respectively; $E = \{e \mid e$ is an *it*-edge, *si*-edge or *ts*-edge$\}$.

To transform an EFSM $M = (S, s_{en}, s_{ex}, I, O, V, T)$ into a flow graph $F = (N, en, ex, E)$, each state $s \in S$, each transition $t \in T$ and each input $i \in I$ at each state s are turned into an s-node, a t-node and an i-node in F, respectively, where s_{en}, s_{ex}, are correspondingly mapped to *en* and *ex*. In addition, the control flow in M is affected by a predicate; therefore, each predicate g in a transition (s_s, s_d, i, g, a) is associated with an *it*-edge. Note that an *si*-edge and a *ts*-edge are employed in F solely to complete the control flow of M.

More formally, to map an EFSM $M = (S, s_{en}, s_{ex}, I, O, V, T)$ into a flow graph $F = (N, en, ex, E)$, we proceed as follows:

Let $s \in S$ be a state, $in \in I$ be an input symbol,

$T_s = \{(s_s, s_d, i, g, a) \in T \mid s_s = s\}$, $T_{s,in} = \{(s_s, s_d, i, g, a) \in T_s \mid i = in\}$ and

$I_s = \{in \in I \mid T_{s,in} \neq \emptyset\}$.

For each $s \in S$ of M, F consists of

- one s-node s,
- one i-node for each $i \in I_s$
- one t-node for each $t \in T_s$
- one si-edge from s-node to each i-node, $i \in I_s$
- one it-edge from each i-node, $i \in I_s$ to each t-node, $t \in T_{s,i}$
- one ts-edge from each t-node, $t \in T_s$ to the s-node s' such that $t = (s_s, s', i, g, a)$

Figure 2.5 shows the flow graph F for the EFSM of Figure 2.3. For example, with state (start) and input *card*(*pin*, *b*), the following can be observed in the graph:

s-nodes : start, S1

i-nodes : *card*(*pin*, *b*)

t-nodes : T1

si-edges : (start, {*card*(*pin*, *b*)})

it-edges : (*card*(*pin*, *b*), T1), ts-edges : (T1, S1)

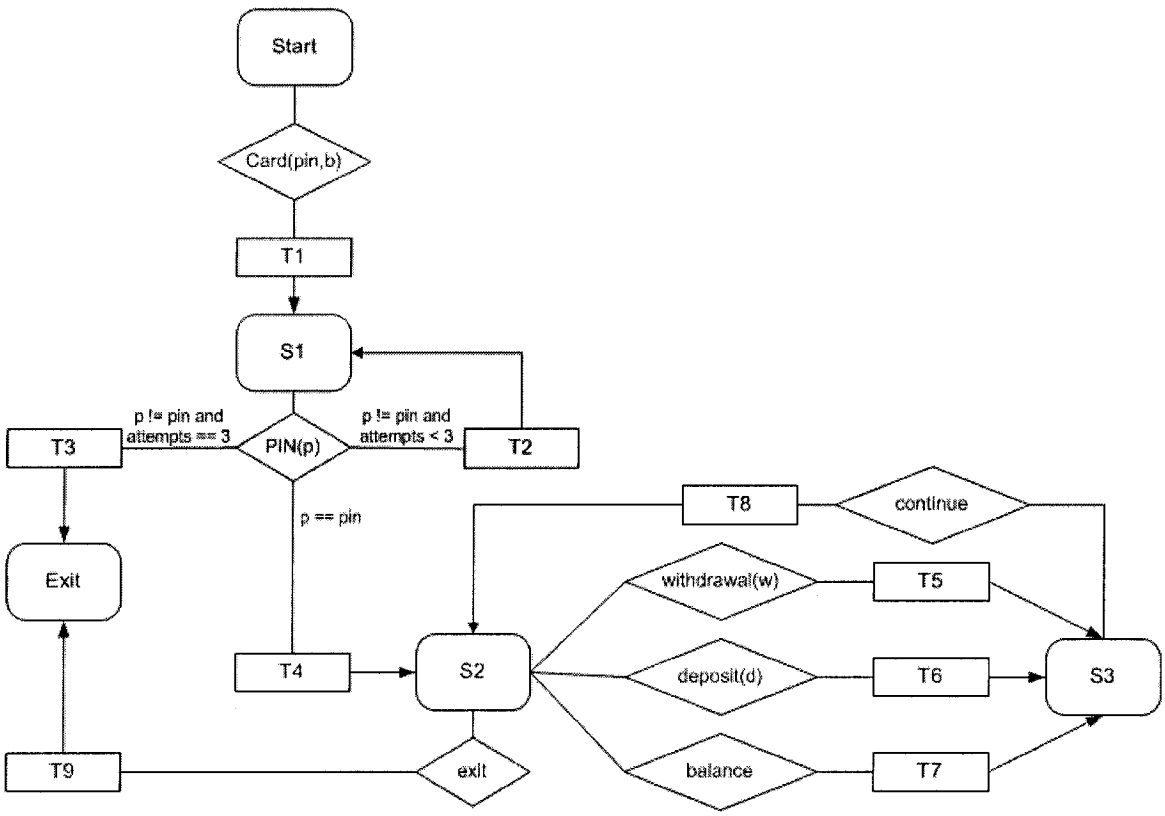


Figure 2.5 Flow graph *F* generated from the ATM EFSM

A *definition* (referred to as def) of a variable v at node n (denoted by d_n^v) is an occurrence of v by which a value is assigned to v (e.g., an occurrence of v on the left hand side of assignment statement, for instance, the definition of "*attempts*" at node T1 in Figure 2.3). A use of a variable v is an occurrence of v by which the value of v is referenced. Each use occurrence is also further classified as a computation use (c-use) or a predicate use (p-use) [10,11]. A c-use of a variable v at node n (denoted by c_n^v) is a use of v that occurs in the RHS of assignment statements, arguments in a procedure call or output statement, for instance, the use of variable "*b*" at node T5 in Figure 2.5. On the other hand, a p-use of a variable v on edge (m, n) (denoted by $p_{(m,n)}^v$) is a use that occurs

in Boolean expressions in a condition statement or in a repetitive statement, for instance, the use of variable "*pin*" on the edge ({*pin*}, T4) in Figure 2.5.

2.2.2.2 Classification of Variable Occurrences

The following convention is used to classify each variable occurrence in a flow graph F derived from an EFSM E as a def, c-use, or p-use:

(a) an input event $e(X_1, \ldots, X_n)$ in an i-node contains c-uses of the actual signal parameters followed by defs of the variables $X_1,\ldots, X_n$; in the special case of an input from the environment, it contains only the defs of the variables $X_1,\ldots, X_n$.

(b) an output event $e(X_1, \ldots, X_n)$ in a t-node contains c-uses of the variables $X_1,\ldots, X_n$ followed by defs of the actual signal parameters; in the special case of an output to the environment, it contains only the c-uses of the variables $X_1,\ldots, X_n$.

(c) an assignment statement Y := expression in a t-node contains c-uses of the variables occurring in the expression followed by a def of Y.

(d) a set statement set(expression, timer_id) in a t-node contains c-uses of the variables occurring in the expression followed by a def of the timer-id.

(e) a reset statement reset(timer_id) in a t-node contains a c-use of the timer-id.

(f) a Boolean expression on an it-edge contains p-uses of all variables occurring in the Boolean expression

(g) a procedure call[1] statement $\pi\ (X_1,\ldots, X_m, e_{m+1},\ldots,e_n)$ in a t-node contains a c-use of each variable X_i $(1 \leq i \leq m)$ and a c-use of each variable Y_j occurring in an expression e_k $(m+1 \leq k \leq n)$, followed by a def of each X_i.

---

[1] A *procedure call* is in the form $pi(X1,\ldots,Xm,e_{m+1},\ldots,e_n)$ where pi is the procedure identifier, $X1,\ldots,Xm$ are variables representing actual *in/out parameters*, and $e_{m+1},\ldots,e_n$ are expressions representing actual *in parameters*

Note that:

1) expression e is a function(v_1, ..., v_n) or constant, where v_1, ..., v_n are variables.

2) The timer-id is represented by a variable.

Data flow information in a directed graph F *is* organized by associating with each i-node the set: def(i) = {variables which have definitions at i-node i}, associating with each t-node the sets: def(t) = {variables which have definitions at t-node t}, c-use(t) = {variables which have c-uses at t-node t} and associating with it-edge, the set p-use (i, t) = {variables which have p-uses on it-edge (i, t)}.

Table 2.1 The def sets associated with *i*-nodes in Figure 2.5

i-node	def sets
card(*pin*, *b*)	{*pin*, *b*}
PIN(*p*)	{*p*}
withdrawal(*w*)	{*w*}
deposit(*d*)	{*d*}

Table 2.2 The def sets and c-use sets associated with *t*-nodes in Figure 2.5

t-node	def sets	c-use sets
T1	{*attempts*}	Ø
T2	{*attempts*}	{*attempts*}
T3	Ø	Ø
T4	Ø	Ø
T5	{*b*}	{*b*, *w*}
T6	{*b*}	{*b*, *d*}
T7	Ø	{*b*}
T8	Ø	Ø
T9	Ø	Ø

Table 2.3 The p-use sets associated with *it*-edges in Figure 2.5

it-edge	p-use sets
({PIN(*p*)}, T2)	{*attempts*, *p*, *pin*}
({PIN(*p*)}, T3)	{*attempts*, *p*, *pin*}
({PIN(*p*)}, T4)	{*p*, *pin*}

A path (n_1, n_2, ..., n_{k-1}, n_k) is a *def-clear path* with respect to a variable v from node n_1 to node n_k or from node n_1 to edge (n_{k-1}, n_k) if there are no definitions of v at nodes n_2 to n_{k-1}.

Based on def, c-use and p-use sets, we define sets of nodes dcu(v, i) = {node j such that $v \in$ c-use(j) and there is a def-clear path w.r.t. v from i to j} and dpu(v, i) = {edge (j, k) such that $v \in$ p-use(j, k) and there is a def-clear path w.r.t. v from i to (j, k)}.

Def-use associations can be categorized into two classes namely def-c-use association and def-p-use association. A *def-c-use* association is a triple (i, j, v) where $v \in$ def(i) and $j \in$ dcu(v, i). A *def-p-use* association is a triple (i, (j, k), v) where $v \in$ def(i) and (j, k) $\in$ dpu(v, i). Accordingly, a *def-use* association is either a def-c-use association or a def-p-use association. A def-use association is also called a *du-pair.* A def-use association is feasible (executable) if a def-clear path related to the association is a sub-path of some executable complete path; otherwise, it is infeasible (unexecutable.)

A path (n_i, ..., n_j, n_k) is a *du-path* w.r.t. variable v if n_i has a definition of v and either

- n_k has a c-use of v and (n_i, ..., n_j, n_k) is a def-clear path w.r.t. v, or
- (n_j, n_k) has p-use of v and (n_i, ..., n_j, n_k) is a def-clear loop-free path w.r.t. v.

For example, consider a def-c-use association (*b*, *card*(*pin*, *b*), T5). The definition of *b* at *i*-node *card*(*pin*, *b*) can reach the c-use of *b* at *t*-node T5 through the def-clear path "*card*(*pin*, *b*), T1, S1, *PIN*(*p*), T4, S2, *withdrawal*(*w*), T5". On the other hand, consider a def-p-use association (*card*(*pin*, *b*), (*PIN*(*p*), T4), *pin*). The definition of *pin* at *i*-node *card*(*pin*, *b*) can reach the p-use of *pin* on the *it*-edge (*PIN*(*p*), T4) through the def-clear path "*card*(*pin*, *b*), T1, S1, *PIN*(*p*), T4".

For the sake of presentation, for all du-pairs identified from F in Figure 2.5, an i-node representing input in will be replaced with a t-node $t = (s_s, s', i, g, a)$ where $in = i$ and it-edge (i, t) will be replaced by a t-node t' where $t' = t$. For example, $(d^{b}_{card(pin,b)}, c^{b}_{T5})$ and $(d^{pin}_{card(pin,b)}, p^{pin}_{PIN(p),T4})$ in F will be represented as (d^{b}_{T1}, c^{b}_{T5}) and $(d^{pin}_{T1}, p^{pin}_{T4})$, respectively. In addition, the def-clear path related to each du-pair is henceforth represented by only a sequence of t-nodes, for instance, a def-clear path, "*card*(*pin*, *b*), T1, S1, *PIN*(*p*), T4, S2, *withdrawal*(*w*), T5" is represented as T1 T4 T5.

For the example EFSM in Figure 2.3, the def-c-use associations and the def-p-use associations for each def of each variable as well as the corresponding def-clear paths are given in Table 2.4

Table 2.4 Def-use Associations and the Corresponding Def-clear Paths for Figure 2.3

Variable	Def(s)	Use(s)	def-clear path
attempts	$d^{attempts}_{T1}$	$c^{attempts}_{T2}$, $p^{attempts}_{T2}$	T1 T2
	$d^{attempts}_{T2}$	$p^{attempts}_{T2}$, $c^{attempts}_{T2}$	T2 T2
		$p^{attempts}_{T3}$	T2 T3
b	d^{b}_{T1}	c^{b}_{T5}	T1 T4 T5
		c^{b}_{T6}	T1 T4 T6
		c^{b}_{T7}	T1 T4 T7
	d^{b}_{T5}	c^{b}_{T5}	T5 T8 T5
		c^{b}_{T6}	T5 T8 T6
		c^{b}_{T7}	T5 T8 T7
	d^{b}_{T6}	c^{b}_{T5}	T6 T8 T5
		c^{b}_{T6}	T6 T8 T6
		c^{b}_{T7}	T6 T8 T7
d	d^{d}_{T6}	c^{d}_{T6}	T6
p	d^{p}_{T2}	p^{p}_{T2}	T2
	d^{p}_{T3}	p^{p}_{T3}	T3
	d^{p}_{T4}	p^{p}_{T4}	T4

pin	d_{T1}^{pin}	p_{T2}^{pin}	T1 T2
		p_{T3}^{pin}	T1 T3
		p_{T4}^{pin}	T1 T4
w	d_{T5}^{w}	c_{T5}^{w}	T5

2.2.2.3 Data Flow Oriented Test Coverage Criteria

The motivation behind data flow oriented testing is to bridge the gap between all-paths and all-edges coverage criteria [10, 12]. A family of data flow oriented test coverage criteria is defined in [11]. The criteria attempt to cover various combinations between defs and uses of the same variable. Roughly speaking, the family of data flow oriented test coverage criteria is based on the requirement that the test cases execute def-clear paths from each node containing a definition of a variable to specified nodes containing *c*-uses and edges containing *p*-uses of that variable. For each variable definition, it is required that all/some def-clear paths with respect to that variable from the node containing the definition to all/some of the uses/c-uses/p-uses reachable by some such paths be executed. These criteria are defined precisely in Table 2.5.

Table 2.5 Some Data Flow Oriented Test Coverage Criteria

Criterion	Association Required
All-defs	Some (i, j, x) such that $j \in$ dcu(x, i) or some $(i, (j, k), x)$ such that $(j, k) \in$ dpu(x, i).
All-c-uses	All (i, j, x) such that $j \in$ dcu(x, i).
All-p-uses	All $(i, (j, k), x)$ such that $(j, k) \in$ dpu(x, i).
All-p-uses/some-c-uses	All $(i, (j, k), x)$ such that $(j, k) \in$ dpu(x, i). In addition, if dpu$(x, i) = \phi$ then some (i, j, x) such that $(j, k) \in$ dcu(x, i). Note that since i has a definition of x, dpu$(x, i) = \phi \Rightarrow$ dcu$(x, i) \neq \phi$.
All-c-uses/some-p-uses	All (i, j, x) such that $j \in$ dcu(x, i). In addition, if dcu$(x, i) = \phi$ then some $(i, (j, k), x)$ such that $j \in$ dpu(x, i). Note that since i has a definition of x, dcu$(x, i) = \phi \Rightarrow$ dpu$(x, i) \neq \phi$.

All-uses	All (i, j, x) such that $j \in$ dcu(x, i) and all $(i, (j, k), x)$ such that $(j, k) \in$ dpu(x, i).
All-du-paths	All du-paths from i to j with respect to x for each $j \in$ dcu(x, i) and all du-paths from i to (j, k) with respect to x from each $(j, k) \in$ dpu(x, i).

If variable x has a definition in node i, the all-defs coverage criterion requires the test cases to exercise some def-clear path w.r.t. x from node i to some node or edge at which the value assigned to x in node i is used. The all-uses coverage criterion requires the test cases to exercise at least one such path to each such node and to each such edge. The all-du-paths coverage criterion requires that all of the du-paths from node i to each such node and each such edge be exercised. The coverage criteria all-p-uses, all-c-uses, all-p-uses/some-c-uses, and all-c-uses/some-p-uses lay the emphasis upon either c-uses or p-uses. Note that any program has only finite set of def-use association, so none of the data flow oriented test coverage criteria requires and infinite number of test units. Figure 2.6 illustrates the subsumption relationships between control and data flow oriented test coverage criteria. A criterion A *subsumes* another criterion B, denoted A=>B, if a set of complete paths Π satisfying A also satisfies B. As we can observe from Figure 2.6, every data flow oriented test coverage criterion is stronger than all-edges coverage criterion. [4,9,10] provide more details on the advantages of data flow oriented testing over control flow oriented testing.

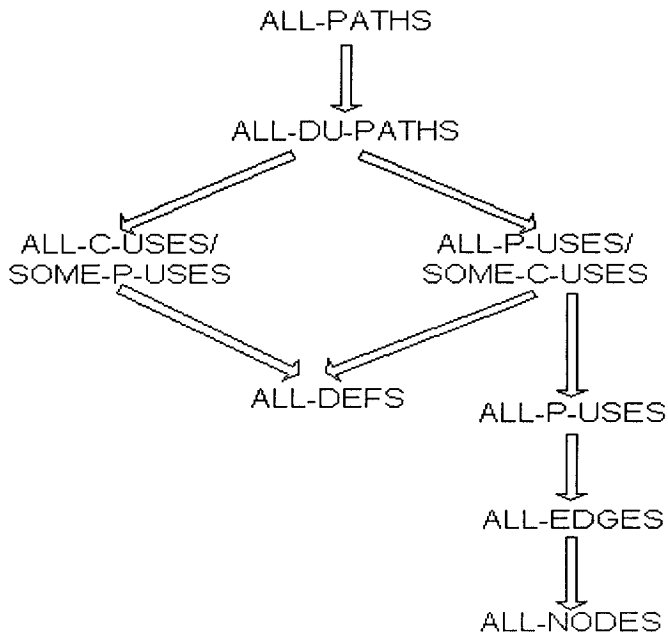


Figure 2.6 Relationships Among Test Coverage Criteria

The family of data flow oriented test coverage criteria mentioned above is based on the association between a definition of a variable and each use of that same variable or so-called "du-pair". However, there exists another type of relationship between variable occurrences. Such a relationship exists between the use of a variable and the def of *another* variable. IPO-df-Chains coverage criterion [16] takes both relations into account.

IPO-df-Chains coverage criterion is based on the analysis of associations between each output interaction and those input interactions which influence that output in an EFSM. Such associations can be defined through the notion of *affect*. A use u_{Tk}^{y} is affected by d_{Ti}^{x} if

- $x = y$ and there exists a du-pair (d_{Ti}^{x}, u_{Tk}^{y}), or

- there is a definition d^{z}_{Tk} given in terms of u^{x}_{Tk} such that (d^{x}_{Ti}, u^{x}_{Tk}) is a du-pair and a use of u^{z}_{Tj} is affected by d^{z}_{Tk}.

If du-pairs in an EFSM are linked by the notion of affect, *data-flow-chains* (*df-chains*) can be formed. A df-chain is an ordered sequence (d^{x1}_{T1}, u^{x1}_{T2}, d^{x2}_{T2}, u^{x2}_{T3}, d^{x3}_{T3} ... d^{xm}_{Tm}, u^{xm}_{Tm+1}) of du-pairs (d^{x1}_{T1}, u^{x1}_{T2}),(d^{x2}_{T2}, u^{x2}_{T3})...(d^{xm}_{Tm}, u^{xm}_{Tm+1}), such that $m \geq 1$ and u^{xm}_{Tm+1} is affected by d^{x1}_{T1} [16]. An *activating path* for df-chain (d^{x1}_{T1}, u^{x1}_{T2}, d^{x2}_{T2}, u^{x2}_{T3}, d^{x3}_{T3} ... d^{xm-1}_{Tm-1}, u^{xm-1}_{Tm}) is a sequence of transitions (T_1, ..., T_2, ..., T_3, ..., T_{m-1}, ..., T_m) in which (T_i, ..., T_{i+1}) is def-clear path with respect to variable x_i, $1 \leq i \leq m$-1.Of particular interest are df-chains that start with input and terminate with output.

An input is a definition of a variable, independent from any other variable. For example, assignment of a value to a variable in an input statement or by an expression made exclusively of constants is an input. In Figure 2.3, d^{b}_{T1}, d^{pin}_{T1} and $d^{attempts}_{T1}$ are considered as inputs.

There are two types of outputs: variable output and constant output. Variable output is a c-use of variable in an output statement, for instance, c^{b}_{T7} in Figure 2.3. On the other hand, constant output is an output statement containing only constants. In case of variable output, we say that a variable output *o* is affected by an input *i* if there is a df-chain starting with *i* and terminating with *o*. However, there is no such df-chain terminating with a constant output; therefore, the effect of input value *i* on a constant output can only be captured by a df-chain starting with *i* and terminating with the last p-use which leads the flow of control to the constant output and then to the exit state, for instance, $p^{attempts}_{T3}$.

Hence, two relations can be defined, namely *input-output relation* and *input-predicate relation*, depending on the type of affect.

X denotes a set of all inputs.

Y denotes a set of all variable outputs.

P denotes a set of last p-uses leading the flow of control to a constant output.

An input-output relation is defined as $R_{XY} = \{(i, o) \mid i \in X, o \in Y$, and o is affected by $i\}$. Also, an input-predicate relation is defined as $R_{XP} = \{(i, p) \mid i \in X, p \in P$, and p is affected by $i\}$. Let $i \in X, o \in Y$ and $p \in P$ then $(i, o) \in R_{XY}$ or $(i, p) \in R_{XP}$ if and only if there is a df-chain starting with i and terminating with o or p, respectively. Such a df-chain is referred to as *IP/O-df-chain* (*Input-Predicate/Output-df-Chain*). This concept is generalized as IPO_n-df-chain with $n \geq 2$. IPO_n-df-chain is an IPO-df-chain such that there are m ($1 \leq m \leq n$) occurrences of $u_T^{@}\ d_T^{\#}$ where @ and # stand for any variable. An activating path for IPO_n-df-chain (d_{T1}^{X1}, u_{T2}^{X1}, d_{T2}^{X2}, u_{T3}^{X2}, d_{T3}^{X3} ... d_{Tm-1}^{Xm-1}, u_{Tm}^{Xm-1}) is a path (T_1, ..., T_2, ..., T_3, ..., T_{m-1}, ..., T_m) in which (T_i, ..., T_{i+1}) is a def-clear path with respect to variable x_i, $1 \leq i \leq m$-1. Accordingly, *IPO_2-df-chains* coverage criteria is satisfied by a set of complete paths Π if

a) an activating path for every IPO_k-df-chain ($1 \leq k \leq 2$) for each $(i, o) \in R_{XY}$ is covered at least once by Π;

b) an activating path for every IPO_1-df-chain for each $(p, o) \in R_{XP}$ is covered at least once by Π.

2.3 Dependency Analysis in the EFSM Model

In an EFSM, there are two types of static dependencies between transitions, namely data dependency and control dependency [1]. Data dependency is defined using the association between definitions and uses of variables. Control dependency, on the other hand, derives from the concept of post-dominance explained below, and exists when the execution of a transition is possible only after the execution of another one.

2.3.1 Data Dependency

Data dependency relies on the notion of definition-use association where one transition defines a value to a variable and the same or another transition uses this value. Formally, there exists a data dependency from transition t_i to transition t_k w.r.t. a variable v if

1. d_{ti}^{v} is the last definition of v in t_i
2. u_{tk}^{v} is a c-use or p-use of v in t_k before v is redefined in t_k (if any)
3. there exists a def-clear path w.r.t. v from t_i to t_k

For example, in Figure 2.3, *def*(T1) = {attempt, b, pin} and *use*(T6) = {b, d}. There is a data dependency from T1 to T6 w.r.t. b due to the fact that $b \in$ *def*(T1), $b \in$ *use*(T6) and there is a def-clear path w.r.t. b from T1 to T6 (namely T1 T4 T6). All data dependencies identified for the EFSM in Figure 2.3 are shown in Table 2.6.

Table 2.6 Data Dependencies Identified in Figure 2.3

From Transition	To Transition	w.r.t. variable	def-clear path
T1	T2	attempts or pin	T1 T2
	T3	pin	T1 T2 T2 T2 T3
	T4	pin	T1 T4
	T5	b	T1 T4 T5
	T6	b	T1 T4 T6
	T7	b	T1 T4 T7
T2	T2	attempts or pin	T2 T2
	T3	attempts	T2 T3
T5	T5	b	T5 T8 T5
	T6	b	T5 T8 T6
	T7	b	T5 T8 T7
T6	T5	b	T6 T8 T5
	T6	b	T6 T8 T6
	T7	b	T6 T8 T7

2.3.2 Control Dependency

The concept of control dependency in an EFSM is extended from the notion of control dependence defined for a program control graph [2]. In an EFSM, a control dependency captures the notion that one transition may potentially affect traversal of another transition and is based on the concept of post-dominance. Let us define the concept of post-dominance first.

Let S_1 and S_2 be states (nodes) and T be a transition outgoing from S_1 in an EFSM. State S_2 *post-dominates* state S_1 if S_2 is on every path from S_1 to exit state. State S_2 *post-dominates* transition T if S_2 is on every path from S_1 to exit state through T. There is a control dependency from transition T_i to transition T_k if:

- state $S_b(T_k)$ does not post-dominate state $S_b(T_i)$ and
- state $S_b(T_k)$ post-dominates transition T_i

where $S_b(T)$ denotes the starting state of T.

It can be observed from Figure 2.3 that S_b(T4) = $S1$, S_b(T6) = $S2$, state $S2$ does not post-dominate state $S1$ but $S2$ post-dominates transition T4. Hence, there is a control dependency from transition T4 to transition T6. All control dependencies identified for the EFSM in Figure 2.3 are shown in Table 2.7.

Table 2.7 Control Dependencies Identified in Figure 2.3

From Transition	To Transition
T4	T5, T6, T7, T9
T5	T8
T6	T8
T7	T8

2.3.3 Static EFSM Dependence Graph (SDG)

Control and data dependencies in an EFSM can be graphically represented as a directed graph where each node represents a transition and each edge represents either a control dependency or data dependency. For example, Figure 2.7 shows the static dependence graph of the EFSM given in Figure 2.3 constructed by using the control and data dependencies between transitions given in Table 2.6 and Table 2.7.

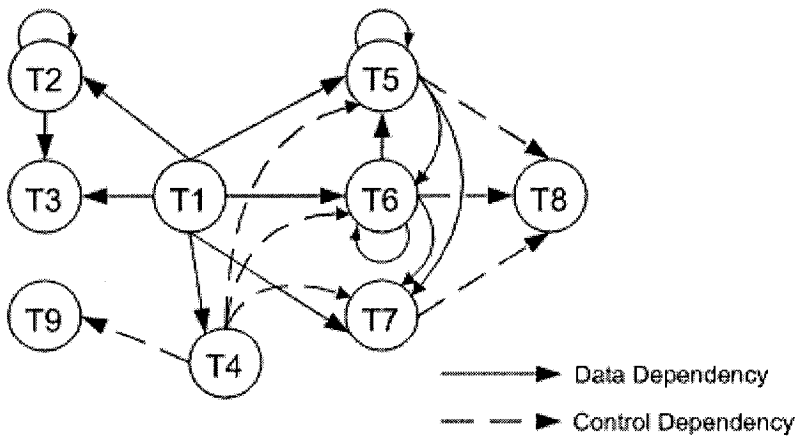


Figure 2.7 Static Dependence Graph of the ATM EFSM

In the next chapter, we present two test suite reduction methods based on two types of dependency analysis (static and dynamic). Also, algorithms in each method for the reduction of requirement-based test suite are presented.

Chapter 3

Control and Data Dependency Analysis for Test Suite Reduction

To test the implementation of a given system, a *test suite* is needed. A test suite is a set of *test cases*, each test case being a sequence of transitions. Several strategies, described in Chapter 2, can be used to generate test cases. However, a tester may be particularly interested in testing a given requirement. For each requirement, there is a transition that represents that requirement, which is called the Transition Under Test (TUT) in this work. For example, for requirement *r*2 (after entering a correct pin in less than four attempts, user selects withdrawal function. The system adjusts the balance, and displays a menu with withdrawal, deposit, balance inquiry, and exit functions.), transition T5 in Figure 2.7 is the representative of *r*2.

This thesis, based on the approach proposed in [1], presents a method to reduce the number of test cases in a given test suite by using EFSM dependency analysis. The motivation behind the approach is that during test execution, different elements of the system interact with each other and some of them may affect the TUT. Capturing only those interactions that influence the TUT yields a pattern of interactions w.r.t. the TUT exercised by the test, which is in turn utilized to identify an equivalent class of test cases having the same interaction pattern w.r.t. the TUT. In particular, two tests are considered being equivalent w.r.t. the TUT if both exhibit the same interaction pattern; hence, one of them can be discarded from the test suite.

To reduce the size of the test suite without significantly reducing its fault detection capability, we simply eliminate all but one of the equivalent test cases from each class of

equivalent test cases of the test suite. Hence, to obtain those classes of equivalent test cases, the problem of defining the interaction pattern exhibited by each test case needs to be addressed.

In this work, this problem is solved based on the assumption that interactions between EFSM transitions ("active" elements of the EFSM) are represented as control or data dependencies. Next, two types of dependency analysis are used: *static dependency analysis* and *dynamic dependency analysis* to capture the pattern of interaction existing for each test case, which is employed later on to formally identify equivalent test cases.

3.1 Test Suite Reduction Based on Static Dependency Analysis

In [1], an approach for test suite reduction based on static dependency analysis has been proposed: for each test case in a test suite (a sequence of transitions), control and data dependencies that are encountered during the traversal of the test sequence are identified and represented as a *dependence sub-graph* (a sub-graph of a static dependence graph (SDG)). The control and data dependencies that affect the transition under test are of particular interest in this approach. Such dependencies can be identified by traversing the dependence sub-graph backwards from the transition under test and marking all traversed nodes and edges. All unmarked nodes and edges are removed from the dependence sub-graph. As a consequence, the resulting dependence sub-graph consists only of control and data dependencies that influence the TUT and it is referred to as a Static Interaction Pattern (SIP). Two test cases are equivalent w.r.t. TUT if both exhibit the same SIP. For example, consider Test_1= T1 T4 T5 T8 T6 T8 T5 T8 T7 T8 T9, and TUT = T5. The dependence sub-graph of Test_1 is presented in Figure 3.1 and the static interaction

pattern of Test_1 is given in Figure 3.2. It is constructed by removing the nodes and edges that are not marked in bold in Figure 3.1(b).

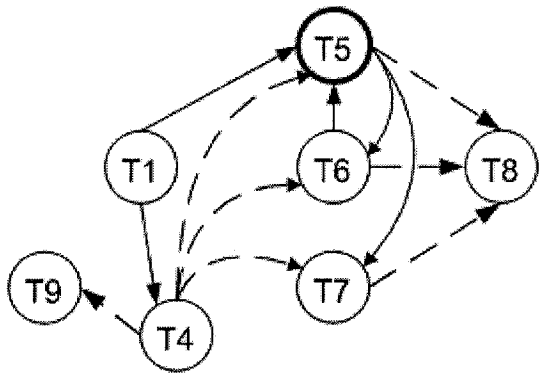


Figure 3.1 (a) Dependence Sub-Graph of Test_1

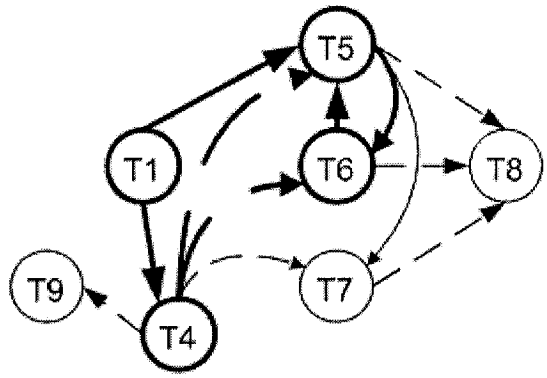


Figure 3.1 (b) Dependence Sub-Graph of Test_1, with transitions affecting T5 marked in bold

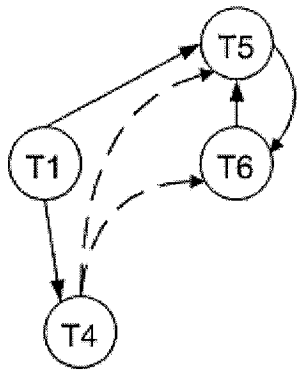


Figure 3.2 The Static Interaction Pattern of Test_1

Let us consider another test case, Test_2 : T1 T2 T4 T5 T8 T6 T8 T5 T8 T7 T8 T9. Test_1 and Test_2 are almost identical except for the extra transition T2 in Test_2. Figure 3.3 shows the dependence sub-graph of Test_2 and Figure 3.4 shows the static interaction pattern of Test_2. Figure 3.2 and 3.4 show that Test_1 and Test_2 are equivalent since

both tests produce the same static interaction pattern. Hence, one of them can be eliminated from the test suite (e.g., using random selection).

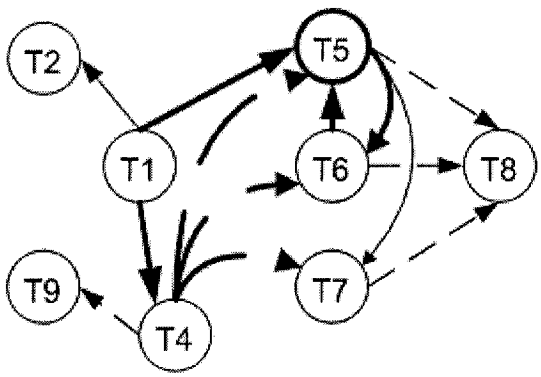


Figure 3.3 Dependence Sub-Graph of Test_2

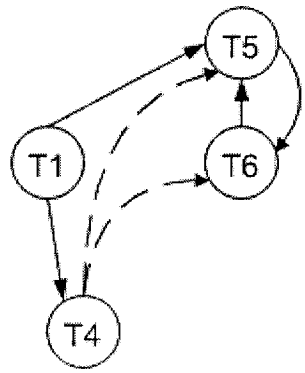


Figure 3.4 Static Interaction Pattern of Test_2

Based on this concept, the problem of test suite reduction arises in terms of selecting a subset of test cases from the test suite, while still satisfying the same test objective, i.e., to cover each requirement with respect to the static interaction patterns for that requirement. Such subset is obtained by selecting randomly one test from each equivalence class of test cases having the same static interaction pattern. To solve the problem, we break down the solution strategy we will take as follows:

Given an EFSM M, its SDG, and a test suite TS, let

SIP_{ts} denote the SIP of a transition sequence ts which is in the test suite to be reduced,

S_r denote a set of all possible SIPs w.r.t. the requirement under test r,

$TS_r = \{ts \mid ts$ is a complete sequence of transitions where r occurs at least once$\}$,

S'_r denote a set of static interaction patterns for r induced by TS_r.

Then,

$\forall r \in R$,

- form $TS_r \subset TS$ where $TS_r = \{ts \mid ts$ is a complete sequence of transitions where r occurs at least once$\}$
- $\forall ts \in TS_r$,
    - form SIP_{ts} from ts using SDG
    - identify SIP_{ts} in S_r, call it SIP_i
    - put ts in $TS_r(SIP_i)$ which is the equivalence class of ts having the same SIP
    - put SIP_i in the set S'_r of static interaction patterns for r induced by TS_r
- identify which $SIP_r \in S_r$ (if any) is not covered by any $ts \in TS_r$ using S_r and S'_r
- randomly select one ts from each $TS_r(SIP_i)$, $1 \leq i \leq |S'_r|$ to form the reduced TS_r

Notice that the number of test cases in the reduced test suite is bounded below by the number of possible static interaction patterns for the given EFSM M.

To implement this solution, three algorithms have been developed in this research. They are presented in the remainder of this Chapter: Section 3.1.1 describes in pseudo-code the algorithm for generating SIP_{ts} for a given ts from SDG, Section 3.1.2 gives the algorithm that compares whether two SIPs are equivalent, and Section 3.1.3 provides the algorithm to identify which $SIP_r \in S_r$ (if any) are not covered by any $ts \in TS_r$ using S_r and S'_r.

3.1.1 Algorithm for generating the SIP_{ts} for a given ts from SDG

In this method, dependency identification w.r.t. a given TUT is done by traversing dependence sub-graph backward from the TUT. Since the transitions placed after the last occurrence of the TUT in a transition sequence do not affect the TUT, there is no need to traverse the whole ts. As a consequence, we propose a SIP generation algorithm that first locates the last transition w.r.t. the TUT in ts (referred to as T_{LTUT}), and then considers

only the sub-sequence of transitions in *ts* from the first transition to T_{LTUT} for the identification of control and data dependencies that affect the TUT. This sub-sequence is used to construct the dependence sub-graph. Details on the SIP generation algorithm are given as follows:

Input: SDG, *tut*, *ts*

Intermediate: a digraph *G*

Output: the Static Interaction Pattern(SIP_{ts}) for *ts*

Steps:

- Locate the last occurrence of *tut* in *ts* and name it T_{LTUT}

/* construct nodes in dependence sub-graph *G**/

- From the first transition to T_{LTUT} in *ts*, first, capture a set of unique transitions, second sort the set in ascending order of transition labels, and third represent each transition in the set as a node in the dependence sub-graph *G*.

/* construct edges in dependence sub-graph *G* */

- Loop from T_{LTUT} in *ts* to the second transition in *ts*
  - take the present transition and name it *t*
  - let n_t denote the node representing *t* in *G*
  - let *l* be a prefix of *ts*, not including *t* in *ts*
  - Loop from the last transition in *l* until the first transition in *l* is reached
    - take the present transition from *l* and name it *pt*
    - let n_{pt} be the node representing transition *pt*
    - if ((there is a control dependency from *pt* to *t* in SDG) and (there is no control dependency edge incoming to n_t in *G*))

put a directed dashed edge in G from n_{pt} to n_t

- if ((there is a data dependency from *pt* to *t* w.r.t. variable *v* in SDG) and (there is no data dependency edge w.r.t. *v* incoming to n_t in G))

  put a directed solid edge in G from n_{pt} to n_t

- end of loop

- end of loop

/* At this point we obtained dependence sub-graph G. Now, we will identify dependencies in G that influence the *tut* in order to generate SIP_{ts}*/

- let n_{tut} be the node representing *tut*
- traverse backward from n_{tut} to the source node of G and during the traversal, mark traversed nodes and edges in G
- remove all unmarked nodes and edges in G
- the resulting dependence sub-graph is SIP_{ts}

Data Structure:

$G = (N, E)$ where

$N = \{n_t \mid n_t$ is a node representing transition t in ts and n_0 denotes the source node of $G\}$ and

$E = \{e \mid e = (n_s, n_d, type), n_s, n_d \in N$ where n_s and n_d are the source node and destination node of e, respectively and *type* is *cd* or *dd* where *cd* and *dd* represent control dependency and data dependency, respectively$\}$

Algorithm:

Let

uniqueT = {*t* | *t* is a transition in *ts*}

node(*i*) denotes a node whose index is equal to *i* in *G*,

$DD(n_{Tk})$ = {(n_{Ti},*v*) | there exists a data dependency edge *e* w.r.t. *v* from n_{Ti} to n_{Tk}}

listDD = {(*n*,*v*) | *n* is a node in SDG and *v* is a variable}

$CD(n_{Tk})$ = {n_{Ti} | there exists a control dependency edge outgoing from n_{Ti} to n_{Tk}}

listCD = {*n* | *n* is a node in SDG }

finish[] denotes an array of integers.

Steps

uniqueT ← ∅, *G* ← (∅,∅)

locate the last occurrence of *tut* in *ts* and name it T_{LTUT}

for each transition *t* in *ts* from the first transition to T_{LTUT} do

if *t* ∉ *uniqueT* then

uniqueT ← *uniqueT* ∪ {*t*}

end for loop

sort the elements of *uniqueT* in ascending order of transitions' labels.

/* Each element of sorted *uniqueT* can be referred by its index e.g., *uniqueT*[0]

denotes the first element of *uniqueT*. */

/* *construct G* */

/* *First, create nodes in G* */

lastIndex ← the index of the last element in *uniqueT*

i = 0

for each $t \in$ *uniqueT* from *uniqueT*[0] to *uniqueT*[*lastIndex*] do

/*if there is no node in N with respect to t, add such a node in N*/

if $n_t \notin N$ then

$N \leftarrow N \cup \{n_t\}$

set the index of node n_t to i

$i = i + 1$

end if

end for loop

/* *Second, create edges in G* */

listDD $\leftarrow \varnothing$, *listCD* $\leftarrow \varnothing$

$k \leftarrow$ the index of T_{LTUT} in *ts*

/* *a transition in ts can be referred to by its index, e.g., ts*[*0*] *indicates the first transition in ts* */

for each *t* in *ts* from *ts*[*k*] to *ts*[1] do

$n_t \leftarrow$ the node representing a transition *t* in *N*

identifyCD = false

listDD $\leftarrow DD(n_t)$

listCD $\leftarrow CD(n_t)$

$i \leftarrow$ the index of *t* in *ts*

for each transition *pt* in *ts* from *ts*[*i*-1] to *ts*[0] do

/* *if a control dependency on* n_t *is not yet identified and there is a control dependency from* n_{pt} *to* n_t *, draw a control dependency edge from node* n_{pt} *to* n_t *in E*/*

if($n_{pt} \in$ *listCD* and *identifyCD* != true) then

$E \leftarrow E \cup \{(n_{pt}, n_t, \text{cd})\}$

identifyCD $\leftarrow$ true

end if

list $\leftarrow \{(n, v) \mid (n, v) \in$ *listDD* and $n = n_{pt}\}$

/ if there is a data dependency from n_{pt} to n_t w.r.t. a non-empty set of variables V, draw a data dependency edge from node n_{pt} to n_t in E */*

if(*list* $\neq \varnothing$) then

$E \leftarrow E \cup \{(n_{pt}, n_t, dd)\}$

/ remove all elements w.r.t. variable $v \in V$ from listDD */*

for each $(n, v) \in$ *list* do

listDD $\leftarrow$ *listDD* $- \{(n', v') \mid (n', v') \in$ *listDD* and $v' = v\}$

end for loop

end if

end for loop

end for loop

/ traverse backward from n_{TUT} to n_0 in G, mark nodes and edges during traversal. After the traversal, remove unmarked nodes and unmarked edges* from *G*/*

/ initialize the value of each element in finish*[] *equal to* 0**/*

l = the index of the last node in N

for i = 0 to l do

finish[i] $\leftarrow$ 0

```
    i ← the index of n_TUT
    call traverse_backward( i, finish[] )
    remove all unmarked nodes and unmarked edges in G
    The resulting graph G is SIP_ts
Procedure traverse_backward(i , finish[]){
/*************************************************************
    finish[i] = 0 designates that node(i) has never been visited,
    finish[i] = -1 designates that node(i) has already been visited,
    finish[i] = 1 designates all paths leading from n_0 to node(i) have been already
                traversed
*************************************************************/
    if(finish[i] != 1) then
        finish[i] = -1; /* to indicate that node(i) has already been visited */
    For each edge e incoming to node(i) do
        s ← the source node of e
        if (e is unmarked) mark e
        if (s is unmarked) mark s
        k ← the index of node s
        if ( (s ≠ n_0) /* s is not the source node in G */
            AND (finish[k] = 0) /* s has never been visited */
            AND (s ≠ node(i)) /* to avoid infinite recursion */)
            traverse_backward(k, finish[])
    end for loop
    finish[i] = 1
```

} // end traverse_backward

3.1.2 Algorithm for comparing two SIPs

Input : a pair of SIPs, i.e., SIP_i and SIP_j

Output : true if SIP_i and SIP_j are equivalent, false otherwise.

Data Structure:

$SIP = (N, E)$ where

$N = \{n_t \mid n_t$ is a node representing transition $t\}$ and

$E = \{e \mid e = (n_s, n_d, type)\ n_s, n_d \in N$ where n_s and n_d are the source node and destination node of e, respectively and $type$ is cd or dd where cd and dd represent control dependency and data dependency, respectively.$\}$

Let

$SIP_i = (N_i, E_i)$

$SIP_j = (N_j, E_j)$

Steps:

for each node $n \in N_i$ do

if $(n \notin N_j)$ then

return false

exit for loop

end if

$E_1 \leftarrow \{e \mid e \in E_i$ and $n_s = n\}$

$E_2 \leftarrow \{e \mid e \in E_j$ and $n_s = n\}$

if ($|E_1| \neq |E_2|$) then

```
            return false
            exit for loop
        end if
        if (E1 - E2 ≠ ∅) then
            return false
            exit for loop
        end if
    end for
    return true
```

3.1.3 Algorithm for identifying $SIP_r \in S_r$ (if any) not covered by any $ts \in TS_r$

Inputs: S_r, S'_r

Output: S_{ucr}

Let

S_{ucr} denote $\{SIP_r \mid SIP_r \in S_r\}$

Steps:

1. sort $SIP_r \in S_r$ in ascending order in number of nodes in SIP_r, node's label and number of incoming edges.
2. sort $SIP_r \in S'_r$ in ascending order in number of nodes in SIP_r, node's label and number of incoming edges
3. $S_{ucr} \leftarrow \varnothing$
4. take the first SIP_r from S'_r and name it SIP'_r

   for each SIP_r in S_r

   if (SIP_r is equivalent to SIP'_r and $SIP'_r \neq$ NULL) then

   take the next SIP_r from S'_r and name it SIP'_r

   else

   $S_{ucr} \leftarrow S_{ucr} \cup \{SIP_r\}$

   end if

   end for loop
5. return S_{ucr}

3.2 Test Suite Reduction Based on Dynamic Dependency Analysis

This method is extended from the previous method (static dependency analysis) and has also been proposed in [1]. The two methods, in a principle, are similar except that repetitions of the same dependency during the traversal of a test sequence are taken into account by the extended method. To do so, each traversed transition is represented as a separate node in the dependence graph, even if the same transition has already been traversed before. The resulting graph is called the Dynamic EFSM Dependence Graph (DDG).

The method proceeds as follows [1]: given a test case, each traversed transition is represented as a node in DDG and each identified control and data dependency is represented as an edge between two nodes (transitions). To capture interactions that affect a TUT, dependencies that influence the TUT are identified by traversing backwards from the TUT and marking all traversed nodes and edges. Then all unmarked nodes and edges are removed from the DDG. The resulting graph is referred to as Dynamic Interaction Pattern (DIP). Two tests are considered being equivalent, if they both exhibit the same DIP. For example, consider Test_2= T1 T2 T4 T5 T8 T6 T8 T5 T8 T7 T8 T9 and TUT=T5. The resulting DDG is shown in Figure 3.5. Then, the DIP is generated (Figure 3.7) by traversing backwards from the last occurrence of the TUT in DDG, marking traversed nodes and edges and eliminating the unmarked nodes and edges.

Consider another test, Test_3= T1 T4 T5 T8 T6 T8 T5 T8 T6 T8 T5 T8 T9, with TUT = T5. The DDG of Test_3 is presented in Figure 3.6 and its DIP in Figure 3.8. If static dependency analysis is applied for TUT=T5, Test_3 is removed from the test suite, since it exhibits the same SIP as Test_2. However, by applying dynamic dependency analysis,

Test_2 and Test_3 are not equivalent since they do not have the same DIP. As a consequence, Test_3 cannot be eliminated from the test suite.

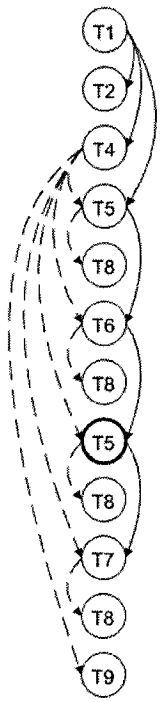


Figure 3.5 DDG of Test_2

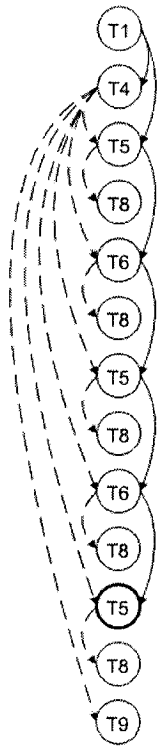


Figure 3.6 DDG of Test_3

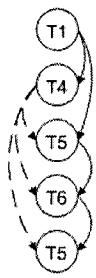


Figure 3.7 DIP of Test_2

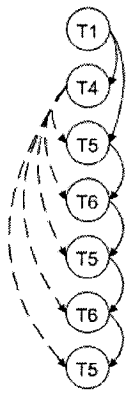


Figure 3.8 DIP of Test_3

Again, the problem of test suite reduction based on this concept is to identify equivalent classes of test cases having the same dynamic interaction patterns. Such equivalent classes are subsequently used to obtain a reduced test suite by randomly selecting one test case from each equivalent class. To solve the problem, we break down the solution strategy we will take as follows:

Given an EFSM M, its SDG and a test suite TS, let DDG_{ts} and DIP_{ts} denote a dynamic dependence graph and a dynamic interaction pattern produced by a transition sequence ts, respectively.

Then,

$\forall r \in R$,

- form $TS_r \subset TS$ where $TS_r = \{ts \mid ts$ is a complete sequence of transitions where r occurs at least once$\}$
- $\forall ts \in TS_r$,
    - form the Dynamic Dependence Graph for ts, i.e., DDG_{ts}, using SDG
    - construct the dynamic interaction pattern DIP_{ts} from DDG_{ts}, call it DIP_i
    - put ts in $TS_r(DIP_i)$ which is the equivalent class of ts having the same DIP
    - put DIP_i in the set D_r of dynamic interaction patterns for r induced by TS_r
- identify which $SIP_r \in S_r$ (if any) not covered by any $ts \in TS_r$ using S_r and D_r
- randomly select one ts from each $TS_r(DIP_i)$, $1 \leq i \leq |D_r|$ to form the reduced TS_r

Three algorithms for dynamic dependency analysis based test suite reduction are proposed in this thesis. Section 3.2.1 describes in pseudo-code the algorithm for

generating DDG_{ts} for a given *ts*, Section 3.2.2 gives the algorithm for generating DIP_{ts} from DDG_{ts}, Section 3.2.3 provides the algorithm for comparing two DIPs.

3.2.1 Algorithm for generating the DDG_{ts} for a given *ts*

Similar to SIP generation, there are no control or data dependencies that can affect the TUT after its last occurrence in the test sequence. Therefore, there is no need to traverse the entire transition sequence to construct its DDG. As a consequence, we propose the following DIP generation algorithm: for a given transition sequence *ts*, first locate the last transition w.r.t. the TUT in *ts* (referred to as T_{LTUT}). Then consider only the sub-sequence of transitions in *ts* from the first transition to T_{LTUT} and identify all control and data dependencies that are encountered to build the DDG.

input : SDG, *tut*, *ts*

output : the *Dynamic Dependence Graph* (DDG_{ts}) for *ts*

Steps:

/* construct DDG_{ts} */

- locate the last occurrence of *tut* in *ts* and name it T_{LTUT}
- take the first transition of *ts* , represent it as the source node in a new graph $G = (N, E)$ and set the index of this node to 0
- move to the next transition of *ts*
- loop from the second transition in *ts* to T_{LTUT}
  - take the present transition from *ts* and name it *t*
  - represent *t* as a new node in *G* and set the index of the node to the index of *t* in *ts* . The node can be referred to as n_t.
  - let *l* be a prefix of *ts*, not including *t* in *ts*

- loop from the last transition in l until the first transition in l is reached
  - take the present transition from l and name it pt
  - let n_{pt} be the node representing transition pt
  - if ((there is a control dependency from pt to t) and (there is no control dependency edge incoming to n_t in G))

    put a directed dashed edge from n_{pt} to n_t
  - if ((there is a data dependency from pt to t w.r.t. variable v in SDG) and (there is no data dependency edge w.r.t. v incoming to n_t in G))

    put a directed solid edge from n_{pt} to n_t
  - end of loop
- end of loop

Note : In DDG_{ts}, a directed solid edge represents data dependency

a directed dashed edge represents control dependency

Data Structure:

$DDG_{ts} = (N, E)$ where

$N = \{n_t \mid n_t$ is a node representing transition t in ts and n_0 denotes the source node of $DDG_{ts}\}$ and

$E = \{e \mid e = (n_s, n_d, type), n_s, n_d \in N$ where n_s and n_d are the source node and destination node of e, respectively and $type$ is cd or dd where cd and dd represent control dependency and data dependency, respectively$\}$

Algorithm:

Let

$DD(n_{Tk}) = \{(n_{Ti}, v) \mid$ there exists a data dependency edge e w.r.t. v from n_{Ti} to $n_{Tk}\}$

listDD = {(*n*,*v*) | *n* is a node in SDG and *v* is a variable}

$CD(n_{Tk})$ = {n_{Ti} | there exists a control dependency edge outgoing from n_{Ti} to n_{Tk}}

listCD = {*n* | *n* is a node in SDG }

Steps:

/* construct DDG_{ts} */

$DDG_{ts} \leftarrow (\varnothing, \varnothing)$

$n_0 \leftarrow$ the node representing the first transition in *ts*

$N \leftarrow N \cup \{n_0\}$

set the index of n_0 to 0

$k \leftarrow$ the index of the last transition in *ts*

For each transition *t* in *ts* from *ts*[1] to *ts*[*k*]

$i \leftarrow$ the index of *t* in *ts*

$n_t \leftarrow$ the node representing transition *t* in *N*

$N \leftarrow N \cup \{n_t\}$

set the index of n_t to *i*

listDD $\leftarrow$ *DD*(*t*)

listCD $\leftarrow$ *CD*(*t*)

identifyCD = false

for each transition *pt* in *ts* from *ts*[*i*-1] to *ts*[0] do

$n_{pt} \leftarrow$ the node representing transition *pt*

if($n_{pt} \in$ *listCD* and *identifyCD* != true) then

$E \leftarrow E \cup \{(n_{pt}, n_t, \text{cd})\}$

identifyCD ← true

end if

list ← $\{(n, v) \mid (n, v) \in listDD \text{ and } n = n_{pt}\}$

if (list ≠ ∅) then

$E \leftarrow E \cup \{(n_{pt}, n_t, dd)\}$

for each $(n, v) \in list$ do

listDD ← *listDD* – $\{(n', v') \mid (n', v') \in listDD \text{ and } v' = v\}$

end for loop

end if

end for loop

end for loop

3.2.2 Algorithm for generating DIP_{ts} from DDG_{ts}

input : DDG_{ts} and TUT t

output : DIP_{ts}

Data Structure:

$DDG_{ts} = (N, E)$ where

$N = \{n_t \mid n_t$ is a node representing transition t in ts and n_0 denotes the source node of $DDG_{ts}\}$ and

$E = \{e \mid e = (n_s, n_d, type), n_s, n_d \in N$ where n_s and n_d are the source node and destination node of e, respectively and *type* is *cd* or *dd* where *cd* and *dd* represent control dependency and data dependency, respectively$\}$

Algorithm:

Let *finish*[] denote an array of integers.

Steps:

For for $i = 0$ to ($|N| - 1$) do

finish[i] ← 0

end for loop

TUTList ← $\{n_t \mid n_t \in N$ and n_t represents TUT $t\}$

For each node $n \in$ *TUTList* from the last node to the first node in *TUTList* do

mark n

i ← the index of n

call traverse_backward(i, *finish*[])

end for loop

remove all unmarked nodes and unmarked edges

The resulting graph DDG_{t_s} is DIP_{t_s}

3.2.3 Algorithm for comparing two DIPs

Input: a pair of DIPs, i.e., DIP_i and DIP_j

Output: true if DIP_i and DIP_j are equivalent, false otherwise

Data Structure:

$DIP = (N, E)$ where

$N = \{n_t \mid n_t$ is a node representing transition $t\}$ and

$E = \{e \mid e = (n_s, n_d, type)\ n_s, n_d \in N$ where n_s and n_d are the source node and destination node of e, respectively and *type* is *cd* or *dd* where *cd* and *dd* represent control dependency and data dependency, respectively }

Let

$DIP_i = (N_i, E_i)$

$DIP_j = (N_j, E_j)$

Steps:

for each node $n \in N_i$ do

if ($n \notin N_j$) then

return false

exit for loop

endif;

$E_1 \leftarrow \{e \mid e \in E_i \text{ and } n_s = n\}$

$E_2 \leftarrow \{e \mid e \in E_j \text{ and } n_s = n\}$

if ($|E_1| \neq |E_2|$) then

return false

exit for loop

endif;

if ($E_1 - E_2 \neq \varnothing$) then

return false

exit for loop

endif;

endfor;

return true

In the next chapter, we present the Test Suite Reduciton program, which has been developed based on algorithms introduced in this chapter.

Chapter 4

TSR Software Tool

4.1 TSR Overview

Based on the algorithms for the reduction of requirement-based test suites using EFSM dependency analysis described in Chapter 3, the *Test Suite Reduction* (TSR) program has been developed as a part of the Test Suite Generation/Reduction Software tool (TSGR) [20], which is implemented in C++ and Java languages and runs on Sun workstations under Solaris Sparc 5.8. Besides the Test Suite Reduction program TSR, there exists another part of TSGR called the *Reduced Test Suite Generation* (TSG) program, whose function is to generate a small yet effective set of executable test cases such that it exercises all possible SIPs w.r.t. each requirement under test at least once.

TSR is built to reduce the number of test cases in a given test suite by eliminating equivalent test cases, according to the definition of either SIP or DIP. TSR is made up of two sub-programs depending on whether SIPs or DIPs are utilized, namely, STSR program and DTSR program. STSR stands for SIP-based Test Suite Reduction, which reduces the number of test cases according to the static interaction patterns. As for DTSR, it stands for DIP-based Test Suite Reduction, which reduces the number of test cases by employing dynamic interaction patterns. Figure 4.1 and Figure 4.2 show the structure of the STSR and DTSR program, respectively.

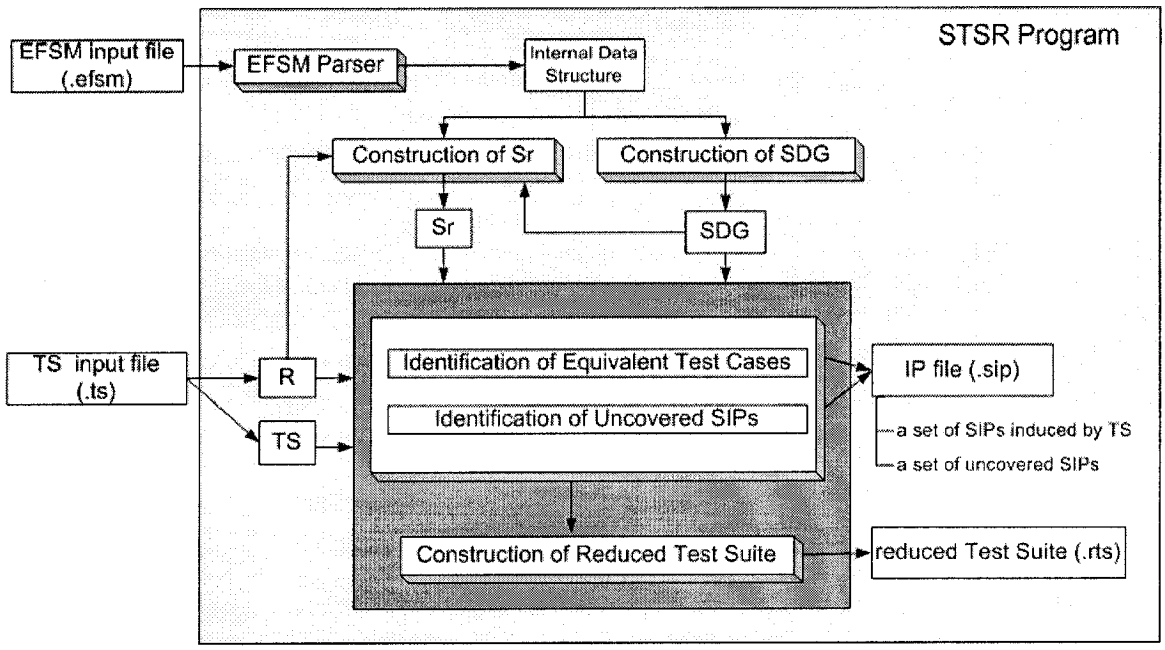


Figure 4.1 Structure of STSR

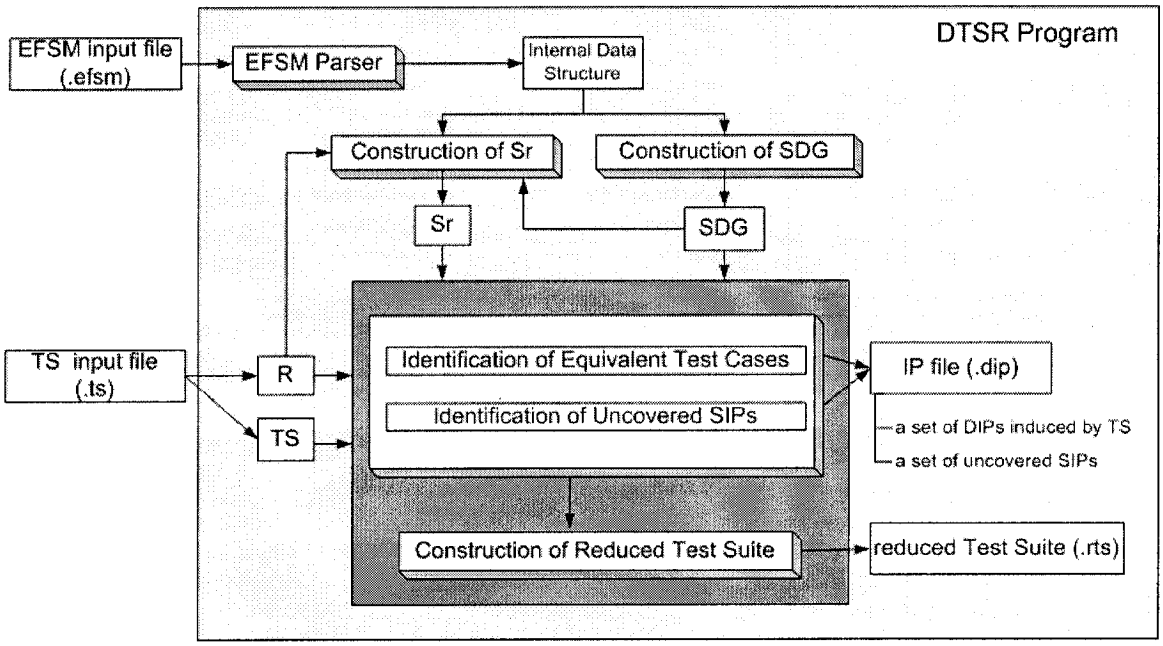


Figure 4.2 Structure of DTSR

Basically, STSR and DTSR have the same architecture and performs the following tasks:

- Phase 1: Construction of SDG;
- Phase 2: Identification of Equivalent Test Cases and Uncovered SIPs;
- Phase 3: Construction of Reduced Test Suite;

It must be noted that Phase 1 was implemented by ASERT Lab members: Tuong Nguyen and Yan Gao where the EFSM parser developed by Tuong yields internal data structures which are subsequently used to build an SDG. SDG construction was developed by Gao based on the algorithm for generating the SDG from an EFSM in [37].

The implementations of these three phases are discussed in detail in Sections 4.4 and 4.5, respectively for STSR and DTSR. Section 4.2 gives the input file formats of STSR and DTSR, while Section 4.3 gives the output file formats.

4.2 Input File Formats

As shown in figures 4.1 and 4.2, STSR and DTSR require two input files: EFSM input file and TS input file. The EFSM input file is a file representing an EFSM. In this thesis, such EFSM files are given the ".efsm" extensions, and are defined formally below using the Backus-Naur Form (BNF):

Table 4.1 BNF definition of an EFSM input file

<efsm> ::=
efsmId
numStates startStateIndex exitStateIndex
<transitions>
<transitions> ::=

```
        <transition> | <transitions> <transition>

<transition> ::=

        transition transitionId

        sourceStateIndex destinationStateIndex

        <requirement>

<requirement> ::=

        [<input>]

        [<enablingPredicate>]

        /

        [<actions>]

<actions> ::=

        <action> | <actions> <action>

<action> ::=

        <output> | <assignment> | <set> | <reset> | <procedureCall>

<input> ::=

        inputId ( [<parameters>] )

<output> ::=

        outputId ( [<parameters>] )

<enablingPredicate> ::=

        <variableIds> [/* booleanExpression */]

<assignment> ::=

        <variableId> := <expression>

<set> ::=
```

```
        set ( constant , timerId )
<reset> ::=
        reset ( timerId )
procedureCall ::=
        procedure ( procedureId ( <variableIds> [; <variableIds>]) ) { <pbrDefs> }
<parameters> ::=
        <parameter> {, <parameter>}*
<parameter> ::=
        <variableId> | constant
<variableIds> ::=
        <variableId> {, <variableId>}*
<pbrDefs> ::=
        <pbrDef> | <pbrDefs> <pbrDef>
<pbrDef> ::=
        <variableId> := <expression> ;
<expression> ::=
        function ( <variableIds> ) | constant
<variableId> ::= id
```

An example ".efsm" file for the EFSM of the ATM system in Figure 2.3 is given in Appendix A.7.

Another input file, TS input file, is a file representing a test suite. A TS file consists of a set of requirements where each requirement is represented by a transition under test (tut) and a collection of test cases. A test case is a complete sequence of transitions that

starts at the entry node and ends at the exit node of the EFSM. TS input file have been given the ".ts" extension. The detailed definition of TS files in BNF is presented below:

Table 4.2 BNF definition of a TS input file

```
<ts> ::=
        efsmId <tuts> <tests>
<tuts> ::=
        <tut> | <tuts> <tut>
<tut> ::=
        transitionId
<tests> ::=
        <test> | <tests> <test>
<test> ::=
        test testId <transitionSeq>
<transitionSeq> ::=
        transitionId | <transitionSeq> transitionId
```

An example TS file for the EFSM of the ATM system presented in Figure 2.3 is given in Appendix A.8.

4.3 Output File Formats

STSR and DTSR each generates two output files: a reduced TS file and an IP file. The reduced TS file is a file representing the reduced test suite where redundant test cases have been eliminated. A reduced TS file is given the ".rts" extension and shares the

format of original TS file. The detailed definition of reduced TS files in BNF is presented below:

Table 4.3 BNF definition of a reduced TS input file

```
<ts> ::=
        efsmId <tests>
<tests> ::=
        <test> | <tests> <test>
<test> ::=
        test testId <transitionSeq>
<transitionSeq> ::=
        transitionId | <transitionSeq> transitionId
```

Another output file, IP, represents, for each TUT, two sets of interaction patterns. The first set consists of unique interaction patterns induced by test cases in a TS input file, which are either static or dynamic interaction patterns w.r.t. the TUT. Each interaction pattern comes with a group of equivalent test cases that exhibit it. Each test case is referred to in the file by its test id. The second set consists of static interaction patterns w.r.t. the TUT that are not covered by any test case in the original TS file. The extension of the file depends on the program used. STSR generates a ".sip" file, while DTSR generates a ".dip" file. The IP file is defined using the Backus-Naur Form (BNF) below:

Table 4.4 BNF definition of an IP output file

```
<sip/dip> ::=
        efsmId <tut> <ips>
```

<tut> ::=

transitionId

<ips> ::=

<ip> | <ips> <ip>

<ip> ::=

ip ipId [<testIds>] <nodes>

<testIds> ::=

testId | <testIds> testId

<nodes> ::=

<node> | <nodes> <node>

<node> ::=

node nodeIndex <label> [<adjacencySet>]

<label> ::=

transitionId

<adjacencySet> ::=

<reverseSet> | <nonreverseSet>

<reverseSet> ::=

<reverse> | <reverseSet> <reverse>

<reverse> ::=

inc sourceIndex <dependencyType>

<nonreverseSet> ::=

<nonreverse> | <nonreverseSet> <nonreverse>

<nonreverse> ::=

```
        out destinationIndex <dependencyType>

<dependencyTye> ::=

        dat | ctl
```

It is noted that an IP output file distinguishes a type of interaction pattern according to the prefix of ipID i.e., "S", "D" and "U" denote static, dynamic and uncovered static interaction pattern, respectively. Examples of ".sip" files for the EFSM of the ATM system given in Figure 2.3 can be found in Appendix A.9.

4.4 STSR Program

STSR uses static dependency analysis to reduce the size of test suites without significantly reducing their fault-detection capability. It eliminates repetitive test cases, i.e. all test cases except one that exhibit the same static interaction pattern. As mentioned earlier, STSR can be broken down into three phases.

Phase 1: Construction of SDG & S_r

Given the two input files, STSR checks that the test suite belongs to the EFSM system (e.g., by checking if efsm ids in EFSM and TS files are identical). Then, the EFSM file is analyzed by lexical and parser utilities of the Unix environment, Lex and Yacc, to get the EFSM information, classify the variables and their occurrences in each transition, and form the EFSM internal data structure. Lex generates lexical analyzers. The code generated by Lex is used to read input characters and produce a sequence of tokens that a parser can use for syntax analysis. Yacc generates parsers that get the tokens and fill the internal data structure of the program according to the given grammar. Based on this internal data structure: control and data dependencies are identified and captured

to build the SDG. For example, from the EFSM input file given in Appendix A.7, the fragment of internal data structure of the SDG representing the fact that there is a data dependency from transition T1 to transition T2 is represented here:

Table 4.5 An Example of Data Dependency Representation

EFSM id: ATM_System
Static Dependency Graph (SDG):
Source node index: 0, Label: T1
Destination node index: 1, Label: T2
Number of edges: 3
List of edges: DType(Data=0,Control=1,CUse=2,PUse=3)
(Variable, DType, OOrder of def, OOrder of use)
(pin,0,2,3)
(attempts,0,3,2)
(attempts,0,3,5)

To generate the SDG from a given EFSM efficiently, the algorithm proposed in [37] is used. The complexiy of SDG generation is quadratic in S which is a set of states in an EFSM [37].

In this phase, besides the construction of SDG, a set S_r of all possible SIPs w.r.t. each requirement under test r is constructed. In order to construct S_r, the information from SDG, the EFSM internal data structure and a set of requirements are needed. At this point, only the set of requirements has not yet been obtained; hence, a TS input file is now required. As stated previously, a TS input file consists of a set of transitions under test and a collection of test cases where each test case is a sequence of transitions starting at the entry node and ending at the exit node of the corresponding EFSM. At this moment, those sets in the TS input file are extracted to form R and TS which stand for a set of requirements under test and a set of test cases, respectively. The algorithm to generate S_r proposed in [37] is used to obtain S_r in this phase.

Phase 2: Identification of Equivalent Test Cases and Uncovered SIPs

In this research, we assume that an individual requirement can be mapped into one transition in an EFSM so that *requirement under test* r and *transition under test* TUT can be referred interchangeably.

In this phase, an IP output file, ".sip" is constructed from R and TS (obtained from the previous phase). As mentioned earlier, ".sip" is composed of, for each requirement under test r, two sets of interaction patterns: a set S'_r of SIPs w.r.t. r induced by test cases in TS and a set of SIPs that are not covered by any test case in TS. The motivation to report uncovered SIPs is based upon the fact that some static patterns of interactions of the system under test may remain uncovered by the TS input file. As a consequence, the reduced test suite obtained in Phase 3 may not be capable of detecting some faults w.r.t. a TUT. Hence, STSR reports a set of uncovered SIPs w.r.t the TUT in ".sip".

In order to generate a ".sip" file, STSR, for each $r \in R$, constructs a set of SIPs w.r.t. r induced by test cases in $TS_r \subset TS$ and a set of uncovered SIPs w.r.t. r as follows:

- Step 1: STSR identifies a set of equivalence classes of transition sequences having the same SIP w.r.t. the TUT, which is subsequently used in Phase 3 to construct a reduced TS file.
- Step 2: STSR identifies and reports SIPs w.r.t. TUT that are not covered by any test case in TS.

Details of the execution of these two steps are given below:

For each transition under test TUT,

/ Step1*/*

1. STSR forms a set of test cases (TS_r) w.r.t. TUT by extracting those test cases from the TS file in which the TUT occurs at least once.
2. STSR identifies sets of equivalent test cases in TS_r by investigating transition sequences that exhibit the same static interaction pattern (SIP). Thus, let $TS_r(SIP_i)$ be the equivalence class of transition sequences having the same SIP_i. For each transition sequence $ts \in TS_r$,
   - STSR forms SIP_{ts} from ts using SDG obtained in Phase 1. The algorithm for generating the SIP_{ts} from SDG is presented in Section 3.2.1.
   - STSR puts ts in $TS_r(SIP_i)$ where SIP_i is equivalent to SIP_{ts}. The algorithm for comparing two SIPs is presented in Section 3.2.2.
   - STSR puts SIP_i in the set S'_r of static interaction patterns for r induced by TS_r.

3. STSR exports all $SIP_r \in S'_r$ (where r corresponds to the TUT) as well as a group of test cases (referred to by test id) from $TS_r(SIP_r)$ into ".sip".

/ Step 2 */*

4. STSR inputs a set of all possible SIPs w.r.t. the TUT, S_r.
5. STSR identifies which $SIP_r \in S_r$ (if any) is not covered by any $ts \in TS_r$ using S_r and S'_r (obtained in Step 1). The algorithm to identify uncovered SIP_r is presented in Section 3.2.3.
6. STSR reports $S_r - S'_r$ by appending such information to ".sip".

Phase 3: Construction of Reduced Test Suite

In the previous phase, repetitive test cases have already been identified in terms of equivalent test cases that exhibit the same SIP w.r.t. the TUT. Therefore, reduced test suite can be constructed by simply selecting randomly one test case of each equivalence class of transition sequences having the same SIP w.r.t. the TUT. More formally,

let RTS = {ts | ts is a complete sequence of transitions};

RTS $\leftarrow \varnothing$

for each $r \in R$ in the TS file

$RTS_r \leftarrow \varnothing$

for $i = 1$ to $|S'_r|$ do

select randomly one ts from $TS_r(SIP_i)$ and $RTS_r \leftarrow RTS_r \cup \{ts\}$

end for

$RTS \leftarrow RTS \cup RTS_r$

end for

At this point, STSR has formed the reduced test suite and exported the contents of the TS file to ".rts" file (including efsm id and set of transitions under test) where the repetitive test cases have been eliminated.

4.5 DTSR Program

DTSR uses dynamic dependency analysis to reduce the size of a given test suite, without significantly reducing its fault-detection capability. DTSR implementation is largely similar to that of STSR, except for some specific details in phases 2 and 3. Those differences are described here.

Phase 2: Identification of Equivalent Test Cases and Uncovered SIPs

Similar to STSR, DTSR extracts *R* and TS from the TS input file in Phase 1. Again in this phase, an IP output file is constructed from *R* and TS; however, the extension of the constructed IP file is ".dip" as opposed to ".sip" in STSR.

The difference between STSR and DTSR in this phase stems from the fact that in DTSR, dynamic interaction patterns are used to define equivalence classes of transition sequences. Therefore, ".dip" is composed of, for each requirement under test *r*, two sets of interaction patterns: a set D_r of DIPs w.r.t. *r* induced by test cases in TS and a set of SIPs that are not covered by any test case in TS. The motivation of DTSR to report uncovered SIPs is based upon two facts: first, the objective of testing (as mentioned in Chapter 3) is to cover different patterns of interactions w.r.t. each TUT and second, the number of SIPs w.r.t. each TUT is bounded. Therefore, a complete test suite should at least cover all static interaction patterns w.r.t. each TUT. Thus, DTSR reports uncovered SIPs w.r.t. each TUT in ".dip".

DTSR generates ".dip" using the same principle as is used in STSR to generate ".sip". That is, DTSR, for each $r \in R$, constructs a set of DIPs w.r.t. r induced by test cases in $TS_r \subset TS$ and a set of uncovered SIPs w.r.t. r as follows:

- Step 1: DTSR identifies a set of equivalence classes of transition sequences having the same DIP w.r.t. the TUT, which is subsequently used in Phase 3 to construct a reduced TS file.
- Step 2: DTSR identifies and reports SIPs w.r.t. TUT that are not covered by any test case in TS.

Details of the execution of these two steps are given below:

For each transition under test TUT

/ Step 1 */*

1. DTSR forms a set of test cases (TS_r) w.r.t. TUT by extracting those test cases from TS in which the TUT occurs at least once.
2. DTSR identifies sets of equivalent test cases in TS_r by investigating transition sequences that exhibit the same dynamic interaction pattern (DIP). Thus, let $TS_r(DIP_i)$ be the equivalence class of transition sequences having the same DIP_i.

   For each transition sequence $ts \in TS_r$,

   - DTSR forms DDG_{ts} from ts using the algorithm for generating the DDG_{ts} for a given ts which is presented in Section 3.3.1
   - DTSR generates DIP_{ts} from DDG_{ts} using the algorithm for generating DIP_{ts} from DDG_{ts} which is presented in Section 3.3.2.

- DTSR puts *ts* in $TS_r(DIP_i)$ where DIP_i is equivalent to DIP_{ts}. The algorithm for comparing two DIPs is presented in Section 3.3.3.
- DTSR puts DIP_i in the set D_r of dynamic interaction patterns for *r* induced by TS_r.

3. DTSR exports all $DIP_r \in D_r$ (where *r* corresponds to the TUT) and, associated with it, a group of test cases (referred to by test id) from $TS_r(DIP_r)$ into ".dip".

/ Step 2 */*

4. DTSR inputs a set of all possible SIPs w.r.t. TUT, S_r.
5. DTSR identifies which $SIP_r \in S_r$ (if any) are not covered by any $ts \in TS_r$ using S_r and D_r (obtained in Step1). This is based on the fact that for each transition sequence *ts*, DTSR maps DIP_{ts} into a SIP_{ts} by collapsing group of nodes representing the same transition into a single node.
6. DTSR reports S_r - D_r by appending such information to ".dip".

Phase 3: Construction of Reduced Test Suite

Just like STSR, DTSR randomly selects one test case from each equivalence class of transition sequences having the same DIP w.r.t. the TUT. More formally,

let RTS = {ts | ts is a complete sequence of transitions},

RTS $\leftarrow \varnothing$

for each $r \in R$ in the TS file

$RTS_r \leftarrow \varnothing$

for $i = 1$ to $|D_r|$ do

select randomly one *ts* from $TS_r(DIP_i)$ and $RTS_r \leftarrow RTS_r \cup \{ts\}$

end for

$RTS \leftarrow RTS \cup RTS_r$

end for

DTSR thus generates the reduced test suite and places it in the output ".rts" file. This file shares the format of the input TS file, except for repetitive test cases which have been removed.

In the next chapter, we evaluate our implementation through four case studies. The results of these experiments are also presented and analyzed.

Chapter 5

Case Studies

The approach proposed in this research has been applied to four systems: ATM1, ATM2, Vending Machine and Cruise Control. The requirements of each system are described in English in appendixes A.1, B.1, C.1, and D.1, respectively. For each system, the following experiments have been carried out:

1. Formal representation of the requirements in an EFSM model;
2. Manual generation of test suites according to branch coverage, all-uses coverage and IPO_2-df-Chains coverage criteria.
3. Reduction of the test suites constructed in step 2 using STSR and DTSR.

Section 5.1 describes the test coverage criteria that have been employed to generate test suites. The subsequent sections give the results of test suite reduction for each system and assessment of our approach.

5.1 Test Coverage Criteria Applied in the Case Studies

To generate test suites, we used both control-flow and data-flow oriented test coverage criteria which test both control and data flow aspects of the system implementation. Three test suites are manually derived from one control-flow oriented test coverage criterion (branch coverage) and two data-flow oriented test coverage criteria (all-uses coverage and IPO_2-df-Chains coverage). These three test coverage criteria are presented in Chapter 2. Test cases in such test suites are guaranteed to be executable since we considered "applicable" versions of these three criteria [11], i.e., those that cover only executable branches, du-pairs and IPO_2-df-chains, respectively.

5.2 ATM1 System

The requirements of the ATM1 System (given in Appendix A.1) are formally represented as an EFSM model (Appendix A.2 and Figure 2.3), from which the test suites are derived (Appendix A.4). The results of test suite reduction are shown in Table 5.1 and Table 5.2, respectively for STSR and DTSR.

Let

TS denote the number of test cases in a test suite,

RTS denote the number of test cases in a reduced test suite,

% denote the percentage of reduction and

$S_r - S'_r$ and $S_r - D_r$ denote the number of uncovered SIPs w.r.t. r (if any of the TS does not cover all SIPs for an r)

Table 5.1 Test Suite Size Reduction of ATM1 using Static Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$
T3	1	1	0%	0	1	1	0%	0	1	1	0%	0
T5	1	1	0%	13	3	3	0%	11	58	14	76%	0
T6	1	1	0%	13	2	2	0%	12	58	14	76%	0
T7	1	1	0%	28	2	2	0%	27	61	29	52%	0
T9	1	1	0%	0	3	1	67%	0	61	1	98%	0

Table 5.2 Test Suite Size Reduction of ATM1 using Dynamic Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$
T3	1	1	0%	0	1	1	0%	0	1	1	0%	0
T5	1	1	0%	13	3	3	0%	11	58	30	48%	0
T6	1	1	0%	13	2	2	0%	12	58	30	48%	0
T7	1	1	0%	28	2	2	0%	27	61	61	0%	0
T9	1	1	0%	0	3	1	67%	0	61	1	98%	0

5.3 ATM2 System

The requirements of the ATM2 System (given in Appendix B.1) are first formally described in SDL. The objective of this experiment is to also demonstrate that SDL specifications may be considered by our method, by converting the SDL specification into an EFSM model. The EFSM model is given in Appendix B.3 and the derived test suites are given in Appendix B.4. Tables 5.3 and 5.4 give the results of the test suite reduction, after STSR and DTSR have been used, respectively.

Table 5.3 Test Suite Size Reduction of ATM2 using Static Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$
T6	1	1	0%	0	1	1	0%	0	0	0	0%	1
T8	1	1	0%	0	1	1	0%	0	0	0	0%	1
T9	1	1	0%	0	6	1	83%	0	9	1	89%	0
T10	1	1	0%	4	1	1	0%	4	3	3	0%	2
T14	1	1	0%	4	2	2	0%	3	3	3	0%	2
T15	1	1	0%	1	4	2	50%	0	7	2	71%	0
T16	1	1	0%	4	3	2	33%	3	3	3	0%	2
T20	1	1	0%	0	1	1	0%	0	1	1	0%	0

Table 5.4 Test Suite Size Reduction of ATM2 using Dynamic Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$
T6	1	1	0%	0	1	1	0%	0	0	0	0%	1
T8	1	1	0%	0	1	1	0%	0	0	0	0%	1
T9	1	1	0%	0	6	1	83%	0	9	1	89%	0
T10	1	1	0%	4	1	1	0%	4	3	3	0%	2
T14	1	1	0%	4	2	2	0%	3	3	3	0%	2
T15	1	1	0%	1	4	2	50%	0	7	3	57%	0
T16	1	1	0%	4	3	2	33%	3	3	3	0%	2
T20	1	1	0%	0	1	1	0%	0	1	1	0%	0

5.4 Vending Machine

The requirements of the Vending Machine System (given in Appendix C.1) are formally represented as an EFSM model (Appendix C.2). Test suites derived from the

EFSM model are given in Appendix C.3. The following tables show the reduction obtained by running STSR and DTSR, respectively.

Table 5.5 Test Suite Size Reduction of Vending Machine System using Static Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$
T2	1	1	0%	59	2	2	0%	58	3	3	0%	57
T5	2	1	50%	79	6	3	50%	77	26	6	73%	74
T8	2	1	50%	49	5	3	40%	47	25	6	76%	44
T11	1	1	0%	55	1	1	0%	55	16	6	63%	50
T12	1	1	0%	47	5	3	40%	45	23	6	74%	42

Table 5.6 Test Suite Size Reduction of Vending Machine System using Dynamic Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$
T2	1	1	0%	59	2	2	0%	58	3	3	0%	57
T5	2	1	50%	79	6	3	50%	77	26	13	50%	74
T8	2	1	50%	49	5	3	40%	47	25	21	16%	44
T11	1	1	0%	55	1	1	0%	55	16	16	0%	50
T12	1	1	0%	47	5	3	40%	45	23	17	26%	42

5.5 Cruise Control System

The requirements of the Cruise Control System (given in Appendix D.1) are formally represented as an EFSM model (Appendix D.2). Three test suites are generated (Appendix D.3), and reduced by STSR (Table 5.7) and DTSR (Table 5.8).

Table 5.7 Test Suite Size Reduction of Cruise Control System using Static Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - S'_r$
T2	2	1	50%	N/A	5	1	80%	N/A	73	1	99%	N/A
T3	1	1	0%	N/A	2	2	0%	N/A	0	0	0%	N/A
T4	1	1	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T5	1	1	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T6	1	1	0%	N/A	5	5	0%	N/A	63	9	86%	N/A
T7	1	1	0%	N/A	5	5	0%	N/A	63	9	86%	N/A
T8	1	1	0%	N/A	1	1	0%	N/A	7	1	86%	N/A
T9	1	1	0%	N/A	2	1	50%	N/A	6	1	83%	N/A
T10	1	1	0%	N/A	2	1	50%	N/A	6	1	83%	N/A

T11	2	2	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T13	1	1	0%	N/A	3	3	0%	N/A	73	20	73%	N/A

Table 5.8 Test Suite Size Reduction of Cruise Control System using Dynamic Dependencies

TUT	Branch Coverage				All-Uses Coverage				IPO_2-df-Chains Coverage			
	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$	$\#TS_r$	$\#RTS_r$	%	$S_r - D_r$
T2	2	1	50%	N/A	5	1	80%	N/A	73	1	99%	N/A
T3	1	1	0%	N/A	2	2	0%	N/A	0	0	0%	N/A
T4	1	1	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T5	1	1	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T6	1	1	0%	N/A	5	5	0%	N/A	63	9	86%	N/A
T7	1	1	0%	N/A	5	5	0%	N/A	63	9	86%	N/A
T8	1	1	0%	N/A	1	1	0%	N/A	7	1	86%	N/A
T9	1	1	0%	N/A	2	1	50%	N/A	6	1	83%	N/A
T10	1	1	0%	N/A	2	1	50%	N/A	6	1	83%	N/A
T11	2	2	0%	N/A	2	2	0%	N/A	19	19	0%	N/A
T13	1	1	0%	N/A	3	3	0%	N/A	73	20	73%	N/A

Note: Due to the delay of S_r generation program, S_r was manually derived; however, S_r in Cruise Control System is too many to obtain manually.

5.6 Evaluation of the Proposed Approach

According to the results of the preceding sections, STSR and DTSR do not considerably reduce the size of the test suites derived from branch coverage. This is caused by the very small size of such test suites. For example, in ATM1 system, there is only one test case for each TUT; therefore, the percentage of reduction w.r.t. each TUT is equal to 0%. However, when the size of the test suites becomes larger as obtained by another test suite generation strategy (especially IPO_2-df-Chains coverage), our approach is successful in its aim of reducing the test suites. For example, for the cruise control system, significant reduction (73 to 99%) is achieved.

Our approach also validates the coverage of the reduced test suite by evaluating the adequacy of the reduced test suite based on the coverage of SIPs. To do so, our approach examines a set of SIPs that are not covered by any test case in the original test suite and

reports such uncovered SIPs. For example, in Vending Machine System, there are 57, 74, 44, 50 and 42 missing SIPs w.r.t. TUT T2, T5, T8, T11 and T12, respectively. As such, testers can consequently construct test cases to cover such uncovered SIPs; therefore, as a consequence, our approach may enhance the capability of the reduced test suite in its interaction coverage.

Our initial experience has shown that when applying test suite reduction based on static dependency analysis, the size of the reduced test suite is bounded below by the number of possible static interaction patterns w.r.t. each TUT. For example, in ATM1 system, there are at most 14 SIPs w.r.t. T5 (as shown in Appendix A.6). Therefore, the minimum size of the reduced test suite w.r.t. T5 is equal to 14. This is the result obtained in Table 5.1. In other words, the number of possible SIPs designates the minimum size of the reduced test suite regardless of the test strategy used to generate it. On the other hand, in reduction based on dynamic dependency analysis, the number of test cases in the reduced test suite is based on three factors: the size of the original test suite, the test strategy applied to generate the test suite and the length of tests (length of transition sequences).

In fact, by using dynamic dependency analysis, we obtain more sophisticated interaction patterns as compared to those obtained by static dependency analysis. This is caused by disregarding the repetitions of the same dependency between transitions in static dependency analysis. For example, as we can observe from Table 5.6 and 5.7 in Vending Machine System, the size of reduced test suite obtained by applying dynamic dependency analysis is larger than those obtained by applying static dependency analysis. As the cost of executing the smaller size of reduced test suite and analyzing their results

is less expensive, it is advised to apply static dependency analysis during early stages of testing to detect some faults, and then dynamic dependency analysis in a later phase of testing to exercise more sophisticated interaction patterns.

After performing the experiment, it is observed that the TSR program provides ease of use and substantial degree of automation for the reduction of requirement-based test suite.

In the next chapter, we present our conclusions, with a summary of contributions and directions for future research.

Chapter 6

Conclusion

6.1 Final Remarks

A large test suite can be reduced, while still maintaining most of its interaction coverage capability. This is done by eliminating all but one of the equivalent test cases from each class of equivalent test cases of the test suite. The stress of this thesis is on requirement-based test suite reduction. In the thesis, we assume that requirements can be represented as a single EFSM and each requirement can be adequately represented by a single transition. Classes of equivalent test cases are defined by using EFSM dependency analysis. Two types of dependencies are identified in an EFSM: control dependency and data dependency. Their analysis yields patterns of interaction that affect a requirement under test. The patterns of interaction are in turn used to identify equivalent test cases w.r.t. the requirement under test. In particular, two tests are considered equivalent w.r.t. the requirement under test if both exhibit the same interaction pattern; hence, one of them can be discarded from the test suite.

There are two methods of requirement-based test suite reduction according to the type of dependency analysis applied. Static dependency analysis yields static interaction patterns (SIP) whereas dynamic dependency analysis yields dynamic interaction patterns (DIP). The difference between the static dependency analysis and dynamic dependency analysis lies in the way the interaction pattern is constructed: the former ignores the repetition of the same dependency between transitions. This leads to two interesting observations:

- When static dependency analysis is applied, the minimum size of the reduced test suite is bounded by the number of possible SIPs, regardless of the test strategy used to generate the original test suite. Conversely, by applying dynamic dependency analysis, the minimum size of the reduced test suite may not be bounded a priori and therefore depends on three factors: the size of the original test suite, the test strategy used to generate the original test suite and the length of the test cases in the original test suite.
- Since dynamic dependency analysis takes into account the repetition of the same dependency between transitions, we obtain more sophisticated interaction patterns as compared to those obtained by static dependency analysis. Therefore, it is appropriate to apply static dependency analysis during early stages of testing, and then dynamic dependency analysis in a later phase to exercise more sophisticated interaction patterns.

A reduced test suite should actually cover at least all static interaction patterns w.r.t. each requirement under test; otherwise, it may not be capable of detecting some faults existing in those missing patterns of interaction. This is why our approach reports a set of SIPs (w.r.t. requirement under test) that are not covered by any test case in the original and thus the reduced test suite.

6.2 Summary of Contributions

Below we list the major contributions of the thesis:

We placed the approach in [1] in a formal setting and extended the applicability of requirement-based test suite reduction to a more complete definition of EFSM and thus to a larger subset of SDL than the ones considered in [1].

We have proposed algorithms to generate interaction patterns of a test case w.r.t. a requirement under test. Such interaction patterns are either static or dynamic. To compare interaction patterns and determine whether they are equivalent or not, we have developed and tested two algorithms: an algorithm for comparing two SIPs and an algorithm for comparing two DIPs. An algorithm to investigate a set of SIPs (w.r.t. the requirement under test) that are not covered by any test case from a given test suite has also been proposed, and implemented.

We have developed a test suite reduction program which consists of two sub-programs: STSR and DTSR, to be used depending on which dependency analysis is applied.

We have shown that the approach of test suite reduction using EFSM dependency analysis is successful in its aim to reduce the size of test suites as observed by the results after applying STSR and DTSR to four case studies.

6.3 Directions for Future Research

It would be interesting to see this work improved and/or extended in the following directions:

- We assumed that each requirement can uniquely be represented by a single transition in the given EFSM. It would be interesting to expand the approach presented in this thesis to those EFSMs where some requirements may uniquely be represented by more than one transition.

- So far, we randomly select one test case from a class of equivalent test cases. A strategy to select a representative test case for each equivalence class should be

developed. This would require a fault model which considers the effects of the input parameter values on revealing faults.

- After the reduced test suite is obtained, we report a set of SIPs (w.r.t. the requirement under test) that are not covered by any test case in the original test suite. It would be useful to have test cases automatically generated according to the set of missing SIPs.

Reference:

[1] Boris Vaysburg, Luay H. Tahat, Bogdan Korei, "Dependence Analysis in Reduction of Requirement Based Test Suites", Proceedings of the International Symposium on Software Testing and Analysis (ISSTA'02, Roma, Italy), pp. 107-111, ACM Press, ISBN: 1-58113-562-9, 2002

[2] Ferrante K., Ottenstein K., and Warren J., "The Program Dependence Graph and its use in Optimization," ACM Trans. Progr. Lang. & Systems, 9(5), pp. 319-349., 1987

[3] L.D. Fosdick and L.J. Osterweil, "Data flow analysis in software reliability", ACM Computing Surveys, 8, 3, pp. 305-330, 1976

[4] L.A. Clarke, A. Podgurski, D. J. Richardson, and S. J. Zeil, "A Formal Evaluation of Data Flow Path Selection Criteria", IEEE Trans. Software Eng., 15, 11, pp. 1318-1332, 1989

[5] ITU-T Recommendation Standard Z.100: Languages for Telecommunications Applications – Specification and Description Language, ITU General Secretariat, Geneva, 1999.

[6] Information processing systems – Open Systems Interconnection – Estelle: A formal description technique based on an extended state transition model. International Standard ISO 9074, 1989

[7] Information processing systems – Open Systems Interconnection – LOTOS: A formal description technique based on the temporal ordering of observational behavior. International Standard ISO 8807, 1989

[8] O. Henniger, H. Ural, "Test generation based on control and data dependencies within multi-process SDL specifications", in Proc. of SAM 2000, Col de Porte, Grenoble, France, 2000

[9] G.J. Myers, The art of software testing. New York: John Wiley & Sons, 1979

[10] S. Rapps and E. J. Weyuker, "Selecting Software Test Data Using Data Flow Information", IEEE Trans. Software Eng., 11, 4, pp. 367-375, 1985

[11] P. G. Frankl, and E. J. Weyuker, "An Applicable Family of Data Flow Testing Criteria", IEEE Trans. Software Eng., 14, 10, pp. 1483-1498, 1988

[12] J.W. Laski and B. Korel, "A data-flow oriented program testing strategy", IEEE Trans. Software Eng., 9, 5, pp. 347-354, 1983

[13] H.Ural, "Test Sequence Selection Based on Static Data Flow Analysis," *Computer Communication*, Vol. 10, No. 5, pp. 234-242, Oct. 1987.

[14] H Ural and B Yang, "A Test Sequence Selection Method for Protocol Testing", *IEEE Transactions on Communications*, Vol. 39, No. 4, pp. 514-523, Apr. 1991.

[15] H Ural and A Williams, "Test Generation by Exposing Control and Data Dependencies within System Specifications in SDL," in *Proceedings of IFIP 6th*

International Conference on Formal Description Techniques FORTE'93, pp. 339-354. Oct 1993.

[16] Ural, H.; Yang, B, "Modeling Software for Accurate Data Flow Representation", Proceedings of the 15th International Conference on Software Engineering, May 17-21, Baltimore, 1993, pp. 277-286

[17] B. Beizer, Software Testing Techniques, Second Edition, Van Nostrand Reinhold Inc. 1990

[18] Wang, C., J., Liu, M., T., "Generating Test Cases for EFSM with Given Fault Models," Proc. IEEE Conf. Comp. and Comm. Societies, 2, pp. 774-781, 1993.

[19] Saleh, K., Ural, H., and Williams, A., "Test generation based on control and data dependencies within system specifications in SDL", Computer Communications, Vol.23, No.7, 2000, pp.609-627.

[20] http://www.site.uottawa.ca/~ural/TSR

[21] Tahat, L., Vaysburg, B., Korel, B., Bader, A., "Requirement-Based Automated Black-Box Test Generation," Proc. IEEE COMPSAC, pp. 489-495, 2001.

[22] Richard A. DeMillo , "A. Jefferson Offutt, Constraint-Based Automatic Test Data Generation", IEEE Transactions on Software Engineering, v.17 n.9, p.900-910, September 1991

[23] S. Budkowski , P. Dembinski, "An introduction to Estelle: a specification language for distributed systems" Computer Networks and ISDN Systems, v.14 n.1, p.3-23, March, 1987

[24] Korel, B., "Automated Software Test Data Generation," IEEE Transactions on Software Engineering, 16(8), pp. 870-879, 1990.

[25] Besse, C., Cavalli, A., Lee, D., "An Automatic and Optimized Test Generation Technique Applying to TCP/IP protocol," Proceedings of the 14th IEEE International Conference on Automated Software Engineering, pp. 73-80, 1999

[26] Bourhfir, C., Dssouli, R., Aboulhamid, E., "Automatic Executable Test Case Generation for EFSM Specified Protocols", Proceeding of IWTCS, pp. 75-90, 1997.

[27] C. Bourhfir, R. Dssouli, and E. M. Aboulhamid, "Automatic Test Generation for EFSM-based Systems", http://citeseer.nj.nec.com/114451.html.

[28] Lee, D., Lee, J., "A Well-Defined Estelle Specification for Automatic Test Generation" IEE Transaction on Communications, 40, pp. 526-542.

[29] Algayres, B., Lejeune, Y., Hugonnet, F., "GOAL: Observing SDL Behavior With Object Code," Proceedings of the 7th SDL Forum, Norway, September 1995.

[30] A. Kerbrat, T. Jeron, and R. Groz, "Automated test generation from SDL specifications", in Proc. Of SDL Forum'99, Montreal, Canada, pp. 135-151, 1999

[31] W. Chun and P.D. Amer, "Test case generation for protocols specified in Estelle", in Proc. Of FORTE'90, Madrid, Spain, pp. 197-210, 1990

[32] Ferguson, R., Korel, B., "The Chaining Approach for Software Test Data Generation", ACM Transactions on Software Engineering and Methodology, 5(1), pp. 63-89, 1996.

[33] Bochmann, G,. Petrenko, A., Bellal, O., Maguiraga, S., "Automating the Process of Test Derivation from SDL Specification," Proceedings of the 8th SDL Forum, Evry, France, Sep 1997.

[34] Dick, J., Faivre, A., "Automating the Generation and Sequencing of Test Case from Model-Based Specification," Industrial Strength Formal Methods, 5th international Symposium of Formal Methods Europe, pp. 268-284, Springer-Verlag, April 1992.

[35] Bromstrup, L., Hogrefe, D., "TESDL: Experience With Generating Test Cases From SDL Specifications," Proceedings of 4th SDL Forum, 1989.

[36] Probert, R. L., New and Old Test Techniques: Grey Box Testing and Software Instrumentation, University of Ottawa, Technical Report No. 80-13, 1980.

[37] Olfa Chemli, "Reduced Test Suite Generation", University of Ottawa, Master Thesis in Computer Science.

APPENDIX A: ATM1 System

A.1 Requirements of the ATM1 System

R = {r | r is a requirement which is a **transition**} or

R = {r | r is a requirement which is a **sequence of transitions**}.

R = {$r1$, $r2$, $r3$, $r4$, $r5$}for the ATM1 EFSM where:

r1: User fails to enter a correct pin that matches with the PIN stored in the ATM card in less than four attempts. If pins don't match, and less than four attempts were performed, the system displays an error message, increments the number of attempts and prompts for pin. At the fourth attempt, if user still fails to enter a correct pin, the system prints an error message and ejects the user's ATM card.

$r1$ = T2 T2 T2 **T3**

r2: After entering a correct pin in less than four attempts, user selects withdrawal function. The system adjusts the balance, and displays a menu with withdrawal, deposit, balance inquiry, and exit functions.

$r2 = x$ T4 **T5**

r3: After entering a correct pin in less than four attempts, user selects deposit function. The system adjusts the balance, and displays a menu with withdrawal, deposit, balance inquiry, and exit functions.

$r3 = x$ T4 **T6**

r4: After entering a correct pin in less than four attempts, user selects balance function. The system displays the balance, and then a menu with withdrawal, deposit, balance inquiry, and exit functions.

$r4 = x$ T4 **T7**

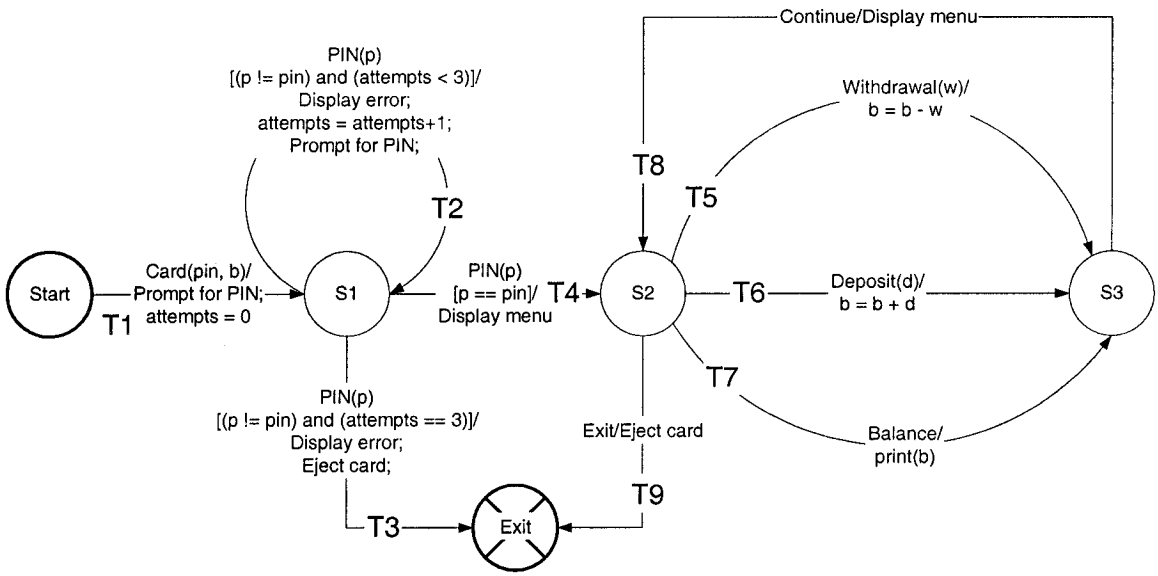
r5: After entering a correct pin in less than four attempts, user selects exit function and the system ejects the user's ATM card.

$r5 = x$ T4 **T9**

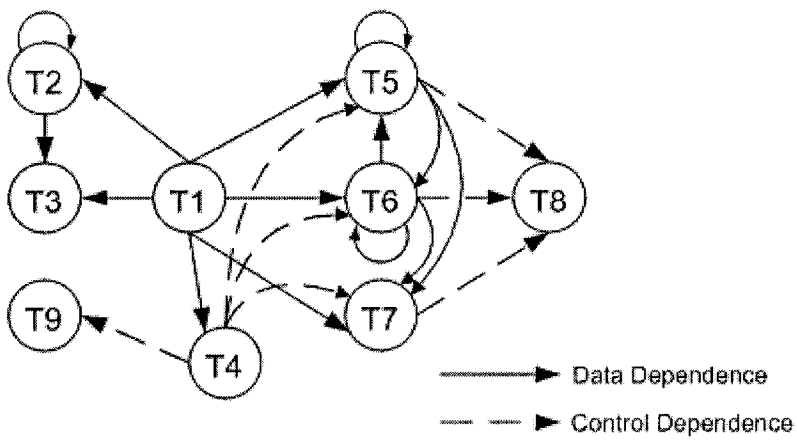
x : T2 may be inserted 0, 1, 2 or 3 times

T# *is the transition under test (TUT)*

A.2 The EFSM Model of the ATM1 System



A.3 SDG of the EFSM of the ATM1 System



A.4 Three Test Suites for the ATM1 System derived from three techniques namely, branch coverage, all-uses coverage and IPO_2-df-chains coverage.

Branch Coverage

Complete Test Suite: {
T1 T4 T5 T8 T6 T8 T7 T8 T9,
T1 T2 T2 T2 T3
}

All-uses Coverage

1. Identify all du-pairs from the EFSM model

Variables	Def(s)	Use(s)
pin	d_{T1}^{pin}	$p_{T2}^{pin}, p_{T3}^{pin}, p_{T4}^{pin}$
attempts	$d_{T1}^{attempts}$	$c_{T2}^{attempts}, p_{T2}^{attempts}$
	$d_{T2}^{attempts}$	$c_{T2}^{attempts}, p_{T2}^{attempts}, p_{T3}^{attempts}$
b	d_{T1}^{b}	c_{T5}^{b}
		c_{T6}^{b}
		c_{T7}^{b}
	d_{T5}^{b}	c_{T5}^{b}
		c_{T6}^{b}
		c_{T7}^{b}
	d_{T6}^{b}	c_{T5}^{b}
		c_{T6}^{b}
		c_{T7}^{b}
w	d^{w}_{T5}	c^{w}_{T5}
d	d^{d}_{T6}	c^{d}_{T6}
p	d^{p}_{T2}	p^{p}_{T2}
	d^{p}_{T3}	p^{p}_{T3}
	d^{p}_{T4}	p^{p}_{T4}

2. Construct an activating path for each du-pair

# DU-Pair	DU-Pairs	Activating Paths
1	$[d^{pin}_{T1}, p^{pin}_{T2}]$	T1 T2
2	$[d^{pin}_{T1}, p^{pin}_{T3}]$	T1 T2 T2 T2 T3
3	$[d^{pin}_{T1}, p^{pin}_{T4}]$	T1 T4
4	$[d^{attempts}_{T1}, p^{attempts}_{T2}]$	T1 T2
5	$[d^{attempts}_{T1}, c^{attempts}_{T2}]$	T1 T2
6	$[d^{attempts}_{T2}, p^{attempts}_{T2}]$	T2 T2
7	$[d^{attempts}_{T2}, c^{attempts}_{T2}]$	T2 T2
8	$[d^{attempts}_{T2}, p^{attempts}_{T3}]$	T2 T2 T2 T3
9	$[d^{b}_{T1}, c^{b}_{T5}]$	T1 T4 T5
10	$[d^{b}_{T1}, c^{b}_{T6}]$	T1 T4 T6
11	$[d^{b}_{T1}, c^{b}_{T7}]$	T1 T4 T7
12	$[d^{b}_{T5}, c^{b}_{T5}]$	T5 T8 T5
13	$[d^{b}_{T5}, c^{b}_{T6}]$	T5 T8 T6
14	$[d^{b}_{T5}, c^{b}_{T7}]$	T5 T8 T7
15	$[d^{b}_{T6}, c^{b}_{T5}]$	T6 T8 T5
16	$[d^{b}_{T6}, c^{b}_{T6}]$	T6 T8 T6
17	$[d^{b}_{T6}, c^{b}_{T7}]$	T6 T8 T7
18	$[d^{w}_{T5}, c^{w}_{T5}]$	T5
19	$[d^{d}_{T6}, c^{d}_{T6}]$	T6
20	$[d^{p}_{T2}, c^{p}_{T2}]$	T2
21	$[d^{p}_{T3}, c^{p}_{T3}]$	T3
22	$[d^{p}_{T4}, c^{p}_{T4}]$	T4

3. Generate test paths to cover all activating paths.

Test id(s)	Test Paths	DU-pairs covered by a test path
1	T1 T2 T2 T2 T3	1, 2, 4, 5, 6, 7, 8, 20, 21
2	T1 T4 T5 T8 T5 T8 T7 T8 T9	3, 9, 12, 14, 18, 22
3	T1 T4 T6 T8 T6 T8 T5 T8 T9	3, 10, 15, 16, 18, 19, 22
4	T1 T4 T7 T8 T5 T8 T6 T8 T7 T8 T9	3, 11, 13, 17, 18, 19, 22

IPO_2-df-chains Coverage

1. Identify all ipo-pairs and their IPO_2-df-chains from the EFSM model

#	ipo-pairs	IPO_2-df-chains and activating paths	
1	$[d_{T1}^{pin}, u_{T3}^{pin}]$	$\{d_{T1}^{pin}, u_{T3}^{pin}\}$ T1 T2 T2 T2 T3	C1
2	$[d_{T1}^{b}, u_{T7}^{b}]$	$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T7	C2
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T7	C3
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T5 T8 T7	C4
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T7	C5
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7	C6
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7	C7
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7	C8
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7	C9
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C10
		$\{d_{T1}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T6}^{b}, d_{T6}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T5}^{b}, d_{T5}^{b}, u_{T7}^{b}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C11

		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T7	C12
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T7	C13
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T5 T8 T7	C14
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T7	C15
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7	C16
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7	C17
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7	C18
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7	C19
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C20
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C21
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T7	C22

		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T7	C23
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T7	C24
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T7	C25
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T7	C26
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T7	C27
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7	C28
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7	C29
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C30
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7	C31
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T7	C32
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T7	C33
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T6 T8 T7	C34
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T7	C35

		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7	C36
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7	C37
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7	C38
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7	C39
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C40
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C41
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T7	C42
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T7	C43
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T6 T8 T7	C44
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T7	C45
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7	C46

		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7	C47
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7	C48
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7	C49
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C50
		$\{d^b_{T1}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C51
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T7	C52
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T7	C53
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T7	C54
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T7	C55
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T7	C56
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T7	C57

		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7	C58
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7	C59
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C60
		$\{d^b_{T1}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T5}, d^b_{T5}, u^b_{T5}, d^b_{T5}, u^b_{T6}, d^b_{T6}, u^b_{T6}, d^b_{T6}, u^b_{T7}\}$ T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7	C61
		$\{d^b_{T1}, u^b_{T7}\}$ T1 T4 T7	C62

Note : $[d^{attmepts}_{T1}, u^{attemtps}_{T3}]$ is not included because its df-chain $\{d^{attempts}_{T1}, u^{attempts}_{T2}, d^{attempts}_{T2}, u^{attempts}_{T2}, d^{attempts}_{T2}, u^{attempts}_{T2}, d^{attempts}_{T2}, u^{pin}_{T3}\}$ violates the definition since there are three occurrences of $u^{attempts}_{T2}, d^{attempts}_{T2}$ in the df-chain.

2. Generate a set of test paths that cover each IPO_2-df-chain at least once

Test id(s)	Test Paths	IPO_2-df-chains covered by a test path
1	T1 T2 T2 T2 T3	C1
2	T1 T4 T6 T8 T5 T8 T7 T8 T9	C2
3	T1 T4 T6 T8 T5 T8 T5 T8 T7 T8 T9	C3
4	T1 T4 T6 T8 T5 T8 T6 T8 T5 T8 T7 T8 T9	C4
5	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T7 T8 T9	C5
6	T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C6
7	T1 T4 T6 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C7
8	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C8
9	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C9
10	T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C10
11	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C11
12	T1 T4 T6 T8 T6 T8 T5 T8 T7 T8 T9	C12
13	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C13
14	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T5 T8 T7 T8 T9	C14
15	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T7 T8 T9	C15

16	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C16
17	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C17
18	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C18
19	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C19
20	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C20
21	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C21
22	T1 T4 T5 T8 T7 T8 T9	C22
23	T1 T4 T5 T8 T5 T8 T7 T8 T9	C23
24	T1 T4 T5 T8 T6 T8 T5 T8 T7 T8 T9	C24
25	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T7 T8 T9	C25
26	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C26
27	T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C27
28	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T7 T8 T9	C28
29	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C29
30	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C30
31	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T7 T8 T9	C31
32	T1 T4 T5 T8 T6 T8 T7 T8 T9	C32
33	T1 T4 T5 T8 T6 T8 T6 T8 T7 T8 T9	C33
34	T1 T4 T5 T8 T6 T8 T5 T8 T6 T8 T7 T8 T9	C34
35	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T7 T8 T9	C35
36	T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C36
37	T1 T4 T5 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C37
38	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C38
39	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C39
40	T1 T4 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C40
41	T1 T4 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C41
42	T1 T4 T5 T8 T5 T8 T6 T8 T7 T8 T9	C42
43	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C43
44	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T6 T8 T7 T8 T9	C44
45	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T7 T8 T9	C45
46	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C46
47	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C47
48	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C48
49	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C49
50	T1 T4 T5 T8 T5 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C50
51	T1 T4 T5 T8 T5 T8 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6	C51

	T8 T7 T8 T9	
52	T1 T4 T6 T8 T7 T8 T9	C52
53	T1 T4 T6 T8 T6 T8 T7 T8 T9	C53
54	T1 T4 T6 T8 T5 T8 T6 T8 T7 T8 T9	C54
55	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T7 T8 T9	C55
56	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C56
57	T1 T4 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C57
58	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T7 T8 T9	C58
59	T1 T4 T6 T8 T6 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C59
60	T1 T4 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C60
61	T1 T4 T6 T8 T6 T8 T5 T8 T5 T8 T6 T8 T6 T8 T7 T8 T9	C61
62	T1 T4 T7 T8 T9	C62

A.5 Test Results after applying STSR and DTSR programs to the test suites in A.4

By applying STSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	1	1	
T6	1	1	
T7	1	1	
T9	1	1	

All-uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	3	3	
T6	2	2	
T7	2	2	
T9	3	1	Test_3, Test_4

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	58	14	Test_7, 9, 10, 11, 14, 15, 16, 17, 18, 19, 20, 21, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 54, 55, 56, 57, 58, 59, 60, 61
T6	58	14	Test_3, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 27, 29, 30, 31, 32, 33, 37, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61
T7	61	29	Test_7, 9, 10, 11, 14, 15, 16, 17, 18, 19, 20, 21, 27, 29, 30, 31, 37, 39, 40, 41, 44, 45, 46, 47, 48, 49, 50, 51, 57, 59, 60, 61
T9	61	1	Test_3 – Test_62

By applying DTSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	1	1	
T6	1	1	
T7	1	1	
T9	1	1	

All-uses Coverage

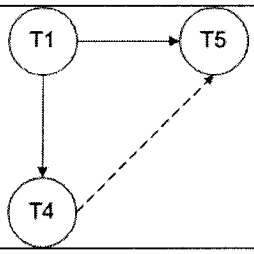
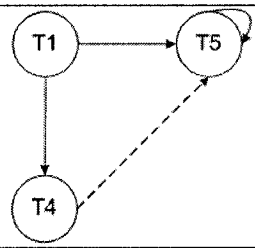
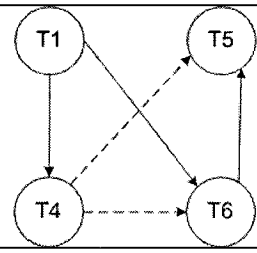
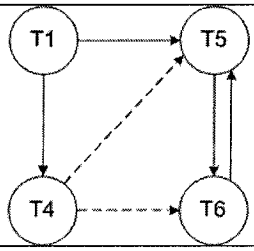
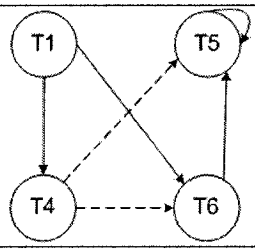
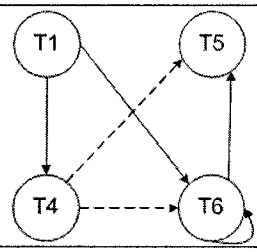
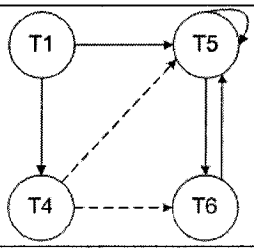
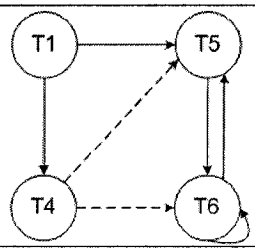
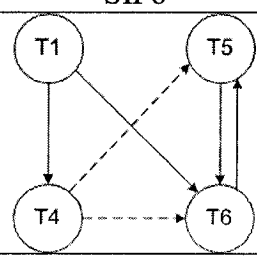
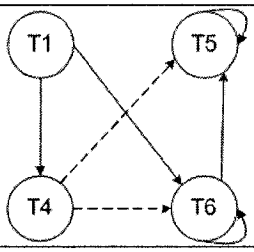
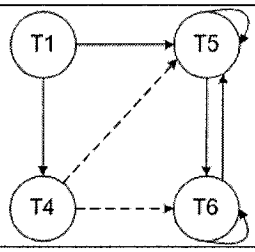
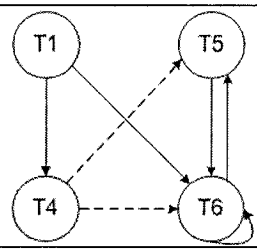
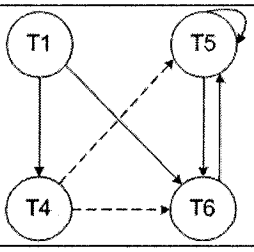
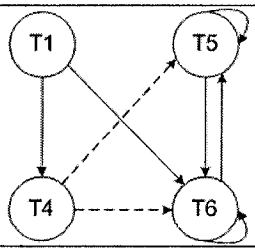
Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	3	3	
T6	2	2	
T7	2	2	

T9	3	1	Test_3, Test_4

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T3	1	1	
T5	58	30	Test_32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 54, 55, 56, 57, 58, 59, 60, 61
T6	58	30	Test_3, 7, 9, 10, 11, 13, 17, 19, 20, 21, 27, 29, 30, 31, 32, 33, 42, 43, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61
T7	61	61	
T9	61	1	Test_3 – Test_62

A.6 SIPs w.r.t. T5

		
SIP1	SIP2	SIP3
		
SIP4	SIP5	SIP6
		
SIP7	SIP8	SIP9
		
SIP10	SIP11	SIP12
		
SIP13	SIP14	

A.7 ".efsm" file for the EFSM of the ATM1 system

```
ATM_System
5 0 4

transition T1
0 1
Card(pin, b) /
Prompt_for_PIN()
attempts := constant

transition T2
1 1
PIN(p)
p, pin, attempts /* [(p!=pin) and (attempts<3)] */ /
Display_error()
attempts := function(attempts)
Prompt_for_PIN()

transition T3
1 4
PIN(p)
p, pin, attempts /* [(p!=pin) and (attempts==3)] */ /
Display_error()
Eject_card()

transition T4
1 2
PIN(p)
p, pin /* [p==pin] */ /
Display_menu()

transition T5
2 3
Withdrawal(w) /
b := function(b, w)

transition T6
2 3
Deposit(d) /
b := function(b, d)

transition T7
2 3
Balance() /
print(b)
```

```
transition T8
3 2
Continue() /
Display_menu()

transition T9
2 4
Exit() /
Eject_card()
```

A.8 An example ".ts" file for the EFSM of the ATM1 System

```
ATM_System
T5 T6

test Test_1 T1 T4 T5 T8 T9
test Test_2 T1 T4 T5 T8 T5 T8 T9
test Test_3 T1 T4 T5 T8 T6 T8 T9
test Test_4 T1 T4 T5 T8 T7 T8 T9
test Test_5 T1 T4 T6 T8 T5 T8 T9
test Test_6 T1 T4 T7 T8 T5 T8 T9
test Test_7 T1 T2 T4 T5 T8 T9
test Test_8 T1 T2 T4 T5 T8 T5 T8 T9
test Test_9 T1 T2 T4 T5 T8 T6 T8 T9
test Test_10 T1 T2 T4 T5 T8 T7 T8 T9
test Test_11 T1 T2 T4 T6 T8 T5 T8 T9
test Test_12 T1 T2 T4 T7 T8 T5 T8 T9
test Test_13 T1 T2 T2 T4 T5 T8 T9
test Test_14 T1 T2 T2 T4 T5 T8 T5 T8 T9
test Test_15 T1 T2 T2 T4 T5 T8 T6 T8 T9
test Test_16 T1 T2 T2 T4 T5 T8 T7 T8 T9
test Test_17 T1 T2 T2 T4 T6 T8 T5 T8 T9
test Test_18 T1 T2 T2 T4 T7 T8 T5 T8 T9
```

A.9 An example ".sip" file for the EFSM of the ATM1 System

```
ATM_System
T5

ip ST5_0 Test_1 Test_3 Test_4 Test_6 Test_7 Test_9 Test_10 Test_12 Test_13 Test_15
Test_16
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 0 dat
```

```
ip ST5_1 Test_2 Test_8 Test_14
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 0 dat
inc 2 dat

ip ST5_2 Test_5 Test_11 Test_17
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
node 3 T6
inc 1 ctl
inc 0 dat

ip UT5_0
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 0 dat
inc 2 dat
inc 3 dat
node 3 T6
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT5_1
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat
```

```
ip UT5_2
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 2 dat
inc 3 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat

ip UT5_3
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT5_4
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 3 dat
node 3 T6
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT5_5
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
```

```
node 3 T6
inc 0 dat
inc 1 ctl
inc 3 dat

ip UT5_6
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 2 dat
inc 3 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT5_7
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 2 dat
inc 3 dat
node 3 T6
inc 0 dat
inc 1 ctl

ip UT5_8
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 3 dat
node 3 T6
inc 1 ctl
inc 2 dat

ip UT5_9
node 0 T1
node 1 T4
```

```
inc 0 dat
node 2 T5
inc 1 ctl
inc 0 dat
inc 2 dat
inc 3 dat
node 3 T6
inc 1 ctl
inc 2 dat

ip UT5_10
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 2 dat
inc 3 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 3 dat


ATM_System
T6

ip ST6_0 Test_5 Test_11 Test_17
node 0 T1
node 1 T4
inc 0 dat
node 2 T6
inc 1 ctl
inc 0 dat

ip ST6_1 Test_3 Test_9 Test_15
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 0 dat
node 3 T6
inc 2 dat
inc 1 ctl
```

```
ip UT6_0
node 0 T1
node 1 T4
inc 0 dat
node 2 T6
inc 0 dat
inc 1 ctl

ip UT6_1
node 0 T1
node 1 T4
inc 0 dat
node 2 T6
inc 0 dat
inc 2 dat

ip UT6_2
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat
node 3 T6
inc 2 dat
inc 1 ctl
inc 3 dat

ip UT6_3
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
inc 2 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat

ip UT6_4
node 0 T1
node 1 T4
```

```
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT6_5
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 3 dat
inc 1 ctl
inc 2 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat

ip UT6_6
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 1 ctl
inc 3 dat
node 3 T6
inc 0 dat
inc 1 ctl
inc 2 dat

ip UT6_7
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
node 3 T6
inc 2 dat
inc 1 ctl
```

```
 ip UT6_8
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
node 3 T6
inc 2 dat
inc 1 ctl
inc 3 dat

ip UT6_9
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 3 dat
node 3 T6
inc 2 dat
inc 1 ctl

ip UT6_10
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 3 dat
node 3 T6
inc 2 dat
inc 1 ctl
inc 3 dat

ip UT6_11
node 0 T1
node 1 T4
inc 0 dat
node 2 T5
inc 0 dat
inc 1 ctl
inc 2 dat
inc 3 dat
```

```
node 3 T6
inc 2 dat
inc 1 ctl
```

APPENDIX B: ATM2 System

B.1 Requirements of the ATM2 System

R = {r | r is a requirement which is a **transition**} or

R = {r | r is a requirement which is a **sequence of transitions**}.

R = {$r1$, $r2$, $r3$, $r4$, $r5$, $r6$, $r7$, $r8$} for the ATM2 EFSM where:

r1: When the system is on, a user inserts an ATM card into the card reader slot but fails to enter his/her Personal Identification Number (PIN) within a given time. System ejects the card and displays welcome message.

$r1$ = T3 **T8**

r2: After inserting an ATM card, a user fails to enter a correct pin that matches with the PIN stored in the ATM card in less than four attempts. If pins don't match, and less than four attempts were performed, the system displays an error message, increments the number of attempts and prompts for pin. At the fourth attempt, if user still fails to enter a correct pin, the system prints an error message and ejects the user's ATM card.

$r2$ = T3 T5 T5 T5 **T6**

r3: After a user inserts an ATM card and enter a correct pin in less than four attempts, the system obtains his/her balance then displays option menu

$r3$ = T3 X T7 **T9**

r4: A user selects withdrawal from the option menu and enters withdrawal amount within a given time. However, the amount specified is more than amount available in the ATM cash dispenser. System indicates that requested amount is unavailable and asks the customer to specify amount then waits for the amount.

*r*4 = T12 **T14**

r5: A user selects withdrawal from the option menu and enters withdrawal amount within a given time. However, the amount specified is more than account balance. System indicates that account balance is not enough and asks the customer to specify amount then waits for the amount.

*r*5 = T12 **T16**

r6: A user selects withdrawal from the option menu and enters withdrawal amount within a given time. System updates the user's account and the amount of money currently in the ATM, waits for the user to take cash for a given delay then displays the option menu.

*r*6 = T12 **T15**

r7: A user selects Balance Inquiry from the option menu. The system displays Balance for a given delay and displays the option menu.

*r*7 = **T10**

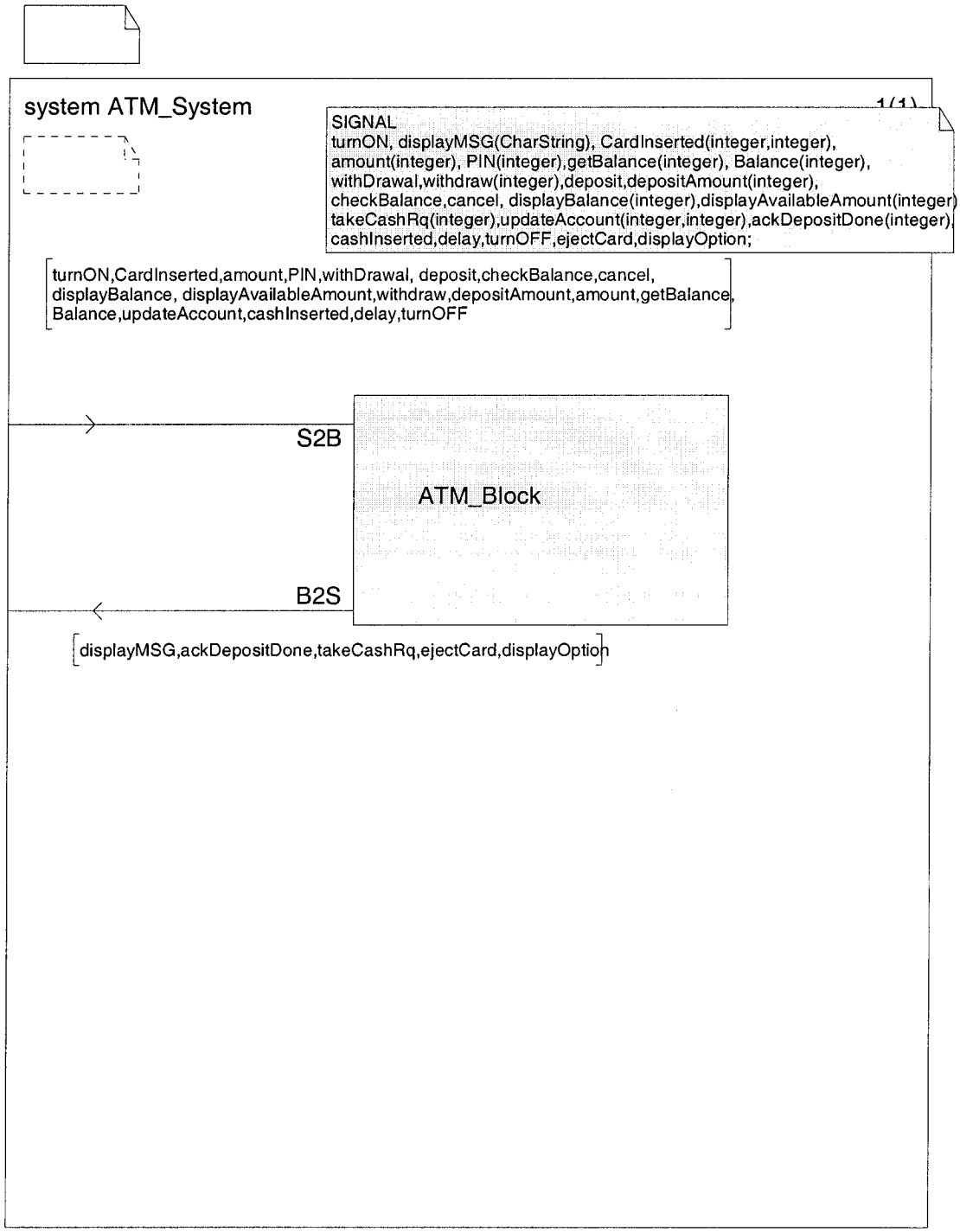
r8: A user selects deposit from the option menu, enters deposit amount and inserts cash within a given delay. System informs the user that the transaction is success then displays the option menu.

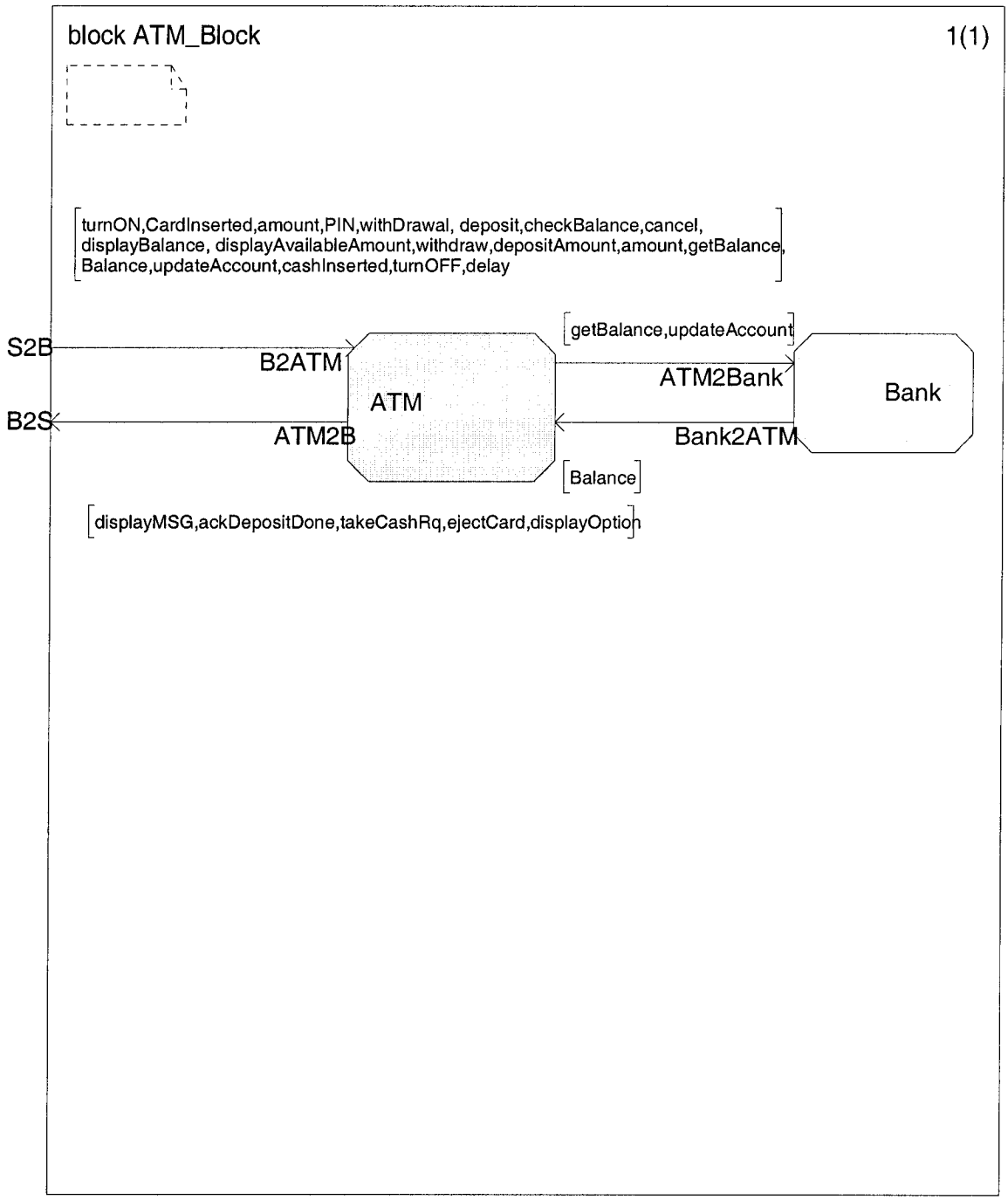
*r*8 = T17 T18 **T20**

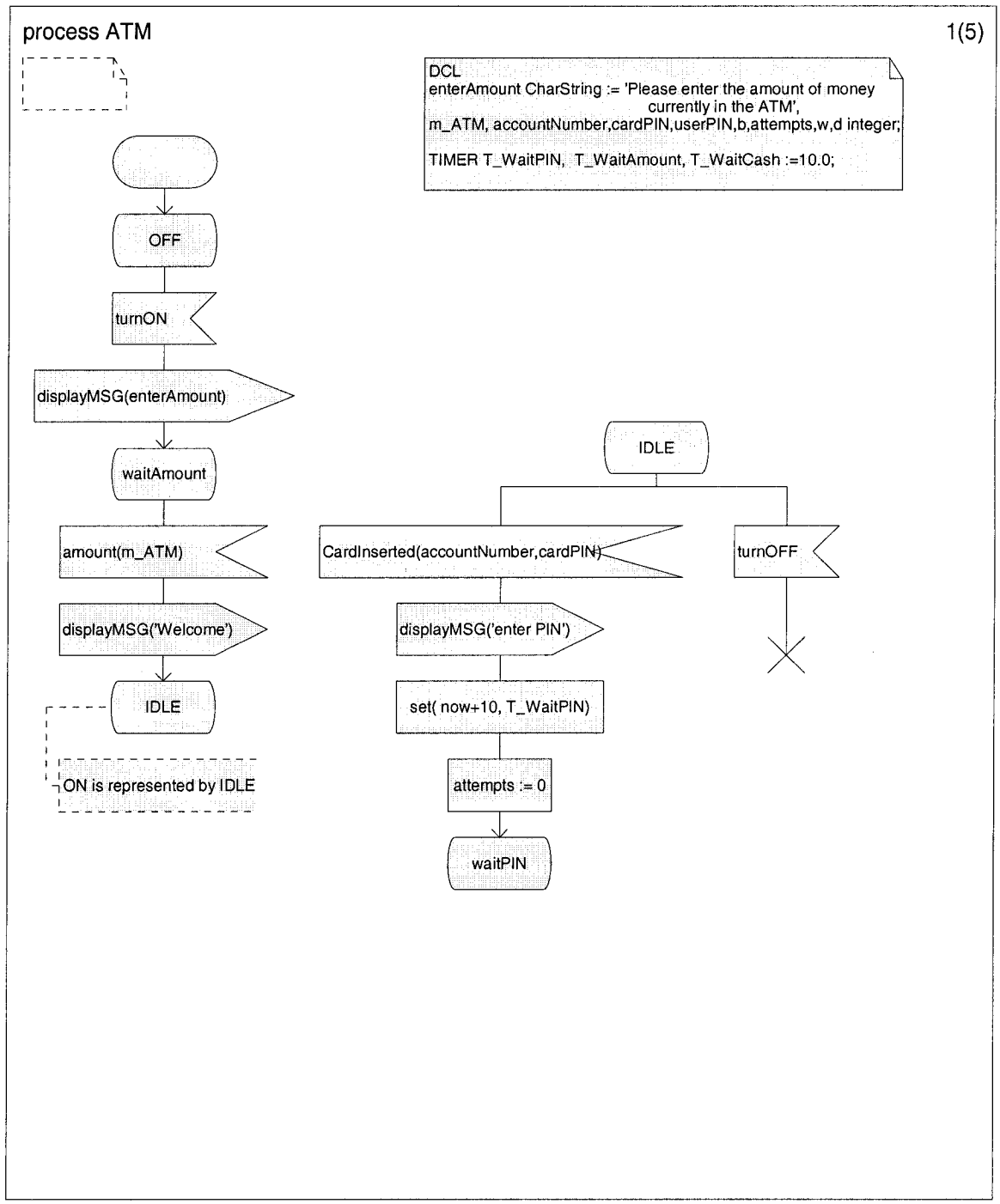
x : T5 may be inserted 0, 1, 2 or 3 times

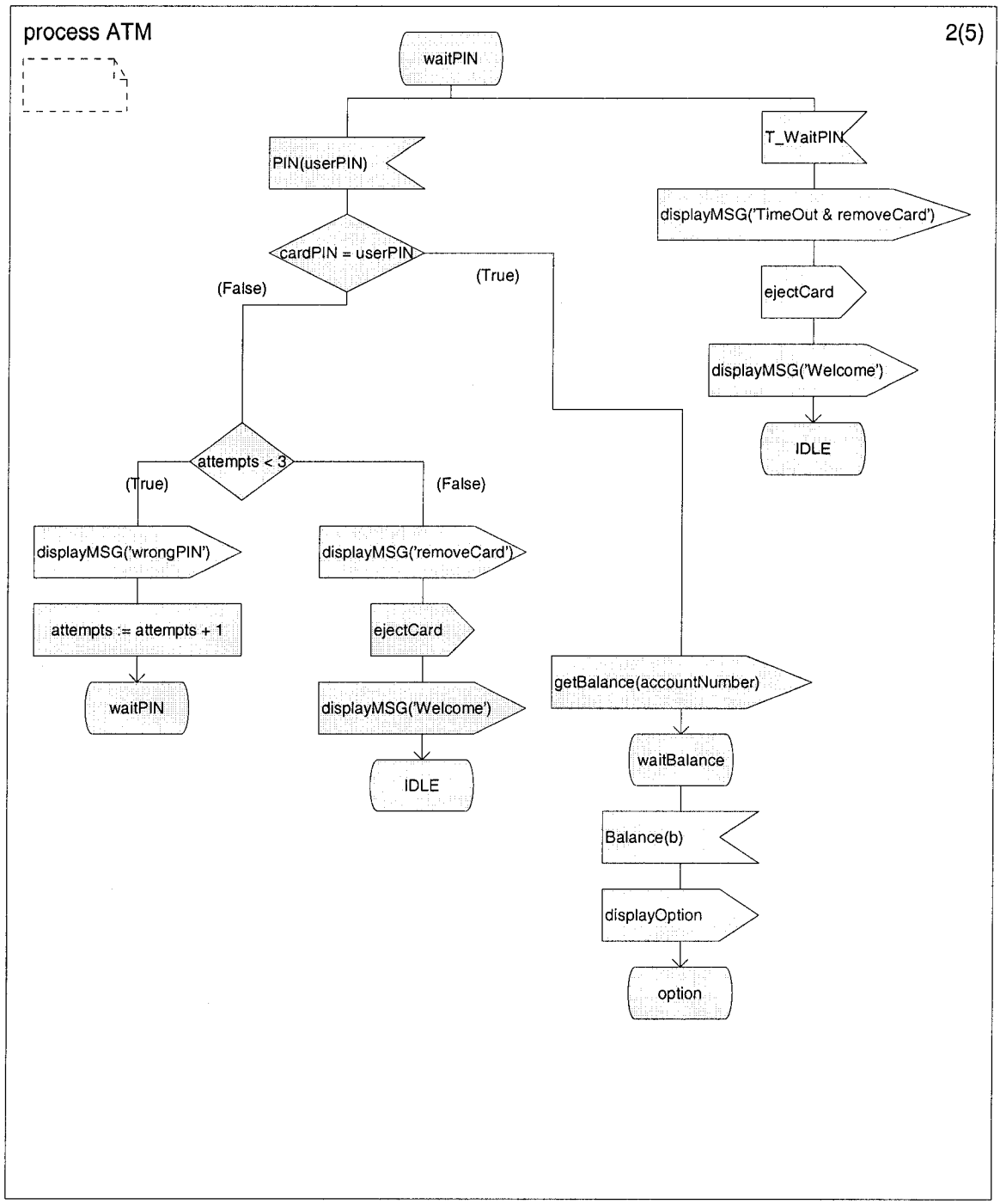
T# *is the transition under test (TUT)*

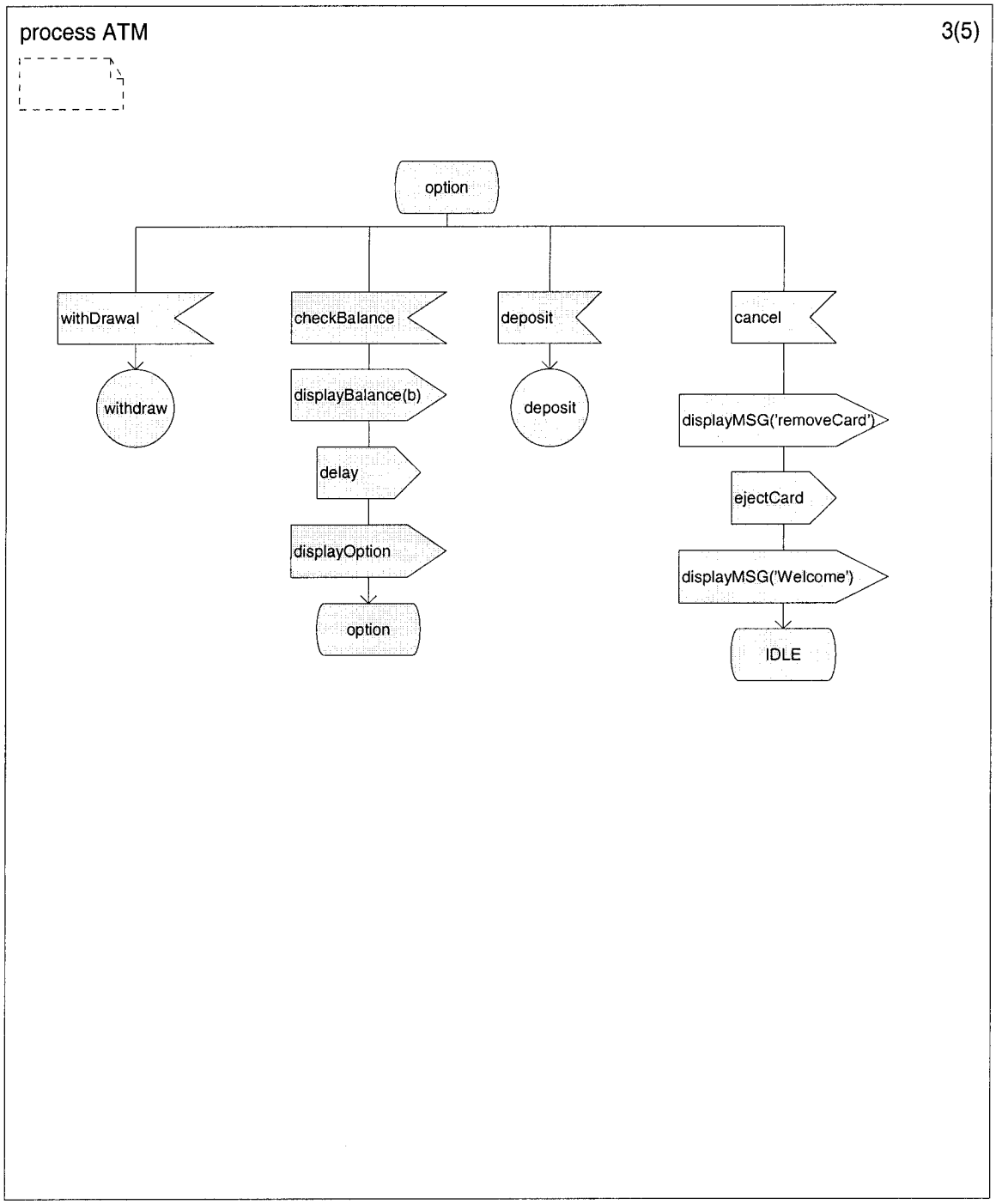
B.2 The SDL Diagram of the ATM2 System

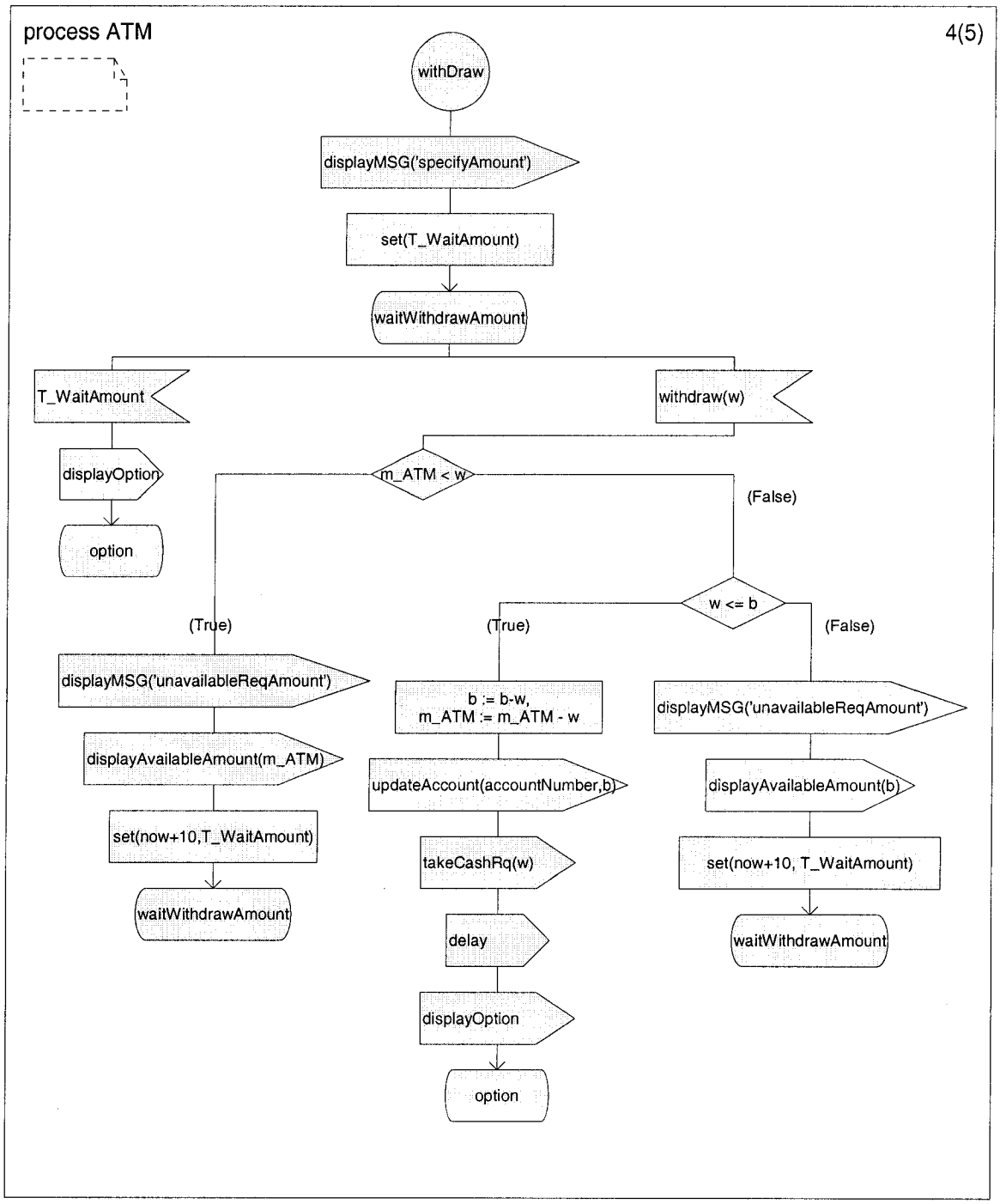


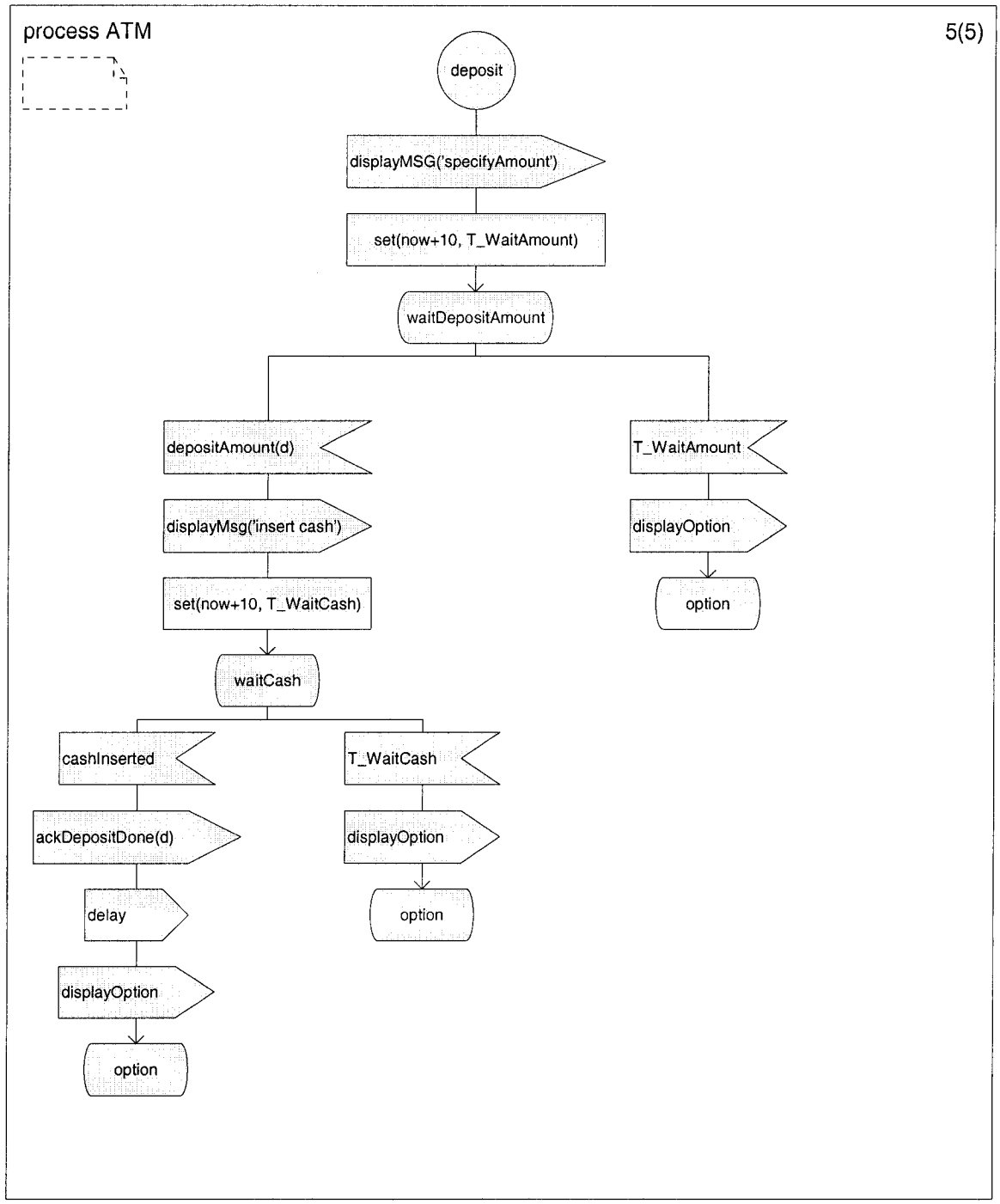

block ATM_Block
1(1)
turnON,CardInserted,amount,PIN,withDrawal, deposit,checkBalance,cancel,
displayBalance, displayAvailableAmount,withdraw,depositAmount,amount,getBalance,
Balance,updateAccount,cashInserted,turnOFF,delay
getBalance,updateAccount
S2B
B2ATM
ATM2Bank
Bank
ATM
B2S
ATM2B
Bank2ATM
Balance
displayMSG,ackDepositDone,takeCashRq,ejectCard,displayOption


process ATM
1(5)
DCL
enterAmount CharString := 'Please enter the amount of money
currently in the ATM',
m_ATM, accountNumber,cardPIN,userPIN,b,attempts,w,d integer;
TIMER T_WaitPIN, T_WaitAmount, T_WaitCash :=10.0;
OFF
turnON
displayMSG(enterAmount)
waitAmount
amount(m_ATM)
displayMSG('Welcome')
IDLE
ON is represented by IDLE
IDLE
CardInserted(accountNumber,cardPIN)
turnOFF
displayMSG('enter PIN')
set(now+10, T_WaitPIN)
attempts := 0
waitPIN

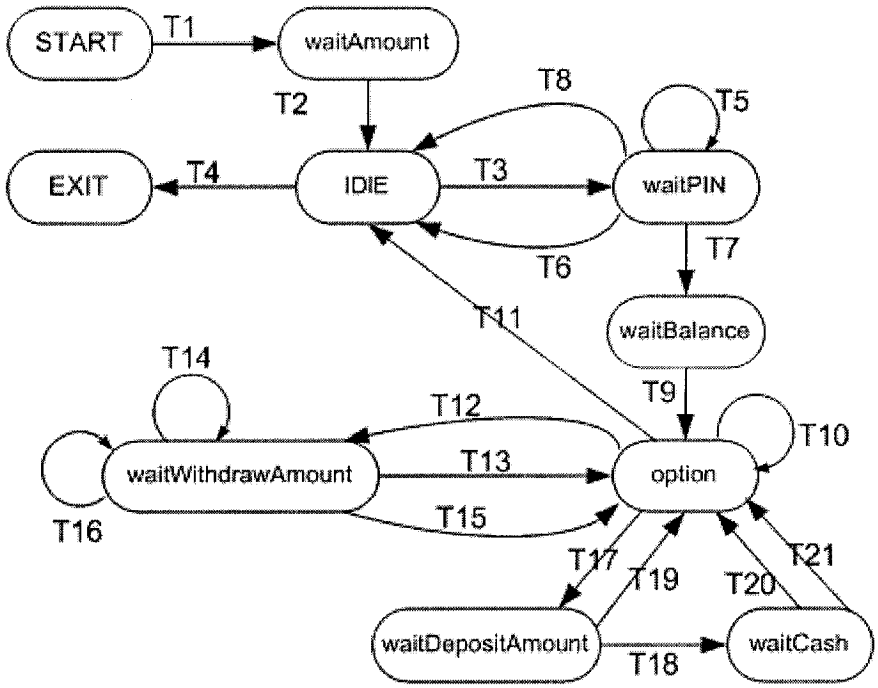

process ATM
2(5)
waitPIN
PIN(userPIN)
cardPIN = userPIN
(False)
(True)
attempts < 3
(True)
(False)
displayMSG('wrongPIN')
attempts := attempts + 1
waitPIN
displayMSG('removeCard')
ejectCard
displayMSG('Welcome')
IDLE
T_WaitPIN
displayMSG('TimeOut & removeCard')
ejectCard
displayMSG('Welcome')
IDLE
getBalance(accountNumber)
waitBalance
Balance(b)
displayOption
option


process ATM
3(5)
option
withDrawal
checkBalance
deposit
cancel
withdraw
displayBalance(b)
deposit
displayMSG('removeCard')
delay
ejectCard
displayOption
displayMSG('Welcome')
option
IDLE


process ATM
4(5)
withDraw
displayMSG('specifyAmount')
set(T_WaitAmount)
waitWithdrawAmount
T_WaitAmount
withdraw(w)
displayOption
option
m_ATM < w
(False)
(True)
w <= b
(True)
(False)
displayMSG('unavailableReqAmount')
displayAvailableAmount(m_ATM)
set(now+10,T_WaitAmount)
waitWithdrawAmount
b := b-w,
m_ATM := m_ATM - w
updateAccount(accountNumber,b)
takeCashRq(w)
delay
displayOption
option
displayMSG('unavailableReqAmount')
displayAvailableAmount(b)
set(now+10, T_WaitAmount)
waitWithdrawAmount


process ATM
5(5)
deposit
displayMSG('specifyAmount')
set(now+10, T_WaitAmount)
waitDepositAmount
depositAmount(d)
displayMsg('insert cash')
set(now+10, T_WaitCash)
waitCash
cashInserted
ackDepositDone(d)
delay
displayOption
option
T_WaitCash
displayOption
option
T_WaitAmount
displayOption
option

B.3 The EFSM Model of the ATM2 System



Transition	
T1	turnON()/ displayMSG(enterAmount)
T2	amount(m_ATM)/ displayMSG('Welcome')
T3	CardInserted(accountNumber,cardPIN)/ displayMSG('enter PIN') set(now+10,T_WaitPIN) attemtps := 0
T4	Turnoff()
T5	PIN(userPIN) [userPIN != cardPIN && attempts <3]/ displayMSG('wrongPIN') attemtps := attempts + 1
T6	PIN(userPIN) [userPIN != cardPIN && attempts >= 3]/ displayMSG('removeCard') ejectCard() displayMSG('Welcome')
T7	PIN(userPIN) [userPIN == cardPIN]/ getBalance(accountNumber)
T8	[T_WaitPIN]/ displayMSG('TimeOut & removeCard') ejectCard()

	displayMSG('Welcome')
T9	Balance(b)/ displayOption()
T10	checkBalance()/ displayBalance(b) delay() displayOption()
T11	cancel()/ displayMSG('removeCard') ejectCard() displayMSG('Welcome')
T12	withDrawal/ displayMSG('specifyAmount') set(Now+10,T_WaitAmount)
T13	[T_WaitAmount]/ displayOption()
T14	withdraw(w) [m_ATM < w] / displayMSG('unavailableReqAmount') displayAvailableAmount(m_ATM) set(now+10,T_WaitAmount)
T15	withdraw(w) [m_ATM >= w and w <= b] / b := b-w m_ATM := m_ATM-w updateAccount(accountNumber, b) takeCashRq(w) delay() displayOption()
T16	withdraw(w) [m_ATM >= w and w > b] / displayMSG('unavailableReqAmount') displayAvailableAmount(b) set(now+10,T_WaitAmount)
T17	deposit()/ displayMSG(constant) set(now+10,T_WaitAmount)
T18	depositAmount(d)/ displayMSG(constant) set(now+10,T_WaitCash)
T19	T_WaitAmount/ displayOption()
T20	cashInserted()/ ackDepositDone(d) delay() displayOption()

T21	T_WaitCash/displayOption()

B.4 Three Test Suites for the ATM2 System derived from three techniques namely, branch coverage, all-uses coverage and IPO_2-df-chains coverage

Branch Coverage

Complete Test Suite: {
T1 T2 T3 T5 T5 T5 T6 T4,
T1 T2 T3 T8 T3 T7 T9 T10 T12 T14 T13 T12 T16 T15 T17 T19 T17 T18 T20 T17 T18 T21 T11 T4
}

All-uses Coverage

1. Identify all du-pairs from the EFSM Model

Variables	Def(s)	Use(s)
attempts	$d_{T3}^{attemtps}$	$p_{T5}^{attemtps}, p_{T6}^{attemtps}$
	$d_{T5}^{attemtps}$	$c_{T5}^{attemtps}, p_{T5}^{attemtps}, p_{T6}^{attemtps}$
B	d_{T9}^{b}	$c_{T10}^{b}, c_{T15}^{b}, c_{T16}^{b}, p_{T15}^{b}, p_{T16}^{b}$
	d_{T15}^{b}	$c_{T15}^{b}, c_{T16}^{b}, p_{T15}^{b}, p_{T16}^{b}$
cardPIN	$d_{T3}^{cardPIN}$	$p_{T5}^{cardPIN}, p_{T6}^{cardPIN}, p_{T7}^{cardPIN}$
D	d_{T18}^{d}	c_{T20}^{d}
m_ATM	$d_{T2}^{m\_ATM}$	$c_{T16}^{m\_ATM}, p_{T14}^{m\_ATM}, p_{T15}^{m\_ATM}, p_{T16}^{m\_ATM}$
	$d_{T15}^{m\_ATM}$	$c_{T16}^{m\_ATM}, p_{T14}^{m\_ATM}, p_{T15}^{m\_ATM}, p_{T16}^{m\_ATM}$
T_WaitAmount	$d_{T14}^{T\_WaitAmount}$	$p_{T13}^{T\_WaitAmount}$
	$d_{T16}^{T\_WaitAmount}$	$p_{T13}^{T\_WaitAmount}$
	$d_{T17}^{T\_WaitAmount}$	$p_{T19}^{T\_WaitAmount}$
T_WaitCash	$d_{T18}^{T\_WaitCash}$	$p_{T21}^{T\_WaitCash}$
T_WaitPIN	$d_{T3}^{T\_WaitPIN}$	$p_{T8}^{T\_WaitPIN}$
userPIN	$d_{T5}^{userPIN}$	$p_{T5}^{userPIN}$
	$d_{T6}^{userPIN}$	$p_{T6}^{userPIN}$
	$d_{T7}^{userPIN}$	$p_{T7}^{userPIN}$
W	d_{T14}^{w}	p_{T14}^{w}
	d_{T15}^{w}	c_{T15}^{w}, p_{T15}^{w}

	d^{w}_{T16}	p^{w}_{T16}

2. Construct an activating path for each du-pair

# DU_Pair	DU-Pairs	Activating Paths
1	$[d^{attemtps}_{T3}, p^{attemtps}_{T5}]$	T3 T5
2	$[d^{attemtps}_{T3}, p^{attemtps}_{T6}]$	T3 T5 T5 T5 T6
3	$[d^{attemtps}_{T5}, c^{attemtps}_{T5}]$	T5 T5
4	$[d^{attemtps}_{T5}, p^{attemtps}_{T5}]$	T5 T5
5	$[d^{attemtps}_{T5}, p^{attemtps}_{T6}]$	T5 T6
6	$[d^{b}_{T9}, c^{b}_{T10}]$	T9 T10
7	$[d^{b}_{T9}, c^{b}_{T15}]$	T9 T12 T15
8	$[d^{b}_{T9}, c^{b}_{T16}]$	T9 T12 T16
9	$[d^{b}_{T9}, p^{b}_{T15}]$	T9 T12 T15
10	$[d^{b}_{T9}, p^{b}_{T16}]$	T9 T12 T16
11	$[d^{b}_{T15}, c^{b}_{T15}]$	T15 T12 T15
12	$[d^{b}_{T15}, c^{b}_{T16}]$	T15 T12 T16
13	$[d^{b}_{T15}, p^{b}_{T15}]$	T15 T12 T15
14	$[d^{b}_{T15}, p^{b}_{T16}]$	T15 T12 T16
15	$[d^{cardPIN}_{T3}, p^{cardPIN}_{T5}]$	T3 T5
16	$[d^{cardPIN}_{T3}, p^{cardPIN}_{T6}]$	T3 T5 T5 T5 T6
17	$[d^{cardPIN}_{T3}, p^{cardPIN}_{T7}]$	T3 T7
18	$[d^{d}_{T18}, c^{d}_{T20}]$	T18 T20
19	$[d^{m\_ATM}_{T2}, c^{m\_ATM}_{T16}]$	T2 T3 T7 T9 T12 T16
20	$[d^{m\_ATM}_{T2}, p^{m\_ATM}_{T14}]$	T2 T3 T7 T9 T12 T14
21	$[d^{m\_ATM}_{T2}, p^{m\_ATM}_{T15}]$	T2 T3 T7 T9 T12 T15
22	$[d^{m\_ATM}_{T2}, p^{m\_ATM}_{T16}]$	T2 T3 T7 T9 T12 T16
23	$[d^{m\_ATM}_{T15}, c^{m\_ATM}_{T16}]$	T15 T12 T16
24	$[d^{m\_ATM}_{T15}, p^{m\_ATM}_{T14}]$	T15 T12 T14
25	$[d^{m\_ATM}_{T15}, p^{m\_ATM}_{T15}]$	T15 T12 T15
26	$[d^{m\_ATM}_{T15}, p^{m\_ATM}_{T16}]$	T15 T12 T16
27	$[d^{T\_WaitAmount}_{T14}, p^{T\_WaitAmount}_{T13}]$	T14 T13
28	$[d^{T\_WaitAmount}_{T16}, p^{T\_WaitAmount}_{T13}]$	T16 T13
29	$[d^{T\_WaitAmount}_{T17}, p^{T\_WaitAmount}_{T19}]$	T17 T19

30	$[d_{T18}^{T\_WaitCash}, p_{T21}^{T\_WaitCash}]$	T18 T21
31	$[d_{T3}^{T\_WaitPIN}, p_{T8}^{T\_WaitPIN}]$	T3 T8
32	$[d_{T5}^{userPIN}, p_{T5}^{userPIN}]$	T5 T5
33	$[d_{T6}^{userPIN}, p_{T6}^{userPIN}]$	T6
34	$[d_{T7}^{userPIN}, p_{T7}^{userPIN}]$	T7
35	$[d_{T14}^{w}, p_{T14}^{w}]$	T14
36	$[d_{T15}^{w}, c_{T15}^{w}]$	T15
37	$[d_{T15}^{w}, p_{T15}^{w}]$	T15
38	$[d_{T16}^{w}, p_{T16}^{w}]$	T16

3. Generate test paths to cover all activating paths

Test id(s)	Test Paths	DU-pairs covered by a test path
1	T1 T2 T3 T7 T9 T12 T16 T13 T11 T4	17, 28, 34, 38
2	T1 T2 T3 T7 T9 T12 T14 T15 T12 T16 T15 T11 T4	12, 14, 17, 20, 23, 26, 34, 35, 36, 37, 38
3	T1 T2 T3 T7 T9 T12 T15 T12 T14 T13 T17 T18 T21 T11 T4	17, 24, 27, 30, 34, 35, 36, 37
4	T1 T2 T3 T8 T3 T5 T7 T9 T10 T11 T4	1, 6, 15, 31
5	T1 T2 T3 T5 T5 T5 T6 T4	1, 2, 3, 4, 5, 15, 16, 32, 33
6	T1 T2 T3 T7 T9 T12 T15 T12 T15 T17 T19 T11 T4	7, 9, 11, 13, 17, 21, 25, 29, 34, 36, 37
7	T1 T2 T3 T7 T9 T12 T16 T15 T17 T18 T20 T11 T4	8, 10, 17, 18, 19, 22, 34, 36, 37, 38

IPO_2-df-chains Coverage

1. Identify all ipo-pairs and their IPO_2-df-chains from the EFSM model

#	ipo-pairs	IPO_2-df-chains	
1	$[d_{T18}^{d}, u_{T20}^{d}]$	$\{d_{T18}^{d}, u_{T20}^{d}\}$	C1
2	$[d_{T2}^{m\_ATM}, u_{T14}^{m\_ATM}]$	$\{d_{T2}^{m\_ATM}, u_{T14}^{m\_ATM}\}$	C2
		$\{d_{T2}^{m\_ATM}, u_{T15}^{m\_ATM}, d_{T15}^{m\_ATM}, u_{T14}^{m\_ATM}\}$	C3
		$\{d_{T2}^{m\_ATM}, u_{T15}^{m\_ATM}, d_{T15}^{m\_ATM}, u_{T15}^{m\_ATM}, d_{T15}^{m\_ATM}, u_{T14}^{m\_ATM}\}$	C4
3	$[d_{T15}^{w}, u_{T14}^{m\_ATM}]$	$\{d_{T15}^{w}, u_{T15}^{w}, d_{T15}^{m\_ATM}, u_{T14}^{m\_ATM}\}$	C5
		$\{d_{T15}^{w}, u_{T15}^{w}, d_{T15}^{m\_ATM}, u_{T15}^{m\_ATM}, d_{T15}^{m\_ATM}, u_{T14}^{m\_ATM}\}$	C6

4	$[d^b_{T9}, u^b_{T10}]$	$\{d^b_{T9}, u^b_{T10}\}$	C7
		$\{d^b_{T9}, u^b_{T15}, d^b_{T15}, u^b_{T10}\}$	C8
		$\{d^b_{T9}, u^b_{T15}, d^b_{T15}, u^b_{T15}, d^b_{T15}, u^b_{T10}\}$	C9
5	$[d^w_{T15}, u^b_{T10}]$	$\{d^w_{T15}, u^w_{T15}, d^b_{T15}, u^b_{T10}\}$	C10
		$\{d^w_{T15}, u^w_{T15}, d^b_{T15}, u^b_{T15}, d^b_{T15}, u^b_{T10}\}$	C11
6	$[d^b_{T9}, u^b_{T16}]$	$\{d^b_{T9}, u^b_{T16}\}$	C12
		$\{d^b_{T9}, u^b_{T15}, d^b_{T15}, u^b_{T16}\}$	C13
		$\{d^b_{T9}, u^b_{T15}, d^b_{T15}, u^b_{T15}, d^b_{T15}, u^b_{T16}\}$	C14
7	$[d^w_{T15}, u^b_{T16}]$	$\{d^w_{T15}, u^w_{T15}, d^b_{T15}, u^b_{T16}\}$	C15
		$\{d^w_{T15}, u^w_{T15}, d^b_{T15}, u^b_{T15}, d^b_{T15}, u^b_{T16}\}$	C16
8	$[d^w_{T15}, u^w_{T15}]$	$\{d^w_{T15}, u^w_{T15}\}$	C17

2. Construct an activating path for each IPO_2-df-chain

#	Activating Path
C1	T18 T20
C2	T2 T3 T7 T9 T12 T14
C3	T2 T3 T7 T9 T12 T15 T12 T14
C4	T2 T3 T7 T9 T12 T15 T12 T15 T12 T14
C5	T15 T12 T14
C6	T15 T12 T15 T12 T14
C7	T9 T10
C8	T9 T12 T15 T10
C9	T9 T12 T15 T12 T15 T10
C10	T15 T10
C11	T15 T12 T15 T10
C12	T9 T12 T16
C13	T9 T12 T15 T12 T16
C14	T9 T12 T15 T12 T15 T12 T16
C15	T15 T12 T16
C16	T15 T12 T15 T12 T16
C17	T15

3. Generate a set of test paths that cover each IPO_2-df-chain at least once

Test id(s)	Test Paths	IPO_2-df-chains covered by a test path
1	T1 T2 T3 T7 T9 T12 T14 T13 T11 T4	C2
2	T1 T2 T3 T7 T9 T12 T15 T12 T14 T13 T11 T4	C3, C5, C17
3	T1 T2 T3 T7 T9 T12 T15 T12 T15 T12	C4, C5, C6, C17

	T14 T13 T11 T4	
4	T1 T2 T3 T7 T9 T12 T15 T10 T11 T4	C8, C10, C17
5	T1 T2 T3 T7 T9 T12 T15 T12 T15 T10 T11 T4	C9, C10, C11, C17
6	T1 T2 T3 T7 T9 T12 T16 T15 T11 T4	C12, C17
7	T1 T2 T3 T7 T9 T12 T15 T12 T16 T15 T11 T4	C13, C15, C17
8	T1 T2 T3 T7 T9 T12 T15 T12 T15 T12 T16 T15 T11 T4	C14, C15, C16, C17
9	T1 T3 T5 T7 T9 T10 T17 T18 T20 T11 T4	C1, C7

B.5 Test Results after applying STSR and DTSR programs to the test suites in B.4

By applying STSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	1	1	
T8	1	1	
T9	1	1	
T10	1	1	
T14	1	1	
T15	1	1	
T16	1	1	
T20	1	1	

All-uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	1	1	
T8	1	1	
T9	6	1	Test_2, 3, 4, 6, 7
T10	1	1	
T14	2	2	
T15	4	2	Test_6, 7
T16	3	2	Test_7
T20	1	1	

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	0	0	
T8	0	0	
T9	9	1	Test_2, 3, 4, 5, 6, 7, 8, 9
T10	3	3	
T14	3	3	
T15	7	2	Test_4, 5, 6, 7, 8
T16	3	3	
T20	1	1	

By applying DTSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	1	1	
T8	1	1	
T9	1	1	
T10	1	1	
T14	1	1	
T15	1	1	
T16	1	1	
T20	1	1	

All-uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	1	1	
T8	1	1	
T9	6	1	Test_2, 3, 4, 6, 7
T10	1	1	
T14	2	2	
T15	4	2	Test_6, 7
T16	3	2	Test_7
T20	1	1	

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T6	0	0	
T8	0	0	
T9	9	1	Test_2, 3, 4, 5 ,6, 7, 8, 9
T10	3	3	
T14	3	3	
T15	7	3	Test_4, 5, 6, 7
T16	3	3	
T20	1	1	

Appendix C: Vending Machine System

C.1 Requirements of the Vending Machine System

The system sells three products: gum, M&Ms and beer which cost $.35, $.50 and $.80, respectively.

R = {r | r is a requirement which is a **transition**} or

R = {r | r is a requirement which is a **sequence of transitions**}.

R = {$r1$, $r2$, $r3$, $r4$, $r5$} for the Vending Machine EFSM where:

r1: The customer selects an item, deposits coins, and the machine dispenses the item only if it is available and if the payment is sufficient

$r1$ = (T3 or T4) x **T8**

r2: If the item selected is sold out, the machine displays an error message and prompts for item selection.

$r2$ = **T2**

r3: When the customer deposits coins, the machine validates them then displays the total amount inserted.

$r3$ = **T5**

r4: The machine returns the correct change when the customer deposits too much money and when coins for change are available. The machine then displays the change amount and the welcome message.

$r4$ = **T11**

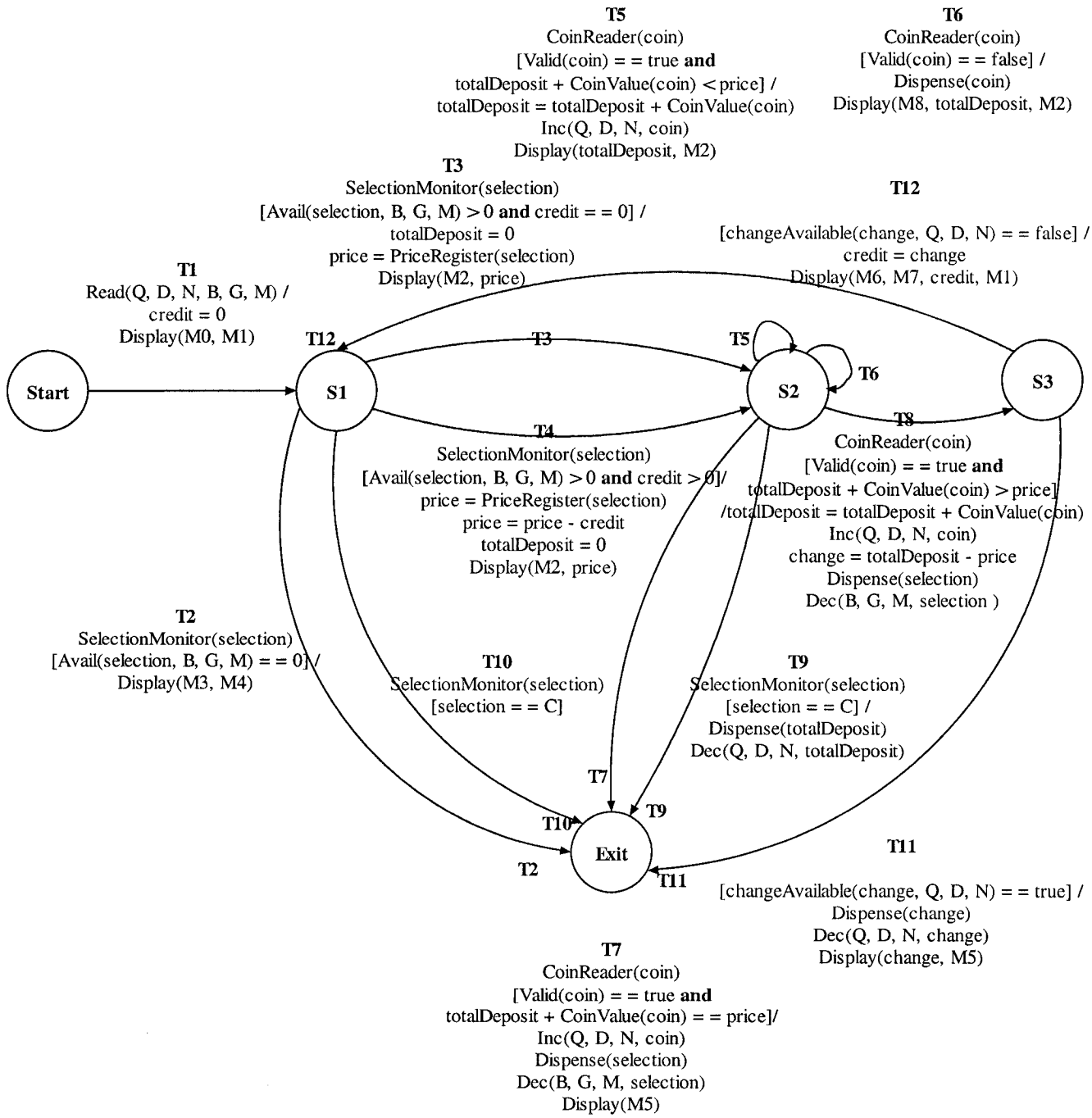
r5: If there are not enough coins for change in the machine, it will show a credit towards the next purchase and prompt for item selection.

$r5$ = **T12**

x : T5 may be inserted1 to 15 times

T# *is the transition under test (TUT)*

C.2 The EFSM Model of the Vending Machine System



Symbol Definitions

- *M0*: message "Welcome! Buy something... Warning: If I do not have enough change I will dispense the product and give a credit for the next purchase".
- *M1*: message "Enter your selection: G, M, B, C to Cancel:".
- *M2*: message "Insert coins: Quarter, Dime, Niquel".
- *M3*: message "Sorry, item is sold out".
- *M4*: message "Make another selection".
- *M5*: message "Thank you!".
- *M6*: message "Sorry! I am out of change".
- *M7*: message "Credit: $".
- *M8*: message "Invalid Coin".
- *Q*: total number of quarters in the machine.
- *N*: total number of nickels in the machine.
- *D*: total number of dimes in the machine.
- *B*: total number of beers in the machine.
- *G*: total number of gums in the machine.
- *M*: total number of M&Ms in the machine.
- *price*: price of the selected item. $price \in \{35, 50, 80\}$
- *selection*: selection of the customer. $selection \in \{B, G, M, C\}$
- *totalDeposit*: total amount of money a customer deposits.
- *coin*: coin inserted by the customer. $coin \in \{Quarter, Dime, Nickel\}$
- *value*: value of the inserted coin. $value \in \{5, 10, 25\}$
- *change*: amount to be returned when the total deposit is greater than the price.

- *credit*: amount of credit available for the customer's next purchase.

Inputs, Outputs, Functions, and Procedures

- *Read(Q, D, N, B, G, M)*: reads initial amounts of *Q, D, N, B, G, M.*
- *SelectionMonitor(selection)*: reads the selection made by the customer.
- *CoinReader(coin)*: reads the inserted *coin.*
- *Display(M)*: displays *M* on the machine's display panel.
- *Dispense(X)*: dispenses *X* to the customer. $X \in$ {*selection, totalDeposit, change, coin*}
- *Avail(selection, B, G, M)*: returns the number of the selected item, *selection*, available in the machine.
- *PriceRegister(selection)*: returns the price of the selected item, *selection.*
- *Valid(coin)*: returns *true* if *coin* $\in$ {*Quarter, Dime, Nickel*}.
- *CoinValue(coin)*: returns 25 or 10 or 5, the value of *coin.*
- *changeAvailable(change, Q, D, N)*: returns *true* if the machine has enough coins to provide the correct change.
- *Inc(Q, D, N, coin)* : According to the type of *coin,* increments *Q, D,* or *N* by 1.
- *Dec(Q, D, N, X)* : According to the value of *X,* decrements *Q, D,* or *N* by the corresponding number of quarters, dimes and nickels in *X*. $X \in$ { *totalDeposit, change*}
- *Dec(B, G, M, selection)* : According to the type of *selection,* decrements *B, G,* or *M* by 1.

States

Start: Idle state.

S1: Waiting for selection.

S2: Accepting coins.

S3: Making change.

Exit.

C.3 Three Test Suites for the Vending Machine System derived from three techniques namely, branch coverage, all-uses coverage and IPO_2-df-chains coverage.

Branch Coverage

Complete Test Suite: {
T1 T3 T5 T6 T8 T12 T4 T7,
T1 T3 T5 T8 T11,
T1 T3 T9,
T1 T10,
T1 T2
}

All-uses Coverage

1. Identify all du-pairs from the EFSM Model

Variables	Def(s)	Use(s)
B	d_{T1}^{B}	$c_{T7}^{B}, c_{T8}^{B}, p_{T2}^{B}, p_{T3}^{B}, p_{T4}^{B}$
	d_{T8}^{B}	$c_{T7}^{B}, c_{T8}^{B}, p_{T2}^{B}, p_{T3}^{B}, p_{T4}^{B}$
change	d_{T8}^{change}	$c_{T11}^{change}, c_{T12}^{change}, p_{T11}^{change}, p_{T12}^{change}$
Credit	d_{T1}^{credit}	$c_{T4}^{credit}, p_{T3}^{credit}, p_{T4}^{credit}$
	d_{T12}^{credit}	$c_{T4}^{credit}, p_{T3}^{credit}, p_{T4}^{credit}$
D	d_{T1}^{D}	$c_{T5}^{D}, c_{T7}^{D}, c_{T8}^{D}, c_{T9}^{D}$
	d_{T5}^{D}	$c_{T5}^{D}, c_{T7}^{D}, c_{T8}^{D}, c_{T9}^{D}$
	d_{T8}^{D}	$c_{T5}^{D}, c_{T7}^{D}, c_{T8}^{D}, c_{T9}^{D}, p_{T11}^{D}, c_{T11}^{D}, p_{T12}^{D}$
G	d_{T1}^{G}	$c_{T7}^{G}, c_{T8}^{G}, p_{T2}^{G}, p_{T3}^{G}, p_{T4}^{G}$
	d_{T8}^{G}	$c_{T7}^{G}, c_{T8}^{G}, p_{T2}^{G}, p_{T3}^{G}, p_{T4}^{G},$
M	d_{T1}^{M}	$c_{T7}^{M}, c_{T8}^{M}, p_{T2}^{M}, p_{T3}^{M}, p_{T4}^{M}$

	d^{M}_{T8}	$c^{M}_{T7}, c^{M}_{T8}, p^{M}_{T2}, p^{M}_{T3}, p^{M}_{T4}$
N	d^{N}_{T1}	$c^{N}_{T5}, c^{N}_{T7}, c^{N}_{T8}, c^{N}_{T9}$
	d^{N}_{T5}	$c^{N}_{T5}, c^{N}_{T7}, c^{N}_{T8}, c^{N}_{T9}$
	d^{N}_{T8}	$c^{N}_{T5}, c^{N}_{T7}, c^{N}_{T8}, c^{N}_{T9}, p^{N}_{T11}, c^{N}_{T11}, p^{N}_{T12}$
Price	d^{price}_{T3}	$c^{price}_{T3}, c^{price}_{T8}, p^{price}_{T5}, p^{price}_{T7}, p^{price}_{T8}$
	d^{price}_{T4}	$c^{price}_{T4}, c^{price}_{T8}, p^{price}_{T5}, p^{price}_{T7}, p^{price}_{T8}$
Q	d^{Q}_{T1}	$c^{Q}_{T5}, c^{Q}_{T7}, c^{Q}_{T8}, c^{Q}_{T9}$
	d^{Q}_{T5}	$c^{Q}_{T5}, c^{Q}_{T7}, c^{Q}_{T8}, c^{Q}_{T9}$
	d^{Q}_{T8}	$c^{Q}_{T5}, c^{Q}_{T7}, c^{Q}_{T8}, c^{Q}_{T9}, c^{Q}_{T11}, p^{Q}_{T11}, p^{Q}_{T12}$
selection	$d^{selection}_{T2}$	$p^{selection}_{T2}$
	$d^{selection}_{T3}$	$c^{selection}_{T3}, c^{selection}_{T7}, c^{selection}_{T8}, p^{selection}_{T3}$
	$d^{selection}_{T4}$	$c^{selection}_{T4}, c^{selection}_{T7}, c^{selection}_{T8}, p^{selection}_{T4}$
	$d^{selection}_{T9}$	$p^{selection}_{T9}$
	$d^{selection}_{T10}$	$p^{selection}_{T10}$
totalDeposit	$d^{totalDeposit}_{T3}$	$c^{totalDeposit}_{T5}, c^{totalDeposit}_{T6}, c^{totalDeposit}_{T8}, c^{totalDeposit}_{T9}, p^{totalDeposit}_{T5}, p^{totalDeposit}_{T7}, p^{totalDeposit}_{T8}$
	$d^{totalDeposit}_{T4}$	$c^{totalDeposit}_{T5}, c^{totalDeposit}_{T6}, c^{totalDeposit}_{T8}, c^{totalDeposit}_{T9}, p^{totalDeposit}_{T5}, p^{totalDeposit}_{T7}, p^{totalDeposit}_{T8}$
	$d^{totalDeposit}_{T5}$	$c^{totalDeposit}_{T5}, c^{totalDeposit}_{T6}, c^{totalDeposit}_{T8}, c^{totalDeposit}_{T9}, p^{totalDeposit}_{T5}, p^{totalDeposit}_{T7}, p^{totalDeposit}_{T8}$

2. Construct an activating path for each du-pair

# DU_Pair	DU-Pairs	Activating Paths
1	$[d^{B}_{T1}, c^{B}_{T7}]$	T1 T3 T5 T7
2	$[d^{B}_{T1}, c^{B}_{T8}]$	T1 T3 T5 T8
3	$[d^{B}_{T1}, p^{B}_{T2}]$	T1 T2
4	$[d^{B}_{T1}, p^{B}_{T3}]$	T1 T3
5	$[d^{B}_{T1}, p^{B}_{T4}]$	T1 T4*

6	$[d^{B}_{T8}, c^{B}_{T7}]$	T8 T12 T4 T7
7	$[d^{B}_{T8}, c^{B}_{T8}]$	T8 T12 T4 T8
8	$[d^{B}_{T8}, p^{B}_{T2}]$	T8 T12 T2
9	$[d^{B}_{T8}, p^{B}_{T3}]$	T8 T12 T3 *
10	$[d^{B}_{T8}, p^{B}_{T4}]$	T8 T12 T4
11	$[d^{change}_{T8}, c^{change}_{T11}]$	T8 T11
12	$[d^{change}_{T8}, c^{change}_{T12}]$	T8 T12
13	$[d^{change}_{T8}, p^{change}_{T11}]$	T8 T11
14	$[d^{change}_{T8}, p^{change}_{T12}]$	T8 T12
15	$[d^{credit}_{T1}, c^{credit}_{T4}]$	T1 T4*
16	$[d^{credit}_{T1}, p^{credit}_{T3}]$	T1 T3
17	$[d^{credit}_{T1}, p^{credit}_{T4}]$	T1 T4*
18	$[d^{credit}_{T12}, c^{credit}_{T4}]$	T12 T4
19	$[d^{credit}_{T12}, c^{credit}_{T12}]$	T12
20	$[d^{credit}_{T12}, p^{credit}_{T3}]$	T12 T3*
21	$[d^{credit}_{T12}, p^{credit}_{T4}]$	T12 T4
22	$[d^{D}_{T1}, c^{D}_{T5}]$	T1 T3 T5
23	$[d^{D}_{T1}, c^{D}_{T7}]$	T1 T3 T7*
24	$[d^{D}_{T1}, c^{D}_{T8}]$	T1 T3 T8*
25	$[d^{D}_{T1}, c^{D}_{T9}]$	T1 T3 T9
26	$[d^{D}_{T5}, c^{D}_{T5}]$	T5 T5
27	$[d^{D}_{T5}, c^{D}_{T7}]$	T5 T7
28	$[d^{D}_{T5}, c^{D}_{T8}]$	T5 T8
29	$[d^{D}_{T5}, c^{D}_{T9}]$	T5 T9
30	$[d^{D}_{T8}, c^{D}_{T5}]$	T8 T12 T4 T5
31	$[d^{D}_{T8}, c^{D}_{T7}]$	T8 T12 T4 T7
32	$[d^{D}_{T8}, c^{D}_{T8}]$	T8 T12 T4 T8
33	$[d^{D}_{T8}, c^{D}_{T9}]$	T8 T12 T4 T9
34	$[d^{D}_{T8}, c^{D}_{T11}]$	T8 T11
35	$[d^{D}_{T8}, p^{D}_{T11}]$	T8 T11
36	$[d^{D}_{T8}, p^{D}_{T12}]$	T8 T12
37	$[d^{G}_{T1}, c^{G}_{T7}]$	T1 T3 T5 T7
38	$[d^{G}_{T1}, c^{G}_{T8}]$	T1 T3 T5 T8

39	$[d^{G}_{T1}, p^{G}_{T2}]$	T1 T2
40	$[d^{G}_{T1}, p^{G}_{T3}]$	T1 T3
41	$[d^{G}_{T1}, p^{G}_{T4}]$	T1 T4*
42	$[d^{G}_{T8}, c^{G}_{T7}]$	T8 T12 T4 T7
43	$[d^{G}_{T8}, c^{G}_{T8}]$	T8 T12 T4 T8
44	$[d^{G}_{T8}, p^{G}_{T2}]$	T8 T12 T2
45	$[d^{G}_{T8}, p^{G}_{T3}]$	T8 T12 T3*
46	$[d^{G}_{T8}, p^{G}_{T4}]$	T8 T12 T4
47	$[d^{M}_{T1}, c^{M}_{T7}]$	T1 T3 T5 T7
48	$[d^{M}_{T1}, c^{M}_{T8}]$	T1 T3 T5 T8
49	$[d^{M}_{T1}, p^{M}_{T2}]$	T1 T2
50	$[d^{M}_{T1}, p^{M}_{T3}]$	T1 T3
51	$[d^{M}_{T1}, p^{M}_{T4}]$	T1 T4*
52	$[d^{M}_{T8}, c^{M}_{T7}]$	T8 T12 T4 T7
53	$[d^{M}_{T8}, c^{M}_{T8}]$	T8 T12 T4 T8
54	$[d^{M}_{T8}, p^{M}_{T2}]$	T8 T12 T2
55	$[d^{M}_{T8}, p^{M}_{T3}]$	T8 T12 T3*
56	$[d^{M}_{T8}, p^{M}_{T4}]$	T8 T12 T4
57	$[d^{N}_{T1}, c^{N}_{T5}]$	T1 T3 T5
58	$[d^{N}_{T1}, c^{N}_{T7}]$	T1 T3 T7*
59	$[d^{N}_{T1}, c^{N}_{T8}]$	T1 T3 T8*
60	$[d^{N}_{T1}, c^{N}_{T9}]$	T1 T3 T9
61	$[d^{N}_{T5}, c^{N}_{T5}]$	T5 T5
62	$[d^{N}_{T5}, c^{N}_{T7}]$	T5 T7
63	$[d^{N}_{T5}, c^{N}_{T8}]$	T5 T8
64	$[d^{N}_{T5}, c^{N}_{T9}]$	T5 T9
65	$[d^{N}_{T8}, c^{N}_{T5}]$	T8 T12 T4 T5
66	$[d^{N}_{T8}, c^{N}_{T7}]$	T8 T12 T4 T7
67	$[d^{N}_{T8}, c^{N}_{T8}]$	T8 T12 T4 T8
68	$[d^{N}_{T8}, c^{N}_{T9}]$	T8 T12 T4 T9
69	$[d^{N}_{T8}, c^{N}_{T11}]$	T8 T11
70	$[d^{N}_{T8}, p^{N}_{T11}]$	T8 T11
71	$[d^{N}_{T8}, p^{N}_{T12}]$	T8 T12

72	$[d_{T3}^{price}, c_{T3}^{price}]$	T3
73	$[d_{T3}^{price}, c_{T8}^{price}]$	T3 T5 T8
74	$[d_{T3}^{price}, p_{T5}^{price}]$	T3 T5
75	$[d_{T3}^{price}, p_{T7}^{price}]$	T3 T5 T7
76	$[d_{T3}^{price}, p_{T8}^{price}]$	T3 T5 T8
77	$[d_{T4}^{price}, c_{T4}^{price}]$	T4
78	$[d_{T4}^{price}, c_{T8}^{price}]$	T4 T8
79	$[d_{T4}^{price}, p_{T5}^{price}]$	T4 T5
80	$[d_{T4}^{price}, p_{T7}^{price}]$	T4 T7
81	$[d_{T4}^{price}, p_{T8}^{price}]$	T4 T8
82	$[d_{T1}^{Q}, c_{T5}^{Q}]$	T1 T3 T5
83	$[d_{T1}^{Q}, c_{T7}^{Q}]$	T1 T3 T7*
84	$[d_{T1}^{Q}, c_{T8}^{Q}]$	T1 T3 T8*
85	$[d_{T1}^{Q}, c_{T9}^{Q}]$	T1 T3 T9
86	$[d_{T5}^{Q}, c_{T5}^{Q}]$	T5 T5
87	$[d_{T5}^{Q}, c_{T7}^{Q}]$	T5 T7
88	$[d_{T5}^{Q}, c_{T8}^{Q}]$	T5 T8
89	$[d_{T5}^{Q}, c_{T9}^{Q}]$	T5 T9
90	$[d_{T8}^{Q}, c_{T5}^{Q}]$	T8 T12 T4 T5
91	$[d_{T8}^{Q}, c_{T7}^{Q}]$	T8 T12 T4 T7
92	$[d_{T8}^{Q}, c_{T8}^{Q}]$	T8 T12 T4 T8
93	$[d_{T8}^{Q}, c_{T9}^{Q}]$	T8 T12 T4 T9
94	$[d_{T8}^{Q}, c_{T11}^{Q}]$	T8 T11
95	$[d_{T8}^{Q}, p_{T11}^{Q}]$	T8 T11
96	$[d_{T8}^{Q}, p_{T12}^{Q}]$	T8 T12
97	$[d_{T2}^{selection}, p_{T2}^{selection}]$	T2
98	$[d_{T3}^{selection}, c_{T3}^{selection}]$	T3
99	$[d_{T3}^{selection}, c_{T7}^{selection}]$	T3 T5 T7
100	$[d_{T3}^{selection}, c_{T8}^{selection}]$	T3 T5 T8
101	$[d_{T3}^{selection}, p_{T3}^{selection}]$	T3
102	$[d_{T4}^{selection}, c_{T4}^{selection}]$	T4
103	$[d_{T4}^{selection}, c_{T7}^{selection}]$	T4 T7
104	$[d_{T4}^{selection}, c_{T8}^{selection}]$	T4 T8

105	$[d^{selection}_{T4}, p^{selection}_{T4}]$	T4
106	$[d^{selection}_{T9}, p^{selection}_{T9}]$	T9
107	$[d^{selection}_{T10}, p^{selection}_{T10}]$	T10
108	$[d^{totalDeposit}_{T3}, c^{totalDeposit}_{T5}]$	T3 T5
109	$[d^{totalDeposit}_{T3}, c^{totalDeposit}_{T6}]$	T3 T6
110	$[d^{totalDeposit}_{T3}, c^{totalDeposit}_{T8}]$	T3 T8*
111	$[d^{totalDeposit}_{T3}, c^{totalDeposit}_{T9}]$	T3 T9
112	$[d^{totalDeposit}_{T3}, p^{totalDeposit}_{T5}]$	T3 T5
113	$[d^{totalDeposit}_{T3}, p^{totalDeposit}_{T7}]$	T3 T7*
114	$[d^{totalDeposit}_{T3}, p^{totalDeposit}_{T8}]$	T3 T8*
115	$[d^{totalDeposit}_{T4}, c^{totalDeposit}_{T5}]$	T4 T5
116	$[d^{totalDeposit}_{T4}, c^{totalDeposit}_{T6}]$	T4 T6
117	$[d^{totalDeposit}_{T4}, c^{totalDeposit}_{T8}]$	T4 T8
118	$[d^{totalDeposit}_{T4}, c^{totalDeposit}_{T9}]$	T4 T9
119	$[d^{totalDeposit}_{T4}, p^{totalDeposit}_{T5}]$	T4 T5
120	$[d^{totalDeposit}_{T4}, p^{totalDeposit}_{T7}]$	T4 T7
121	$[d^{totalDeposit}_{T4}, p^{totalDeposit}_{T8}]$	T4 T8
122	$[d^{totalDeposit}_{T5}, c^{totalDeposit}_{T5}]$	T5 T5
123	$[d^{totalDeposit}_{T5}, c^{totalDeposit}_{T6}]$	T5 T6
124	$[d^{totalDeposit}_{T5}, c^{totalDeposit}_{T8}]$	T5 T8
125	$[d^{totalDeposit}_{T5}, c^{totalDeposit}_{T9}]$	T5 T9
126	$[d^{totalDeposit}_{T5}, p^{totalDeposit}_{T5}]$	T5 T5
127	$[d^{totalDeposit}_{T5}, p^{totalDeposit}_{T7}]$	T5 T7
128	$[d^{totalDeposit}_{T5}, p^{totalDeposit}_{T8}]$	T5 T8

* means an activating path is infeasible.

3. Generate test paths to cover all activating paths

Test id(s)	Test Paths	DU-pairs covered by a test path
1	T1 T2	3, 39, 49
2	T1 T10	107
3	T1 T3 T9	4, 16, 25, 40, 50, 60, 85, 111
4	T1 T3 T5 T7	1, 4, 16, 22, 27, 37, 40, 47, 50, 57, 62, 74, 75, 82, 87, 99, 108, 112, 127
5	T1 T3 T5 T8 T12 T4 T9	2, 4, 10, 12, 14, 16, 18, 21, 22, 28, 33, 36, 38, 40, 46, 48, 50, 56, 57, 63, 68, 71, 73,

		74, 76, 82, 88, 93, 96, 100, 108, 112, 118, 124, 128
6	T1 T3 T5 T8 T12 T4 T5 T9	2, 4, 10, 12, 14, 16, 18, 21, 22, 28, 29, 30, 36, 38, 40, 46, 48, 50, 56, 57, 63, 64, 65, 71, 73, 74, 76, 79, 82, 88, 89, 90, 96, 100, 108, 112, 115, 119, 124, 125, 128
7	T1 T3 T5 T8 T12 T4 T6 T8 T11	2, 4, 10, 11, 12, 13, 14, 16, 18, 21 , 22, 28, 34, 35, 36, 38, 40, 46, 48, 50, 56, 57, 63, 69, 70, 71, 73, 74, 76, 82, 88, 94, 95, 96, 100, 108, 112, 116, 124, 128
8	T1 T3 T5 T8 T12 T4 T8 T12 T2	2, 4, 7, 8, 10, 12, 14, 16, 18, 21, 22, 28, 32, 36, 38, 40, 43, 44, 46, 48, 50 , 53, 54, 56, 57, 63, 67, 71, 73, 74, 76, 78, 81, 82, 88, 92, 96, 100, 104, 108, 112, 117, 121, 124, 128
9	T1 T3 T6 T5 T5 T6 T8 T12 T4 T7	4, 6, 10, 12, 14, 16, 18, 21, 26, 31, 36, 40, 42, 46, 50, 52, 56, 61, 66, 76, 80, 86, 91, 96, 103, 109, 120, 122, 123, 126

IPO_2-df-chains Coverage

1. Identify all ipo-pairs and their IPO_2-df-chains from the EFSM model

#	ipo-pairs	IPO_2-df-chains	
1	$[d^{B}_{T1}, u^{B}_{T2}]$	$\{d^{B}_{T1}, u^{B}_{T2}\}$	C1
2	$[d^{selection}_{T3}, u^{B}_{T2}]$	$\{d^{selection}_{T3}, u^{selection}_{T8}, d^{B}_{T8}, u^{B}_{T2}\}$	C2
3	$[d^{selection}_{T4}, u^{B}_{T2}]$	$\{d^{selection}_{T4}, u^{selection}_{T8}, d^{B}_{T8}, u^{B}_{T2}\}$	C3
4	$[d^{G}_{T1}, u^{G}_{T2}]$	$\{d^{G}_{T1}, u^{G}_{T2}\}$	C4
5	$[d^{selection}_{T3}, u^{G}_{T2}]$	$\{d^{selection}_{T3}, u^{selection}_{T8}, d^{G}_{T8}, u^{G}_{T2}\}$	C5
6	$[d^{selection}_{T4}, u^{G}_{T2}]$	$\{d^{selection}_{T4}, u^{selection}_{T8}, d^{G}_{T8}, u^{G}_{T2}\}$	C6
7	$[d^{M}_{T1}, u^{M}_{T2}]$	$\{d^{M}_{T1}, u^{M}_{T2}\}$	C7
8	$[d^{selection}_{T3}, u^{M}_{T2}]$	$\{d^{selection}_{T3}, u^{selection}_{T8}, d^{M}_{T8}, u^{M}_{T2}\}$	C8
9	$[d^{selection}_{T4}, u^{M}_{T2}]$	$\{d^{selection}_{T4}, u^{selection}_{T8}, d^{M}_{T8}, u^{M}_{T2}\}$	C9
10	$[d^{selection}_{T3}, u^{price}_{T4}]$	$\{d^{selection}_{T3}, u^{selection}_{T3}, d^{price}_{T3}, u^{price}_{T8}, d^{change}_{T8}, u^{change}_{T12}, d^{credit}_{T12}, u^{credit}_{T4}, d^{price}_{T4}, u^{price}_{T4}\}$	C10
11	$[d^{totalDeposit}_{T3}, u^{price}_{T4}]$	$\{d^{totalDeposit}_{T3}, u^{totalDeposit}_{T8}, d^{change}_{T8}, u^{change}_{T12}, d^{credit}_{T12}, u^{credit}_{T4}, d^{price}_{T4}, u^{price}_{T4}\}$	C11
		$\{d^{totalDeposit}_{T3}, u^{totalDeposit}_{T5}, d^{totalDeposit}_{T5}, u^{totalDeposit}_{T8}, d^{change}_{T8}, u^{change}_{T12}, d^{credit}_{T12}, u^{credit}_{T4}, d^{price}_{T4}, u^{price}_{T4}\}$	C12

		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C13
12	$[d_{T4}^{selection}, u_{T4}^{price}]$	$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C14
13	$[d_{T4}^{totalDeposit}, u_{T4}^{price}]$	$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C15
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C16
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C17
		$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C18
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C19
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C20
14	$[d_{T5}^{coin}, u_{T4}^{price}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C21
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C22
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C23
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C24

		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C25
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C26
15	$[d_{T8}^{coin}, u_{T4}^{price}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C27
		$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T4}^{price}\}$	C28
16	$[d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}]$	$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}\}$	C29
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}\}$	C30
17	$[d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}]$	$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}\}$	C31
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}\}$	C32
18	$[d_{T5}^{coin}, u_{T5}^{totalDeposit}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}\}$	C33
19	$[d_{T3}^{totalDeposit}, u_{T6}^{totalDeposit}]$	$\{d_{T3}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C34
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C35
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C36
20	$[d_{T4}^{totalDeposit}, u_{T6}^{totalDeposit}]$	$\{d_{T4}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C37
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C38
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C39
21	$[d_{T5}^{coin}, u_{T6}^{totalDeposit}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C40
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C41
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T6}^{totalDeposit}\}$	C42

22	$[d_{T3}^{selection}, u_{T7}^{price}]$	$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T7}^{price}\}$	C43
		$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C44
23	$[d_{T4}^{selection}, u_{T7}^{price}]$	$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T7}^{price}\}$	C45
		$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C46
24	$[d_{T3}^{totalDeposit}, u_{T7}^{price}]$	$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C47
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C48
25	$[d_{T4}^{totalDeposit}, u_{T7}^{price}]$	$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C49
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C50
26	$[d_{T5}^{coin}, u_{T7}^{price}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C51
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C52
27	$[d_{T8}^{coin}, u_{T7}^{price}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T7}^{price}\}$	C53
28	$[d_{T3}^{totalDeposit}, u_{T7}^{totalDeposit}]$	$\{d_{T3}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C54
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C55
29	$[d_{T4}^{totalDeposit}, u_{T7}^{totalDeposit}]$	$\{d_{T4}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C56
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C57
30	$[d_{T5}^{coin}, u_{T7}^{totalDeposit}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C58
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T7}^{totalDeposit}\}$	C59
31	$[d_{T3}^{selection}, u_{T11}^{change}]$	$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C60
		$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C61
32	$[d_{T3}^{totalDeposit}, u_{T11}^{change}]$	$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C62

		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C63
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C64
		$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C65
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C66
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C67
		$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C68
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C69
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C70
33	$[d_{T4}^{selection}, u_{T11}^{change}]$	$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C71
		$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C72
34	$[d_{T4}^{totalDeposit}, u_{T11}^{change}]$	$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C73
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C74
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C75
		$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C76
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C77

		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C78
		$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C79
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C80
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C81
35	$[d_{T5}^{coin}, u_{T11}^{change}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C82
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C83
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C84
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C85
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C86
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C87
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C88
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C89

		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C90
36	$[d_{T8}^{coin}, u_{T11}^{change}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T11}^{change}\}$	C91
		$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C92
		$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T11}^{change}\}$	C93
37	$[d_{T1}^{D}, u_{T11}^{D}]$	$\{d_{T1}^{D}, u_{T8}^{D}, d_{T8}^{D}, u_{T11}^{D}\}$	C94
		$\{d_{T1}^{D}, u_{T5}^{D}, d_{T5}^{D}, u_{T8}^{D}, d_{T8}^{D}, u_{T11}^{D}\}$	C95
38	$[d_{T5}^{coin}, u_{T11}^{D}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{D}, u_{T8}^{D}, d_{T8}^{D}, u_{T11}^{D}\}$	C96
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{D}, u_{T5}^{D}, d_{T5}^{D}, u_{T8}^{D}, d_{T8}^{D}, u_{T11}^{D}\}$	C97
39	$[d_{T8}^{coin}, u_{T11}^{D}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{D}, u_{T11}^{D}\}$	C98
40	$[d_{T1}^{N}, u_{T11}^{N}]$	$\{d_{T1}^{N}, u_{T8}^{N}, d_{T8}^{N}, u_{T11}^{N}\}$	C99
		$\{d_{T1}^{N}, u_{T5}^{N}, d_{T5}^{N}, u_{T8}^{N}, d_{T8}^{N}, u_{T11}^{N}\}$	C100
41	$[d_{T5}^{coin}, u_{T11}^{N}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{N}, u_{T8}^{N}, d_{T8}^{N}, u_{T11}^{N}\}$	C101
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{N}, u_{T5}^{N}, d_{T5}^{N}, u_{T8}^{N}, d_{T8}^{N}, u_{T11}^{N}\}$	C102
42	$[d_{T8}^{coin}, u_{T11}^{N}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{Q}, u_{T11}^{N}\}$	C103
43	$[d_{T1}^{Q}, u_{T11}^{Q}]$	$\{d_{T1}^{Q}, u_{T8}^{Q}, d_{T8}^{Q}, u_{T11}^{Q}\}$	C104
		$\{d_{T1}^{Q}, u_{T5}^{Q}, d_{T5}^{Q}, u_{T8}^{Q}, d_{T8}^{Q}, u_{T11}^{Q}\}$	C105
44	$[d_{T5}^{coin}, u_{T11}^{Q}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{Q}, u_{T8}^{Q}, d_{T8}^{Q}, u_{T11}^{Q}\}$	C106
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{Q}, u_{T5}^{Q}, d_{T5}^{Q}, u_{T8}^{Q}, d_{T8}^{Q}, u_{T11}^{Q}\}$	C107
45	$[d_{T8}^{coin}, u_{T11}^{Q}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{Q}, u_{T11}^{Q}\}$	C108
46	$[d_{T1}^{credit}, u_{T12}^{credit}]$	$\{d_{T1}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C109
47	$[d_{T3}^{selection}, u_{T12}^{credit}]$	$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C110
		$\{d_{T3}^{selection}, u_{T3}^{selection}, d_{T3}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C111
48	$[d_{T3}^{totalDeposit}, u_{T12}^{credit}]$	$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C112

		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C113
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C114
		$\{d_{T3}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C115
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C116
		$\{d_{T3}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C117
49	$[d_{T4}^{selection}, u_{T12}^{credit}]$	$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C118
		$\{d_{T4}^{selection}, u_{T4}^{selection}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C119
50	$[d_{T4}^{totalDeposit}, u_{T12}^{credit}]$	$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C120
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C121
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C122
		$\{d_{T4}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C123
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C124
		$\{d_{T4}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C125
51	$[d_{T5}^{coin}, u_{T12}^{credit}]$	$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C126
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C127

		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C128
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C129
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C130
		$\{d_{T5}^{coin}, u_{T5}^{coin}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T5}^{totalDeposit}, d_{T5}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C131
	$[d_{T8}^{coin}, u_{T12}^{credit}]$	$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C132
		$\{d_{T8}^{coin}, u_{T8}^{coin}, d_{T8}^{totalDeposit}, u_{T8}^{totalDeposit}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T4}^{credit}, d_{T4}^{price}, u_{T8}^{price}, d_{T8}^{change}, u_{T12}^{change}, d_{T12}^{credit}, u_{T12}^{credit}\}$	C133

2. Construct an activating path for each IPO_2-df-chain

#	Activating Path
C1	T1 T2
C2	T3 T5 T8 T12 T2
C3	T4 T8 T12 T2
C4	T1 T2
C5	T3 T5 T8 T12 T2
C6	T4 T8 T12 T2
C7	T1 T2
C8	T3 T5 T8 T12 T2
C9	T4 T8 T12 T2
C10	T3 T5 T8 T12 T4
C11	T3 T8 T12 T4*
C12	T3 T5 T8 T12 T4
C13	T3 T5 T5 T8 T12 T4
C14	T4 T8 T12 T4
C15	T4 T8 T12 T4
C16	T4 T5 T8 T12 T4
C17	T4 T5 T5 T8 T12 T4

C18	T4 T8 T12 T4 T8 T12 T4
C19	T4 T5 T8 T12 T4 T8 T12 T4
C20	T4 T5 T5 T8 T12 T4 T8 T12 T4
C21	T5 T8 T12 T4
C22	T5 T5 T8 T12 T4
C23	T5 T5 T5 T8 T12 T4
C24	T5 T8 T12 T4 T8 T12 T4
C25	T5 T5 T8 T12 T4 T8 T12 T4
C26	T5 T5 T5 T8 T12 T4 T8 T12 T4
C27	T8 T12 T4
C28	T8 T12 T4 T8 T12 T4
C29	T3 T5
C30	T3 T5 T5
C31	T4 T5
C32	T4 T5 T5
C33	T5 T5
C34	T3 T6
C35	T3 T5 T6
C36	T3 T5 T5 T6
C37	T4 T6
C38	T4 T5 T6
C39	T4 T5 T5 T6
C40	T5 T6
C41	T5 T5 T6
C42	T5 T5 T5 T6
C43	T3 T5 T7
C44	T3 T5 T8 T12 T4 T7
C45	T4 T7
C46	T4 T8 T12 T4 T7
C47	T3 T8 T12 T4 T7*
C48	T3 T5 T8 T12 T4 T7
C49	T4 T8 T12 T4 T7
C50	T4 T5 T8 T12 T4 T7
C51	T5 T8 T12 T4 T7
C52	T5 T5 T8 T12 T4 T7
C53	T8 T12 T4 T7
C54	T3 T7*
C55	T3 T5 T7
C56	T4 T7
C57	T4 T5 T7
C58	T5 T7
C59	T5 T5 T7
C60	T3 T5 T8 T11
C61	T3 T5 T8 T12 T4 T8 T11
C62	T3 T8 T11*

C63	T3 T5 T8 T11
C64	T3 T5 T5 T8 T11
C65	T3 T8 T12 T4 T8 T11*
C66	T3 T5 T8 T12 T4 T8 T11
C67	T3 T5 T5 T8 T12 T4 T8 T11
C68	T3 T8 T12 T4 T8 T12 T4 T8 T11*
C69	T3 T5 T8 T12 T4 T8 T12 T4 T8 T11
C70	T3 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11
C71	T4 T8 T11
C72	T4 T8 T12 T4 T8 T11
C73	T4 T8 T11
C74	T4 T5 T8 T11
C75	T4 T5 T5 T8 T11
C76	T4 T8 T12 T4 T8 T11
C77	T4 T5 T8 T12 T4 T8 T11
C78	T4 T5 T5 T8 T12 T4 T8 T11
C79	T4 T8 T12 T4 T8 T12 T4 T8 T11
C80	T4 T5 T8 T12 T4 T8 T12 T4 T8 T11
C81	T4 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11
C82	T5 T8 T11
C83	T5 T5 T8 T11
C84	T5 T5 T5 T8 T11
C85	T5 T8 T12 T4 T8 T11
C86	T5 T5 T8 T12 T4 T8 T11
C87	T5 T5 T5 T8 T12 T4 T8 T11
C88	T5 T8 T12 T4 T8 T12 T4 T8 T11
C89	T5 T5 T8 T12 T4 T8 T12 T4 T8 T11
C90	T5 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11
C91	T8 T11
C92	T8 T12 T4 T8 T11
C93	T8 T12 T4 T8 T12 T4 T8 T11
C94	T1 T3 T8 T11*
C95	T1 T3 T5 T8 T11
C96	T5 T8 T11
C97	T5 T5 T8 T11
C98	T8 T11
C99	T1 T3 T8 T11*
C100	T1 T3 T5 T8 T11
C101	T5 T8 T11
C102	T5 T5 T8 T11
C103	T8 T11
C104	T1 T3 T8 T11*
C105	T1 T3 T5 T8 T11
C106	T5 T8 T11
C107	T5 T5 T8 T11

C108	T8 T11
C109	T1 T4 T8 T12*
C110	T3 T5 T8 T12
C111	T3 T5 T8 T12 T4 T8 T12
C112	T3 T8 T12*
C113	T3 T5 T8 T12
C114	T3 T5 T5 T8 T12
C115	T3 T8 T12 T4 T8 T12*
C116	T3 T5 T8 T12 T4 T8 T12
C117	T3 T5 T5 T8 T12 T4 T8 T12
C118	T4 T8 T12
C119	T4 T8 T12 T4 T8 T12
C120	T4 T8 T12
C121	T4 T5 T8 T12
C122	T4 T5 T5 T8 T12
C123	T4 T8 T12 T4 T8 T12
C124	T4 T5 T8 T12 T4 T8 T12
C125	T4 T5 T5 T8 T12 T4 T8 T12
C126	T5 T8 T12
C127	T5 T5 T8 T12
C128	T5 T5 T5 T8 T12
C129	T5 T8 T12 T4 T8 T12
C130	T5 T5 T8 T12 T4 T8 T12
C131	T5 T5 T5 T8 T12 T4 T8 T12
C132	T8 T12
C133	T8 T12 T4 T8 T12

* means an activating path is infeasible.

3. Generate a set of test paths that cover each IPO_2-df-chain at least once

Test id(s)	Test Paths	IPO_2-df-chains covered by a test path
1	T1 T2	C1, C4, C7
2	T1 T3 T5 T8 T12 T2	C2, C5, C8, C29, C110, C113, C126, C132
3	T1 T3 T5 T5 T8 T12 T4 T5 T8 T12 T4 T8 T12 T4 T8 T12 T2	C3, C6, C9, C13, C14, C15, 16, C19, C21, C22, C24, C27, C28, C29, C30, C31, C33, C114, C118, C119, C120, C121, C123, C124, C126, C127, C129, C132, C133
4	T1 T3 T5 T7	C29, C43, C55, C58
5	T1 T3 T5 T8 T12 T4 T7	C10, C12, C21, C27, C29, C44, C45, C48, C51, C53, C56, C110, C113, C126, C132
6	T1 T3 T5 T8 T12 T4 T8 T12 T4 T7	C10, C12, C14, C15, C21, C24,

		C27, C28, C29, C45, C46, C49, C53, C56, C110, C111, C113, C116, C118, C120, C126, C129, C132, C133
7	T1 T3 T5 T8 T12 T4 T5 T8 T12 T4 T7	C10, C12, C16, C21, C27, C29, C31, C45, C50, C51, C53, C56, C110, C113, C121, C126, C132
8	T1 T3 T5 T8 T12 T4 T5 T5 T8 T12 T4 T5 T5 T8 T12 T4 T5 T7	C10, C12, C17, C21, C22, C27, C29, C31, C32, C33, C57, C58, C110, C113, C122, C126, C127, C132
9	T1 T3 T5 T8 T12 T4 T8 T12 T4 T8 T12 T4 T8 T12 T4 T5 T8 T12 T4 T7	C10, C12, C14, C15, C16, C18, C21, C24, C27, C28, C29, C31, C45, C50, C51, C53, C56, C110, C111, C113, C116, C118, C119, C120, C121, C123, C126, C129, C132, C133
10	T1 T3 T5 T5 T8 T12 T4 T8 T12 T4 T5 T8 T12 T4 T8 T12 T4 T5 T6 T5 T5 T7	C13, C14, C15, C16, C19, C21, C22, C24, C25, C27, C28, C29, C30, C31, C33, C38, C40, C58, C59, C114, C117, C118, C120, C121, C124, C126, C127, C129, C130, C132, C133
11	T1 T3 T5 T8 T12 T4 T5 T5 T8 T12 T4 T8 T12 T4 T6 T5 T5 T8 T12 T4 T7	C10, C12, C14, C15, C17, C20, C21, C22, C24, C25, C27, C28, C29, C31, C32, C33, C37, C45, C51, C52, C53, C56, C110, C113, C118, C120, C122, C126, C127, C129, C130, C132, C133
12	T1 T3 T5 T8 T11	C29, C60, C63, C82, C91, C95, C96, C98, C100, C101, C103, C105, C106, C108
13	T1 T3 T5 T5 T8 T11	C29, C30, C33, C64, C82, C83, C91, C96, C97, C98, C101, C102, C103, C106, C107, C108
14	T1 T3 T5 T8 T12 T4 T8 T11	C10, C12, C21, C27, C29, C61, C66, C71, C73, C85, C91, C92, C98, C103, C108, C110, C113, C126, C132
15	T1 T3 T5 T5 T8 T12 T4 T8 T11	C13, C21, C22, C27, C29, C30, C33, C67, C71, C73, C85, C86, C91, C92, C98, C103, C108, C114, C126, C127, C132
16	T1 T3 T5 T8 T12 T4 T5 T8 T11	C10, C12, C21, C27, C29, C31, C74, C82, C91, C96, C98, C101, C103, C106, C108, C110, C113,

		C126, C132
17	T1 T3 T5 T8 T12 T4 T5 T5 T8 T11	C10, C12, C21, C27, C29, C31, C32, C33, C75, C82, C83, C91, C96, C97, C98, C101, C102, C103, C106, C107, C108, C110, C113, C126, C132
18	T1 T3 T5 T5 T5 T8 T12 T4 T8 T11	C21, C22, C23, C27, C29, C30, C33, C71, C73, C85, C86, C87, C91, C92, C98, C103, C108, C126, C127, C128, C132
19	T1 T3 T5 T8 T12 T4 T8 T12 T4 T8 T11	C10, C12, C14, C15, C21, C24, C27, C28, C29, C69, C71, C72, C73, C76, C88, C91, C92, C93, C98, C103, C108, C110, C111, C113, C116, C118, C120, C126, C129, C132, C133
20	T1 T3 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11	C13, C14, C15, C21, C22, C24, C25, C27, C28, C29, C30, C33, C70, C71, C72, C73, C76, C88, C89, C91, C92, C93, C98, C103, C108, C114, C117, C118, C120, C126, C127, C129, C130, C132, C133
21	T1 T3 T5 T8 T12 T4 T5 T8 T12 T4 T8 T11	C10, C12, C16, C21, C27, C29, C31, C71, C73, C77, C85, C91, C92, C98, C103, C108, C110, C113, C121, C126, C132
22	T1 T3 T5 T8 T12 T4 T5 T5 T8 T12 T4 T8 T11	C10, C12, C17, C21, C22, C27, C29, C31, C32, C33, C71, C73, C78, C85, C86, C91, C92, C98, C103, C108, C110, C113, C122, C126, C127, C132
23	T1 T3 T5 T8 T12 T4 T8 T12 T4 T8 T12 T4 T8 T11	C10, C12, C14, C15, C18, C21, C24, C27, C28, C29, C71, C72, C73, C76, C79, C91, C92, C93, C98, C103, C108, C110, C111, C113, C116, C118, C119, C120, C123, C126, C129, C132, C133
24	T1 T3 T5 T6 T5 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11	C14, C21, C22, C23, C24, C25, C26, C27, C28, C29, C33, C35, C40, C71, C72, C73, C76, C88, C89, C90, C91, C92, C93, C98, C103, C108, C118, C120, C126, C127, C128, C129, C130, C131, C132, C133
25	T1 T3 T5 T8 T12 T4 T5 T8 T12 T4 T8	C10, C12, C14, C15, C16, C19,

	T12 T4 T8 T11	C21, C24, C27, C28, C29, C31, C71, C72, C73, C76, C80, C88, C91, C92, C93, C98, C103, C108, C110, C113, C118, C120, C121, C124, C126, C129, C132, C133
26	T1 T3 T5 T5 T6 T8 T12 T4 T5 T5 T8 T12 T4 T8 T12 T4 T8 T11	C10, C14, C15, C17, C20, C21, C22, C24, C25, C27, C28, C29, C30, C31, C32, C33, C36, C40, C41, C71, C72, C73, C76, C81, C88, C89, C91, C92, C93, C98, C103, C108, C118, C120, C122, C126, C127, C129, C130, C132, C133
27	T1 T3 T6 T5 T5 T5 T6 T5 T5 T5 T8 T12 T4 T5 T5 T6 T5 T5 T5 T8 T11	C21, C22, C23, C27, C31, C32, C33, C34, C39, C40, C41, C42, C82, C83, C84, C91, C96, C97, C98, C101, C102, C103, C106, C107, C108, C126, C127, C128, C132

C.4 Test Results after applying STSR and DTSR programs to the test suites in C.3

By applying STSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	1	1	
T5	2	1	Test_2
T8	2	1	Test_2
T11	1	1	
T12	1	1	

All–uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	2	2	
T5	6	3	Test_5, Test_7, Test_8
T8	5	3	Test_6, Test_8
T11	1	1	
T12	5	3	Test_6, Test_7

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	3	3	
T5	26	6	Test_4, 5, 6, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
T8	25	6	Test_5, 8, 9, 10, 11, 12, 14, 16, 17, 18, 19, 20, 21, 22, 3, 24, 25, 26, 27
T11	16	6	Test_18, 19, 20, 21, 22, 23, 24, 25, 26, 27
T12	23	6	Test_5, 8, 9, 10, 11, 14, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27

By applying DTSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	1	1	
T5	2	1	Test_2
T8	2	1	Test_2
T11	1	1	
T12	1	1	

All-uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	2	2	
T5	6	3	Test_4, Test_6, Test_7
T8	5	3	Test_5, Test_7
T11	1	1	
T12	5	3	Test_5, Test_6

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	3	3	
T5	26	13	Test_4, 5, 6, 12, 14, 15, 16, 19, 20, 21, 22, 23, 25
T8	25	21	Test_5, 12, 14, 16
T11	16	16	
T12	23	17	Test_5, 14, 16, 17, 19, 21

Appendix D: Cruise Control System

D.1 Requirements of the Cruise Control System (CCS)

A Cruise Control System (CCS) maintains the speed of a car at a pre-selected value. The CCS calculates the required output to be passed on to the actuator and maintains the cruise speed by controlling the throttle. Physically the system consists of a control module, a sensor that measures wheel revolutions, the accelerator, the brake, and a control panel with all the displays and buttons to inform the user. There are six control buttons on the control panel: On, Off, Set, Resume, Accel and Coast. There are two displays on the control panel: the current speed and the cruise speed.

R = {r | r is a requirement which is a **transition**} or

R = {r | r is a requirement which is a **sequence of transitions**}.

R = {$r1$, $r2$, $r3$, $r4$, $r5$, $r6$, $r7$, $r8$, $r9$, $r10$, $r11$} for the CCS EFSM where:

r1: When the CCS is on, the user can set the cruise speed as the current speed by using the *Set* button. The CCS is now activated. The CCS will store the current speed as the new set value for the cruise speed. The current speed must be at least 40km/h. The CCS displays the cruise speed set. The unit used in the speed display is km/h.

$r1$ = **T2**

r2: The CCS measures the current speed every 2 seconds and shows it on the control panel's speed display. The speed is measured by a sensor detecting wheel revolutions of the car.

$r2$ = **T3**

r3: If the current speed is greater than the cruise speed, the cruise control calculates the required speed value to decrease the current speed.

*r*3 = **T4**

r4: If the current speed is smaller than the cruise speed, the cruise control calculates the required speed value to increase the current speed.

*r*4 = **T5**

r5: If the current speed drops below 35km/h, the CCS is suspended automatically.

*r*5 = **T8**

r6: The user can increase the cruise speed using the *Accel* button. Every hit on the *Accel* button will increase the cruise speed by 1km/h.

*r*6 = **T6**

r7: The user can decrease the cruise speed by using the *Coast* button. Every hit on the *Coast* button will decrease the cruise speed by 1km/h.

*r*7 = **T7**

r8: When the user presses the *brake* pedal, the CCS is suspended and the car speed changes according to the manual control. Cruise control may be reactivated by using the *Resume* button.

*r*8 = **T9**

r9: When the user presses the *accelerator* pedal, the CCS is suspended and the car speed changes according to the manual control. The CCS may be reactivated by using the *Resume* button.

*r*9 = **T10**

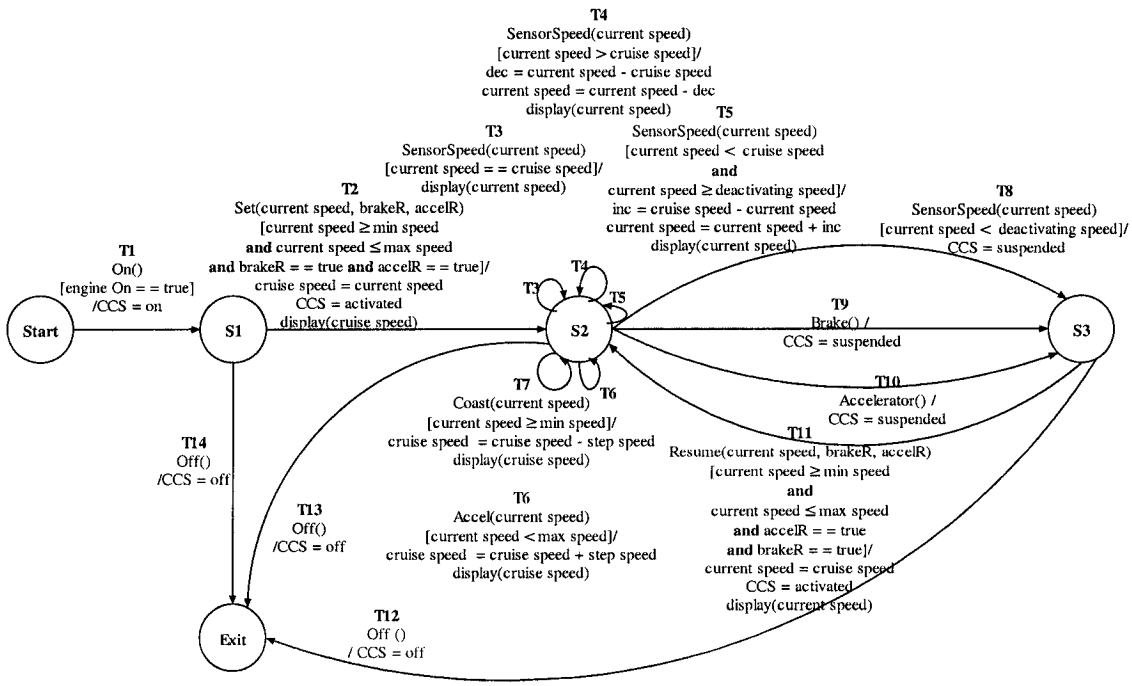
***r*10**: The *Resume* button reactivates the CCS and the current speed is reset to the last set cruise speed.

*r*10 = **T11**

***r*11**: While CCS is activated, the user turns the CCS off by using the *Off* button.

*r*11 = **T13**

D.2 The EFSM Model of the CCS



Symbol Definition

System inputs

- *On*(): the *On* button is pressed by the driver.
- *Off*(): the *Off* button is pressed by the driver.
- *Set*(*current speed*, *brakeR*, *accelR*): the *Set* button is pressed by the driver.

- *Resume*(*current speed*, *brakeR*, *accelR*): the *Resume* button is pressed by the driver.
- *SensorSpeed*(*current speed*): the CCS reads the current speed of the car.
- *Accel*(*current speed*): the *Accel* button is pressed by the driver.
- *Coast*(*current speed*): the *Coast* button is pressed by the driver.
- *Brake*(): the brake pedal is pressed by the driver.
- *Accelerator*(): the accelerator pedal is pressed by the driver.

System outputs

- *current speed*: current driving speed of the car.
- *cruise speed*: speed stored in the controller to be maintained.

System variables

- *CCS*: status of the CCS. *CCS* ∈ {*on, off, activated, suspended*}
- *dec*: amount of speed to be decreased by the accelerator to reach the cruise speed.
- *inc*: amount of speed to be increased by the accelerator to reach the cruise speed.
- *brakeR*: Boolean variable indicating that the brake is released if set to true.
- *accelR*: Boolean variable indicating that the accelerator is released if set to true.
- *engine On*: Boolean variable indicating that the engine is On if set to true.
- *current speed*
- *cruise speed*

Constants

- *max speed*: maximum cruise speed. *max speed* is 180 km/h.
- *min speed*: minimum cruise speed. *min speed* is 40 km/h.

- *deactivating speed*: speed that suspends the CCS automatically. *deactivating speed* is 35 km/h.
- *step speed*: *step speed* is 1 km/h.

States

Start: the CCS is Off.

S1: Ready. The CCS is On and not yet activated. It is waiting for the driver to activate it by setting a cruise speed using the *Set* button.

S2: Set. A cruise speed is set and the CCS is maintaining it. The CCS is now activated.

S3: Wait. The CCS is suspended and there is a cruise speed set. The CCS is waiting until the driver presses the *Resume* button to reactivate the CCS.

Exit: The CCS is Off.

D.3 Three Test Suites for the CCS derived from three techniques namely, branch coverage, all-uses coverage and IPO_2-df-chains coverage.

Branch Coverage

Complete Test Suite: {
T1 T14,
T1 T2 T3 T4 T8 T11 T9 T12,
T1 T2 T5 T6 T7 T10 T11 T13
}

All-uses Coverage

1. Identify all du-pairs from the EFSM model

Variables	Def(s)	Use(s)
accelR	d_{T2}^{accelR}	p_{T2}^{accelR}
	d_{T11}^{accelR}	p_{T11}^{accelR}
brakeR	d_{T2}^{brakeR}	p_{T2}^{brakeR}

	d_{T11}^{brakeR}	P_{T11}^{brakeR}
cruise_speed	$d_{T2}^{cruise\_speed}$	$c_{T2}^{cruise\_speed}, p_{T3}^{cruise\_speed}, c_{T4}^{cruise\_speed}, p_{T4}^{cruise\_speed}, c_{T5}^{cruise\_speed}, p_{T5}^{cruise\_speed}, c_{T6}^{cruise\_speed}, c_{T7}^{cruise\_speed}, c_{T11}^{cruise\_speed}$
	$d_{T6}^{cruise\_speed}$	$p_{T3}^{cruise\_speed}, c_{T4}^{cruise\_speed}, p_{T4}^{cruise\_speed}, c_{T5}^{cruise\_speed}, p_{T5}^{cruise\_speed}, c_{T6}^{cruise\_speed}, c_{T7}^{cruise\_speed}, c_{T11}^{cruise\_speed}$
	$d_{T7}^{cruise\_speed}$	$p_{T3}^{cruise\_speed}, c_{T4}^{cruise\_speed}, p_{T4}^{cruise\_speed}, c_{T5}^{cruise\_speed}, p_{T5}^{cruise\_speed}, c_{T6}^{cruise\_speed}, c_{T7}^{cruise\_speed}, c_{T11}^{cruise\_speed}$
current_speed	$d_{T2}^{current\_speed}$	$c_{T2}^{current\_speed}, p_{T2}^{current\_speed}$
	$d_{T3}^{current\_speed}$	$c_{T3}^{current\_speed}, p_{T3}^{current\_speed}$
	$d_{T4}^{current\_speed}$	$c_{T4}^{current\_speed}, p_{T4}^{current\_speed}$
	$d_{T5}^{current\_speed}$	$c_{T5}^{current\_speed}, p_{T5}^{current\_speed}$
	$d_{T6}^{current\_speed}$	$p_{T6}^{current\_speed}$
	$d_{T7}^{current\_speed}$	$p_{T7}^{current\_speed}$
	$d_{T8}^{current\_speed}$	$p_{T8}^{current\_speed}$
	$d_{T11}^{current\_speed}$	$c_{T11}^{current\_speed}, p_{T11}^{current\_speed}$
Dec	d_{T4}^{dec}	c_{T4}^{dec}
inc	d_{T5}^{inc}	c_{T5}^{inc}

2. Construct an activating path for each du-pair

# DU-Pair	DU-Pairs	Activating Paths
1	$[d_{T2}^{accelR}, p_{T2}^{accelR}]$	T2
2	$[d_{T11}^{accelR}, p_{T11}^{accelR}]$	T11
3	$[d_{T2}^{brakeR}, p_{T2}^{brakeR}]$	T2
4	$[d_{T11}^{brakeR}, p_{T11}^{brakeR}]$	T11
5	$[d_{T2}^{cruise\_speed}, c_{T2}^{cruise\_speed}]$	T2
6	$[d_{T2}^{cruise\_speed}, p_{T3}^{cruise\_speed}]$	T2 T3
7	$[d_{T2}^{cruise\_speed}, c_{T4}^{cruise\_speed}]$	T2 T4
8	$[d_{T2}^{cruise\_speed}, p_{T4}^{cruise\_speed}]$	T2 T4
9	$[d_{T2}^{cruise\_speed}, c_{T5}^{cruise\_speed}]$	T2 T5
10	$[d_{T2}^{cruise\_speed}, p_{T5}^{cruise\_speed}]$	T2 T5
11	$[d_{T2}^{cruise\_speed}, c_{T6}^{cruise\_speed}]$	T2 T6

12	$[d_{T2}^{cruise\_speed}, c_{T7}^{cruise\_speed}]$	T2 T7
13	$[d_{T2}^{cruise\_speed}, c_{T11}^{cruise\_speed}]$	T2 T9 T11
14	$[d_{T6}^{cruise\_speed}, p_{T3}^{cruise\_speed}]$	T6 T3
15	$[d_{T6}^{cruise\_speed}, c_{T4}^{cruise\_speed}]$	T6 T4
16	$[d_{T6}^{cruise\_speed}, p_{T4}^{cruise\_speed}]$	T6 T4
17	$[d_{T6}^{cruise\_speed}, c_{T5}^{cruise\_speed}]$	T6 T5
18	$[d_{T6}^{cruise\_speed}, p_{T5}^{cruise\_speed}]$	T6 T5
19	$[d_{T6}^{cruise\_speed}, c_{T6}^{cruise\_speed}]$	T6 T6
20	$[d_{T6}^{cruise\_speed}, c_{T7}^{cruise\_speed}]$	T6 T7
21	$[d_{T6}^{cruise\_speed}, c_{T11}^{cruise\_speed}]$	T6 T10 T11
22	$[d_{T7}^{cruise\_speed}, p_{T3}^{cruise\_speed}]$	T7 T3
23	$[d_{T7}^{cruise\_speed}, c_{T4}^{cruise\_speed}]$	T7 T4
24	$[d_{T7}^{cruise\_speed}, p_{T4}^{cruise\_speed}]$	T7 T4
25	$[d_{T7}^{cruise\_speed}, c_{T5}^{cruise\_speed}]$	T7 T5
26	$[d_{T7}^{cruise\_speed}, p_{T5}^{cruise\_speed}]$	T7 T5
27	$[d_{T7}^{cruise\_speed}, c_{T6}^{cruise\_speed}]$	T7 T6
28	$[d_{T7}^{cruise\_speed}, c_{T7}^{cruise\_speed}]$	T7 T7
29	$[d_{T7}^{cruise\_speed}, c_{T11}^{cruise\_speed}]$	T7 T8 T11
30	$[d_{T2}^{current\_speed}, c_{T2}^{current\_speed}]$	T2
31	$[d_{T2}^{current\_speed}, p_{T2}^{current\_speed}]$	T2
32	$[d_{T3}^{current\_speed}, c_{T3}^{current\_speed}]$	T3
33	$[d_{T3}^{current\_speed}, p_{T3}^{current\_speed}]$	T3
34	$[d_{T4}^{current\_speed}, c_{T4}^{current\_speed}]$	T4
35	$[d_{T4}^{current\_speed}, p_{T4}^{current\_speed}]$	T4
36	$[d_{T5}^{current\_speed}, c_{T5}^{current\_speed}]$	T5
37	$[d_{T5}^{current\_speed}, p_{T5}^{current\_speed}]$	T5
38	$[d_{T6}^{current\_speed}, p_{T6}^{current\_speed}]$	T6
39	$[d_{T7}^{current\_speed}, p_{T7}^{current\_speed}]$	T7
40	$[d_{T8}^{current\_speed}, p_{T8}^{current\_speed}]$	T8
41	$[d_{T11}^{current\_speed}, c_{T11}^{current\_speed}]$	T11
42	$[d_{T11}^{current\_speed}, p_{T11}^{current\_speed}]$	T11
43	$[d_{T4}^{dec}, c_{T4}^{dec}]$	T4
44	$[d_{T5}^{inc}, c_{T5}^{inc}]$	T5

3. Generate test paths to cover all activating paths

Test id(s)	Test Paths	DU-pairs covered by a test path
1	T1 T2 T3 T9 T11 T6 T4 T7 T3 T8 T11 T13	1, 2, 3, 4, 6, 13, 15, 16, 22, 30, 31, 32, 33, 34, 35, 38, 39, 40, 41, 42, 43
2	T1 T2 T4 T6 T3 T10 T11 T7 T4 T13	1, 2, 3, 4, 7, 8, 14, 23, 24, 30, 31, 32, 33, 34, 35, 38, 39, 41, 42, 43
3	T1 T2 T5 T6 T7 T6 T5 T13	1, 3, 9, 10, 17, 18, 20, 30, 31, 36, 37, 38, 39, 44
4	T1 T2 T6 T6 T7 T5 T10 T12	1, 3, 11, 19, 20, 25, 26, 30, 31, 36, 37, 38, 39, 44
5	T1 T2 T7 T7 T6 T9 T12	1, 3, 12, 27, 28, 30, 31, 38, 39

IPO_2-df-chains Coverage

1. Identify all ipo-pairs and their IPO_2-df-chains from the EFSM model

#	Ipo-pairs	IPO_2-df-chains	
1	$[d_{T2}^{current\_speed}, u_{T4}^{current\_speed}]$	$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T4	C1
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T4	C2
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T4	C3
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T6 T4	C4
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T7 T4	C5

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T6 T4	C6
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T7 T4	C7
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T6 T7 T4	C8
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T6 T7 T7 T4	C9
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T7 T6 T4	C10
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T7 T6 T7 T4	C11
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T7 T7 T4	C12

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T7 T7 T6 T4	C13
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T6 T6 T4	C14
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T6 T6 T7 T4	C15
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T6 T7 T4	C16
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T6 T7 T6 T4	C17
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T7 T6 T4	C18
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T7 T7 T6 T6 T4	C19

2	$[d_{T2}^{current\_speed}, u_{T5}^{current\_speed}]$	$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T5	C20
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T5	C21
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T5	C22
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T6 T5	C23
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T7 T5	C24
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T6 T5	C25
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T7 T5	C26
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T6 T7 T5	C27

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T4}^{cruise\_speed}, d_{T4}^{dec}, u_{T4}^{dec}, d_{T4}^{current\_speed}, u_{T4}^{current\_speed}\}$ T2 T6 T6 T7 T7 T5	C28
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T7 T6 T5	C29
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T7 T6 T7 T5	C30
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T7 T7 T5	C31
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T6 T7 T7 T6 T5	C32
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T6 T6 T5	C33
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T6 T6 T7 T5	C34

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T6 T7 T5	C35
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T6 T7 T6 T5	C36
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T7 T6 T5	C37
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T5}^{cruise\_speed}, d_{T5}^{inc}, u_{T5}^{inc}, d_{T5}^{current\_speed}, u_{T5}^{current\_speed}\}$ T2 T7 T7 T6 T6 T5	C38
3	$[d_{T2}^{current\_speed}, u_{T6}^{cruise\_speed}]$	$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T6	C39
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T6 T6	C40
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T7 T6	C41
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T7 T7 T6	C42

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T6 T7 T6	C43
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T7 T6 T6	C44
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T7 T7 T6 T6	C45
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T7 T6 T7 T6	C46
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}\}$ T2 T6 T7 T7 T6	C47
4	$[d_{T2}^{current\_speed}, u_{T7}^{cruise\_speed}]$	$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T7	C48
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T7 T7	C49
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T6 T7	C50
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T6 T6 T7	C51

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T7 T6 T7	C52
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T6 T7 T7	C53
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T6 T6 T7 T7	C54
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T6 T7 T6 T7	C55
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}\}$ T2 T7 T6 T6 T7	C56
5	$[d_{T2}^{current\_speed}, u_{T11}^{current\_speed}]$	$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T8 T11	C57
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T9 T11	C58
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T10 T11	C59
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T6 T8 T11	C60

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T7 T9 T11	C61
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T6 T10 T11	C62
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T7 T8 T11	C63
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T6 T7 T9 T11	C64
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T6 T7 T7 T10 T11	C65
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T7 T6 T8 T11	C66
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T7 T6 T7 T9 T11	C67
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T7 T7 T10 T11	C68

		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T6 T7 T7 T6 T8 T11	C69
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T6 T6 T9 T11	C70
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T6 T6 T7 T10 T11	C71
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T6 T7 T8 T11	C72
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T6 T7 T6 T9 T11	C73
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T7 T6 T10 T11	C74
		$\{d_{T2}^{current\_speed}, u_{T2}^{current\_speed}, d_{T2}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T7}^{cruise\_speed}, d_{T7}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T6}^{cruise\_speed}, d_{T6}^{cruise\_speed}, u_{T11}^{cruise\_speed}, d_{T11}^{current\_speed}, u_{T11}^{current\_speed}\}$ T2 T7 T7 T6 T6 T8 T11	C75

2. Construct an activating path for each IPO_2-df-chain

	Activating Paths
C1	T2 T4
C2	T2 T6 T4
C3	T2 T7 T4
C4	T2 T6 T6 T4
C5	T2 T6 T7 T4
C6	T2 T7 T6 T4
C7	T2 T7 T7 T4
C8	T2 T6 T6 T7 T4
C9	T2 T6 T6 T7 T7 T4
C10	T2 T6 T7 T6 T4
C11	T2 T6 T7 T6 T7 T4
C12	T2 T6 T7 T7 T4
C13	T2 T6 T7 T7 T6 T4
C14	T2 T7 T6 T6 T4
C15	T2 T7 T6 T6 T7 T4
C16	T2 T7 T6 T7 T4
C17	T2 T7 T6 T7 T6 T4
C18	T2 T7 T7 T6 T4
C19	T2 T7 T7 T6 T6 T4
C20	T2 T5
C21	T2 T6 T5
C22	T2 T7 T5
C23	T2 T6 T6 T5
C24	T2 T6 T7 T5
C25	T2 T7 T6 T5
C26	T2 T7 T7 T5
C27	T2 T6 T6 T7 T5
C28	T2 T6 T6 T7 T7 T5
C29	T2 T6 T7 T6 T5
C30	T2 T6 T7 T6 T7 T5
C31	T2 T6 T7 T7 T5
C32	T2 T6 T7 T7 T6 T5
C33	T2 T7 T6 T6 T5
C34	T2 T7 T6 T6 T7 T5
C35	T2 T7 T6 T7 T5
C36	T2 T7 T6 T7 T6 T5
C37	T2 T7 T7 T6 T5
C38	T2 T7 T7 T6 T6 T5
C39	T2 T6
C40	T2 T6 T6
C41	T2 T7 T6
C42	T2 T7 T7 T6

C43	T2 T6 T7 T6
C44	T2 T7 T6 T6
C45	T2 T7 T7 T6 T6
C46	T2 T7 T6 T7 T6
C47	T2 T6 T7 T7 T6
C48	T2 T7
C49	T2 T7 T7
C50	T2 T6 T7
C51	T2 T6 T6 T7
C52	T2 T7 T6 T7
C53	T2 T6 T7 T7
C54	T2 T6 T6 T7 T7
C55	T2 T6 T7 T6 T7
C56	T2 T7 T6 T6 T7
C57	T2 T8 T11
C58	T2 T6 T9 T11
C59	T2 T7 T10 T11
C60	T2 T6 T6 T8 T11
C61	T2 T6 T7 T9 T11
C62	T2 T7 T6 T10 T11
C63	T2 T7 T7 T8 T11
C64	T2 T6 T6 T7 T9 T11
C65	T2 T6 T6 T7 T7 T10 T11
C66	T2 T6 T7 T6 T8 T11
C67	T2 T6 T7 T6 T7 T9 T11
C68	T2 T6 T7 T7 T10 T11
C69	T2 T6 T7 T7 T6 T8 T11
C70	T2 T7 T6 T6 T9 T11
C71	T2 T7 T6 T6 T7 T10 T11
C72	T2 T7 T6 T7 T8 T11
C73	T2 T7 T6 T7 T6 T9 T11
C74	T2 T7 T7 T6 T10 T11
C75	T2 T7 T7 T6 T6 T8 T11

3. Generate a set of test paths that cover each IPO_2-df-chain at least once

Test id(s)	Test Paths	IPO_2-df-chains covered by a test path
1	T1 T2 T4 T13	C1
2	T1 T2 T6 T4 T13	C2
3	T1 T2 T7 T4 T13	C3
4	T1 T2 T6 T6 T4 T13	C4
5	T1 T2 T6 T7 T4 T13	C5
6	T1 T2 T7 T6 T4 T13	C6
7	T1 T2 T7 T7 T4 T13	C7

8	T1 T2 T6 T6 T7 T4 T13	C8
9	T1 T2 T6 T6 T7 T7 T4 T13	C9
10	T1 T2 T6 T7 T6 T4 T13	C10
11	T1 T2 T6 T7 T6 T7 T4 T13	C11
12	T1 T2 T6 T7 T7 T4 T13	C12
13	T1 T2 T6 T7 T7 T6 T4 T13	C13
14	T1 T2 T7 T6 T6 T4 T13	C14
15	T1 T2 T7 T6 T6 T7 T4 T13	C15
16	T1 T2 T7 T6 T7 T4 T13	C16
17	T1 T2 T7 T6 T7 T6 T4 T13	C17
18	T1 T2 T7 T7 T6 T4 T13	C18
19	T1 T2 T7 T7 T6 T6 T4 T13	C19
20	T1 T2 T5 T13	C20
21	T1 T2 T6 T5 T13	C21
22	T1 T2 T7 T5 T13	C22
23	T1 T2 T6 T6 T5 T13	C23
24	T1 T2 T6 T7 T5 T13	C24
25	T1 T2 T7 T6 T5 T13	C25
26	T1 T2 T7 T7 T5 T13	C26
27	T1 T2 T6 T6 T7 T5 T13	C27
28	T1 T2 T6 T6 T7 T7 T5 T13	C28
29	T1 T2 T6 T7 T6 T5 T13	C29
30	T1 T2 T6 T7 T6 T7 T5 T13	C30
31	T1 T2 T6 T7 T7 T5 T13	C31
32	T1 T2 T6 T7 T7 T6 T5 T13	C32
33	T1 T2 T7 T6 T6 T5 T13	C33
34	T1 T2 T7 T6 T6 T7 T5 T13	C34
35	T1 T2 T7 T6 T7 T5 T13	C35
36	T1 T2 T7 T6 T7 T6 T5 T13	C36
37	T1 T2 T7 T7 T6 T5 T13	C37
38	T1 T2 T7 T7 T6 T6 T5 T13	C38
39	T1 T2 T6 T6 T13	C39, C40
40	T1 T2 T7 T6 T13	C41
41	T1 T2 T7 T7 T6 T13	C42
42	T1 T2 T6 T7 T6 T13	C43
43	T1 T2 T7 T6 T6 T13	C44
44	T1 T2 T7 T7 T6 T6 T13	C45
45	T1 T2 T7 T6 T7 T6 T13	C46
46	T1 T2 T6 T7 T7 T6 T13	C47
47	T1 T2 T7 T7 T13	C48, C49
48	T1 T2 T6 T7 T13	C50
49	T1 T2 T6 T6 T7 T13	C51
50	T1 T2 T7 T6 T7 T13	C52
51	T1 T2 T6 T7 T7 T13	C53
52	T1 T2 T6 T6 T7 T7 T13	C54

53	T1 T2 T6 T7 T6 T7 T13	C55
54	T1 T2 T7 T6 T6 T7 T13	C56
55	T1 T2 T8 T11 T13	C57
56	T1 T2 T6 T9 T11 T13	C58
57	T1 T2 T7 T10 T11 T13	C59
58	T1 T2 T6 T6 T8 T11 T13	C60
59	T1 T2 T6 T7 T9 T11 T13	C61
60	T1 T2 T7 T6 T10 T11 T13	C62
61	T1 T2 T7 T7 T8 T11 T13	C63
62	T1 T2 T6 T6 T7 T9 T11 T13	C64
63	T1 T2 T6 T6 T7 T7 T10 T11 T13	C65
64	T1 T2 T6 T7 T6 T8 T11 T13	C66
65	T1 T2 T6 T7 T6 T7 T9 T11 T13	C67
66	T1 T2 T6 T7 T7 T10 T11 T13	C68
67	T1 T2 T6 T7 T7 T6 T8 T11 T13	C69
68	T1 T2 T7 T6 T6 T9 T11 T13	C70
69	T1 T2 T7 T6 T6 T7 T10 T11 T13	C71
70	T1 T2 T7 T6 T7 T8 T11 T13	C72
71	T1 T2 T7 T6 T7 T6 T9 T11 T13	C73
72	T1 T2 T7 T7 T6 T10 T11 T13	C74
73	T1 T2 T7 T7 T6 T6 T8 T11 T13	C75

D.4 Test Results after applying STSR and DTSR programs to the test suites in D.3

By applying STSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	2	1	Test_3
T3	1	1	
T4	1	1	
T5	1	1	
T6	1	1	
T7	1	1	
T8	1	1	
T9	1	1	
T10	1	1	
T11	2	2	
T13	1	1	

All–uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	5	1	Test_2 – Test_5
T3	2	2	
T4	2	2	
T5	2	2	
T6	5	5	
T7	5	5	
T8	1	1	
T9	2	1	Test_5
T10	2	1	Test_4
T11	2	2	
T13	3	3	

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	73	1	Test_2 – Test_73
T3	0	0	
T4	19	19	
T5	19	19	
T6	63	9	Test_5, 8, 9, 11, 12, 15, 16, 21, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 56, 58, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73
T7	63	9	Test_6, 10, 13, 14, 17, 18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 57, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73
T8	7	1	Test_58, 61, 64, 67, 70, 73
T9	6	1	Test_59, 62, 65, 68, 71
T10	6	1	Test_60, 63, 66, 69, 72
T11	19	19	
T13	73	20	Test_2 – Test_54

By applying DTSR

Branch Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	2	1	Test_3
T3	1	1	
T4	1	1	
T5	1	1	
T6	1	1	
T7	1	1	
T8	1	1	
T9	1	1	
T10	1	1	
T11	2	2	
T13	1	1	

All-uses Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	5	1	Test_2 – Test_5
T3	2	2	
T4	2	2	
T5	2	2	
T6	5	5	
T7	5	5	
T8	1	1	
T9	2	1	Test_5
T10	2	1	Test_4
T11	2	2	
T13	3	3	

IPO_2-df-chains Coverage

Transition	Number of test cases	Number of reduced test cases	List of eliminated test cases (specified by test ids)
T2	73	1	Test_2 – Test_73
T3	0	0	
T4	19	19	
T5	19	19	
T6	63	9	Test_5, 8, 9, 11, 12, 15, 16, 21, 23, 24, 25, 27, 28, 29,

			30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 56, 58, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73
T7	63	9	Test_6, 10, 13, 14, 17, 18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 57, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73
T8	7	1	Test_58, 61, 64, 67, 70, 73
T9	6	1	Test_59, 62, 65, 68, 71
T10	6	1	Test_60, 63, 66, 69, 72
T11	19	19	
T13	73	20	Test_2 – Test_54