Reliability Measurement Based on the Markov Models for Real-time Reactive Systems: Design and Implementation

Fong-An Lee

A Major Report

In

Department

Of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montreal, Quebec, Canada

August 2003

Your file Votre référence
ISBN: 0-612-83909-5
Our file Notre référence
ISBN: 0-612-83909-5

Abstract

Reliability Measurement Based on the Markov Models for Real-time Reactive Systems: Design and Implementation

Fong-An Lee

This major report describes the design and implementation of the TROM-SRMS: software reliability measurement system for real-time reactive system in the TROMLAB environment. The TROM-SRMS is responsible for the reliability computation of each individual sub-system in a real-time reactive system. In order to achieve cross-platform capability, Java has been chosen as the implementation tool. A GUI interface is added on top of the real computation unit to facilitate the use of the software. Finally, the key algorithms used in the design and implementations are presented in detail as well as the interactions among the underlying objects.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Olga Ormandjieva. She gave me helpful guidance and advices all along the way.

My sincere thanks to Dr. Alagar for being the examiner of this project, his advises and comments were very helpful.

Also, I would like to thank my family for the support during the years of my graduate studies.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 The importance of software quality

Software Engineering is a discipline for the systematic construction and support of software products so they can safely fill the uses to which they may be subjected. As any engineering approach, software engineering requires a quality control mechanism to provide a feedback and assist the software development, testing and maintenance. In the context of real-time systems, which are mostly safety-critical, the main motivation for quality control comes from the requirements for a reliable implementation of safety and time-dependent behavior. Examples of software systems that are safety critical are air traffic control systems, process control systems in chemical, pharmaceutical plants and nuclear reactor systems [SOM95]. The malfunctioning of these software systems poses a potential threat to the human life and the living environment. Since the software cannot be certified as 100% safe, in order to avoid the potential dangers related to software in these safety critical systems, an attempt to largely use hardware instead of software is made. Safety-validation and assurance techniques have been developed for hardware [SOM95]. As an example of hardware safety control, a switch that is made of metallic strip that bends under high temperature has its safety controlled by hardware.

Nevertheless, as systems are becoming more and more complex, safety control based on hardware is no long sufficient. As an example, the military crafts that fly under an aerodynamic unstable environment need software-control system to adjust the flight surface as the environment changes.

As software is a mandatory element in building a complex system, software quality becomes an essential factor that determines the success of the software.

1.2 Purpose and Problem Statement

The reliability of software is crucial when it is being used in a mission critical environment such as nuclear reactor controller, air traffic controller system, etc. The reliability measurement could serve as an indication as whether or not the software is reliable enough to be used, or should be rejected to avoid failure or catastrophes.

The main goal of this major report is to describe the developed reliability measurement mechanism and its implementation in TROMLAB [AAM98], an environment for real-time reactive systems development based on the TROM formalism [Ach95]. The theory that is used for reliability computation is based on [Orm02] with some modifications. The main contributions of the report are:

- Studying the algorithm proposed by Dr. Olga Ormandjieva [Orm02], and make modifications to the reliability measurement algorithms;
- Implementation of the updated algorithms;
- Testing the implementation on the gate-train-controller case study and compare the testing results to the theoretical ones.

This major report presents one possible implementation of the reliability computation module in the TROMLAB environment. This reliability computation

unit, TROM-SRMS, is capable of computing the reliability of a real-time reactive system that is formalized by TROM formalism.

1.3 Report Outline

The major report is organized as follows: Chapter 2 introduces briefly the TROM formalism, and gives an overview on the TROMLAB environment. Chapter 3 overviews the reliability measurement, and introduces the reliability computation theory based on Markov model [Orm02]. The software architecture and the design of the reliability measurement implementation, TROM-SRMS, are documented in Chapter 4. Chapter 5 contains the key algorithms used in the reliability computation. Chapter 6 describes the G*ate-Train-Controller* case study. Theoretical computation results are presented in the comparison to the testing results gathered from the output of TROM-SRMS. The conclusions and the future work are outlined in Chapter 7.

Chapter 2. Background

The proposed reliability measurement theory is derived from TROM formalism in the context of TROMLAB, an environment for rigorous development of real-time reactive system. The goal of this chapter is to give an overview of the TROM formalism and the TROMLAB environment for development of real-time reactive systems.

2.1 Real-time Reactive Systems

Real-time reactive systems are systems which are continuously interacting with the environment through stimulus-response behaviour. These systems are event-driven, and their behaviours are controlled by strict timing constrains.

Real-time reactive systems possess the properties of *stimulus synchronization* and *response synchronization,* which distinguish them from other systems. *Stimulus synchronizatio*n states that the system is constantly reacting to a stimulus from its surrounding environment, whereas r*esponse synchronization* asserts that the elapsed time between a stimulus and its response is within a range such that the dynamics of the environment is kept and thus the environment stays receptive to the response.

2.2 TROM Formalism

TROM formalism provides ways to specify, analyze and refine real-time reactive systems.

2.2.1 Reactive Object Model

A reactive object is an abstract state machine built-up with ports, attributes, logical assertions on the attributes, and timing constraints [Orm02]. Simple and complex states exist in the context of reactive object. A complex state is an encapsulation of another state machine which itself could be either a simple or complex state. Synchronous message passing is the communication mechanism among the reactive objects. Each reactive object possesses a number of ports and both external input/output events can only occur at port of a specific type, whereas an internal event occurs at null port. The port links are the linkages among reactive objects. Two ports are compatible if one port can receive as its input messages the set of output messages from the other port. All the messages that can be exchanged between two ports are determined by the port links. The type of an attribute can be either port type, or abstractly modeled as LSL trait. Pre and post condition on a transition and port condition can be specified using logical assertions and timing constraints. The response time constraint of a reactive object to a stimulus is described on each transition. A generic reactive class is a collection of reactive objects. All the reactive objects, which are generated from the same generic reactive class, have the same attributes. Putting in an object-oriented perspective, a generic reactive class, GRC, is a class declaration, whereas each individual reactive object is an instance of that particular class. TROM formalism is the basis used to specify GRCs and thus reactive objects in a system. TROM is a three-tier formalism; each tier has its own output and processing mechanism.

2.2.2 Layer one – Larch Formalism

Larch[GH93] is a two tier specifications. The top tier is *Larch Interface Language* (LIL), which is tailored to a specific programming language. The bottom tier is *Larch Shared Language* (LIL), which is a common specification method for the programming languages.

Typically, LIL uses the declaration syntax of a specific programming language, and the behaviors are specified by adding annotations. These annotations consist of pre and post conditions.

LSL is a language for specifying mathematical theories. The specifications in LSL consist of first-order equations between terms. Two symbols are defined in LSL, namely s*orts and operators*. The sorts correspond to the types in programming language and they are names for sets of values. The domains and ranges of the operators are indicated by the sorts' symbol. The operators are equivalent to procedure in conventional programming language and they represent total functions from tuples of values to values.

An LSL *trait* is used to describe a mathematical theory. A trait acts as a basic unit of specification. In each trait, operators are introduced and defined with a set of equations that defines which terms are equal to one another, and may assert additional properties about the sorts and operators. A trait can use another trait by including it in the *includes* section of the specification. Figure 1 shows an example of the *BirthdayBook* trait:

BirthdayBook : **trait**

includes

```
    Set(Name,NameSet),
    FiniteMap(NameToDate,Name,Date)
  introduces
    known: -> NameSet
    birthday: -> NameToDate
  asserts
    forall n: Name
      n \in known <=> defined(birthday,n)
```

Figure 1 LSL trait for Birthday Book

2.2.3 Layer two – TROM Class

A TROM class is equivalent to a Generic Reactive Class (GRC). It is a hierarchical finite state machine augmented with ports, attributes, logical assertions on the attributes and time constraints. Figure 2 shows a TROM object. All the reactive objects generated from generic reactive class communicate among them and the environment through their ports. The port can receive and send a synchronous message.

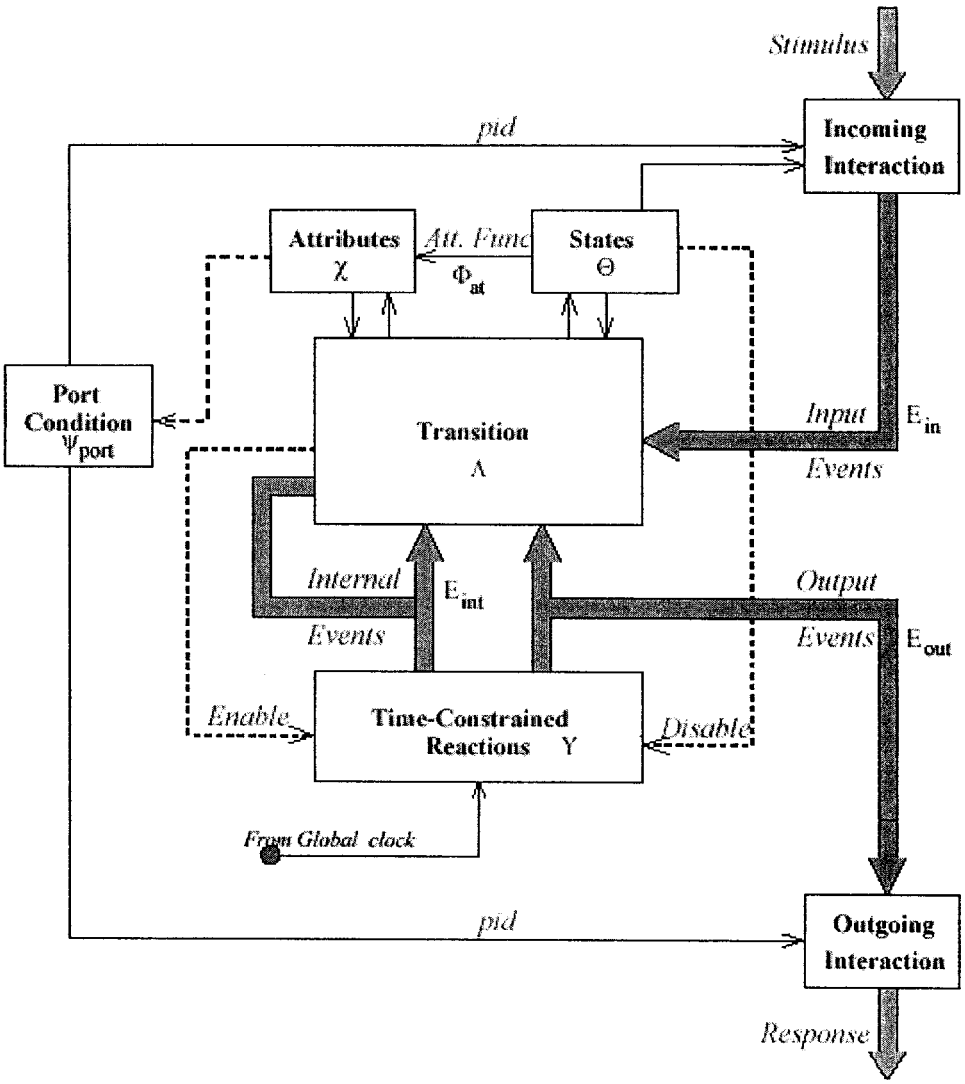

Figure 2 Reactive Object

A TROM object is made of a set of events, states, attributes and its related functions, transitions and a set of timing constraints. Three types of events exist, *external input*, *external output* and *internal* and they are represented symbolically by e?, e! and e respectively. An attribute of TROM object can be either one of the following types:

- Abstract data type imported from the first tier
- Port types

Each reactive object instance can have multiple port types. A port is an abstractly modeled bi-directional access point between environment and a TROM object. A port can only process a set of messages defined by its port type. The signature of a port type K is denoted as $\mathcal{E}^K$. The relationships between attributes and states are defined by attribute functions. A transition function describes the state change due to a particular event. A transition is caused by the occurrence of either an internal event or an external event. A timing constraint on a transition specifies the constraint of response to stimulus in terms of time.

A generic reactive object [Ach95] is an 8-tuple ($P, \mathcal{E}$, Θ, X, £, Φ, Λ, Γ) such that P represents a finite set of port-types, $\mathcal{E}$ is a finite set of events, Θ is a finite set of states, X is a finite set of typed attributes, £ is a finite set of LSL traits, Φ is a function-vector, Λ is a finite set of transition specifications and Γ is a finite set of time-constraints. A template of a GRC class is shown in Figure 3. Examples of GRC formal specifications for the Train, Controller and Gate are given in Figures 13, 15 and 17 respectively (Chapter 6).

Class<name>

Events:

States:

Attributes:

Traits:

Attribute-Function:

Transition-specifications:

Time-Constraints:

End

Figure 3. Template for GRC class

2.2.4 Layer three – System Configuration Specification

A system configuration specification specifies how a reactive system is configured based on the objects instantiated from the second tier. A template for a system configuration specification is shown in Figure 4.

Subsystem <name>

Include:

Instantiate:

Configure:

End

Figure 4. Template for System Configuration Specification

The system configuration specification for the Train, Controller and Gate system is illustrated in Figure 20 (Chapter 6).

2.3 TROMLAB

TROMLAB is a framework that provides an environment for rigorous development of real-time reactive system. It is based on TROM, which is a formalism based on object-oriented and the real-time technologies.

TROMLAB has a number of tools to assist in the development of real-time reactive systems. These tools provide an automatic mechanism to collect and analyze quality measurement data.

The current architecture of the TROMLAB consists of the following components:

- Rose-GRC Translator – [Pop99] module which maps graphical model in Rational Rose to formal specification based on TROM;
- Interpreter – [Tao96] performs syntactically verification on specification and generate an internal representation of it;
- Simulator – [Mut96] creates animation of a subsystem based on internal representation generated by the interpreter;
- Browser for Reuse – [Nag99] an interface to a library which helps the user to navigate, query and access system components during the development;
- Graphical User Interface – [Sri99] an interface for the system developer to interact with the TROMLAB environment;
- Reasoning System – [Hai99] Debugging facility that allows the user to query the system behaviour based on interact queries;
- Verification Assistant – [Pom99] an automated tool that extracts mechanized axiom from real-time reactive systems;
- Test Cases Generator – [Zhe02] and [Che02] an automated tool for generating test cases from specifications.

The aim of this work is to develop and integrate within TROMLAB a tool for reliability assessment before the implementation. The concept of the reliability

measurement, and the reliability measurement mechanism based on Markov model, are addressed in the next chapter.

Chapter 3. Overview of Reliability Measurement

3.1 Reliability Module in TROMLAB

The reliability measurement module TROM-SRMS is a new component in the TROMLAB environment. It is a standalone independent module responsible for evaluating the reliability of a real-time reactive system based on GRC specifications, SCS specifications, and the synchronous product machine of these GRC objects. The formal specifications are generated by Rose-GRC Translator [Pop99], and the product machine is outputted by the Specification-based Testing System tool [Che02].

3.2 Markov Model

Markov model is a very powerful tool for scientists and engineers to analyze and predict the behaviors of a complex system.

A Markov model analysis can yield a variety of useful performance measures describing the operation of the system. These performance measures include the following:

- System reliability
- Availability
- Mean time to failure (MTTF)
- Mean time between failures (MTBF)
- The probability of being in a given state at a given time

- The probability of repairing the system within a given time period (maintainability)
- The average number of visits to a given state within a given time period

The Markov property states that given the current state of the system, the future evolution of the system is not dependant of its past history.

Markov model is applicable to model the reliability of a reactive system is due to the following reasons:

- Environmental laws are considered as random and not controlled by system laws;
- Being in a particular state, a system may choose to execute any of the transition available at that state in order to move to another state.

3.3 Reliability Model

3.3.1 State Diagram

A state diagram represents the dynamic behavior of an object. It describes the set of states and a set of transitions between the states. Each transition is associated to a particular event that causes the triggering of state change. Figures 14, 16 and 18 are examples of state transition diagrams that correspond to the reactive objects *Train*, *Controller* and *Gate*.

3.3.1 Transition Matrix

Each reactive object has an associated state transition diagram that visualizes its states and transitions among them. Each event that triggers a transition has some probability to happen. When an object is in state *i*, the

probability of it moving to state *j* is denoted as P_{ij}, and is called the transition probability. A transition matrix is a matrix *M* such that for each ij_{th} entry corresponds to P_{ij} . We compute the transition probabilities from the specifications of the individual objects' behavior, and the specification of their collective behavior – the synchronous product machine.

3.3.2 Synchronous Product Machine

A synchronous product machine describes the interactions between the reactive objects. The interactions between two reactive objects are triggered by shared events.

Let *P* and *Q* be two reactive objects. Each object can transit from one state to the other according to a fixed probability. If a shared event occurs when the objects are in the states which are reactive to such event, then both objects perform simultaneously a transition. The synchronous product machine captures all these simultaneous transitions between reactive objects. Figure 5 shows an example of synchronous product machine for reactive objects *Train* and *Controller*.

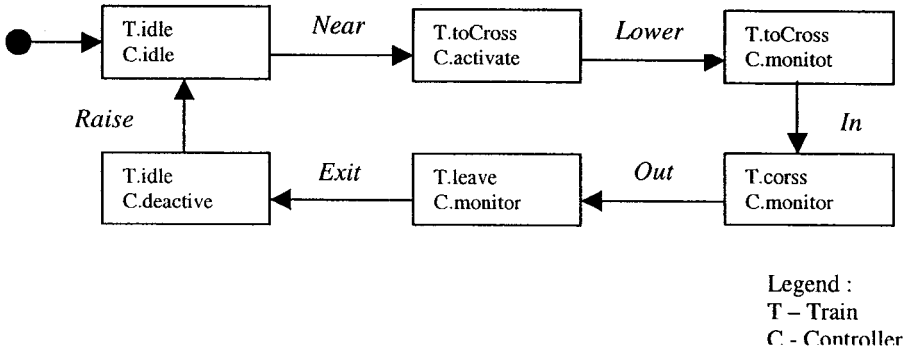

Figure 5. Synchronous Product Machine for GRC Train and Controller

The specification of a synchronous product machine can be obtained by running the Specification-based Testing System [Che02]. The specification has the following two sections:

- *State List:* A list of all the states in the synchronous product machine. Notice that a state in a synchronous product machine is a *composite state* that is formed by merging individual GRC state from each reactive object into one state.
- *Transition Spec List:* A list of transitions along with source and destination state and the event that triggers the transition.

Figure 19 (Chapter 6) shows an example of the specification of a synchronous product machine for a *Gate-Train-Controller* subsystem.

3.3.3 Property

The reliability assessment model [Orm02] used in the reliability module is significantly different from the conventional reliability evaluation methods in the following ways:

- Based purely on architecture model of the reactive system and the state machine descriptions of the reactive units;
- Based on Markov system;
- The prediction of the reliability is derived from the steady state of the Markov system.

The advantage of this reliability model is its applicability at early stage of the development cycle such as design specification phase.

Chapter 4. TROM-SRMS

This main goal of the project is to develop a reliability computation module, TROM-SRMS, in the TROMLAB environment. TROM-SRMS computes the prediction of reliability of a reactive system based on the Markov model. In this chapter, system architecture and detail design of the reliability assessment tool are presented.

4.1 Description of the System Functionalities

The TROM-SRMS is a standalone independent module in the TROMLAB environment. The inputs to this system are:

1. GRC specifications files of a subsystem;
2. Synchronous product machine specification generated from the GRC specification files;
3. System configuration file;

The outputs from the system are:

1. For each subsystem, there is a corresponding output file that contains all the intermediate computational results, these include:
    - Transition matrix for each GRC specification in the subsystem;
    - Transition matrix for synchronous product machine generated from the GRC specifications;
    - Steady vector of each GRC specification;
    - Steady vector of the synchronous product machine specification;

- Various debugging information on the contents of the internal data structure;
- The reliability of the subsystem.

2. An output file that contains the reliability of each subsystem and the reliability of the overall system.

All the input files are generated by the TROM-SBTS [Che02]. In order to achieve cross-platform capability, the system is implemented in Java. The version of the JVM and JDK chosen is JSDK 1.4.1

4.2 Description of the System Components

The system is composed of three main components: *GrcStateParser*, *ProductMachineParser* and *Subsystem*. Each GRC specification has its corresponding *GrcStateParser*. The main functionalities of the *GrcStatePaser* include:

- Parse GRC specifications to identify a list of states, events and transitions;
- Compute transition matrix of the GRC specification based on the transitions on each state;
- Compute the steady vector of the GRC;
- Compute entropy value of the GRC.

The *ProductMachineParser* is associated to a synchronous product machine specification of a particular subsystem. The main functionalities provided by *ProductMachineParser* are:

- Parse synchronous product machine specification to identify a list of states, events and transitions;

- Compute transition matrix of the synchronous product machine based on the probabilities of each transition of each individual sub-state in a composite state;
- Compute the steady vector of the synchronous product machine specification;
- Compute the entropy value of the synchronous product machine;
- Compute the reliability of the subsystem.

The *subsystem* groups all the *GrcStateParser* and *ProductMachineParser* of a particular subsystem. It provides encapsulation abstraction to a subsystem and hence the internal processing of the reliability computation is private and independent to each subsystem. The functionalities of the *subsystem* include:

- Encapsulate subsystem whose reliability is to be computed;
- Compute the final reliability of the overall system;
- Provides access point for the GUI and the computational unit of the TROM-SRMS.

TROM-SRMS has a graphical user interface (GUI) for the user to operate. The GUI provides functionality to allow the user to choose GRC specification files, synchronous product machine specification file and system configuration file. The user can also check the current status of the file configuration, meaning which files have been chosen for which subsystem. Lastly, the GUI allows the user to start the reliability computation after the relevant files are selected. Figure 6, 7 and 8 are snapshots of the GUI of TROM-SRMS.

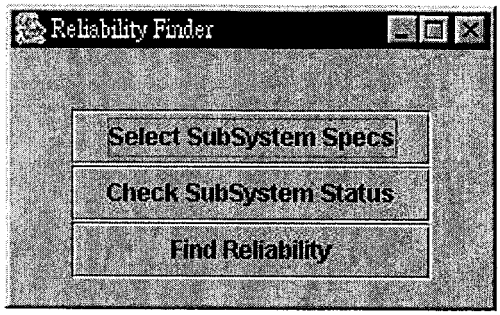

Figure 6 Snapshot of TROM-SRMS GUI Front page

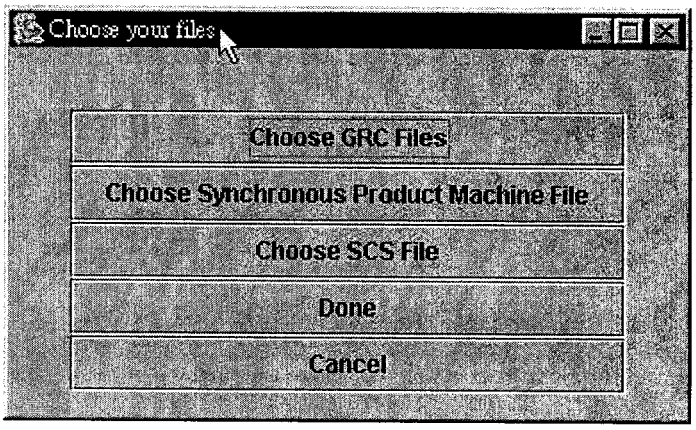

Figure 7 Snapshot of TROM-SRMS GUI File Chooser Menu

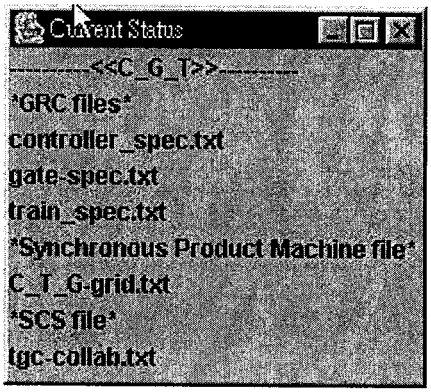

Figure 8 Snapshot of TROM-SRMS Status Window

Assumptions

The major restriction imposed by the *GrcStateParser* and is that the format of the specification files must respect the format described in [Orm02]. As

for the *ProductMachineParser*, the format of the specification file has to follow the one defined in TROM-SBTS. Since all these input files are generated by the TROM-SBTS, it is assumed that they are all in the correct format.

4.3 Architecture Diagram

The architecture diagram of the TROM-SRMS is shown in Figure 9.

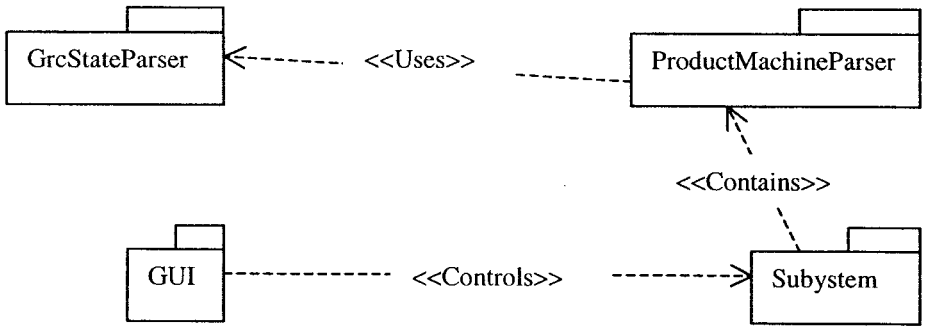

Figure 9 The Architecture Diagram of the TROM-SRMS

4.4 Data Flow Diagram

The GUI takes as input the specification files (GRC and synchronous product machine), it then creates a subsystem object. The *Subsystem* then creates *GrcStateParser* and *ProductMachineParser* objects by passing the appropriate specification files. The *ProductMachineParser* then uses *GrcStateParser* objects in the final computation of the reliability. The *GrcStateParser* and *ProductMachineParser* generate the subsystem profiles in the end and the reliability profile is created by the Subsystem. The data flow diagram (Figure 10) shows the flow of data in the TROM-SRMS.

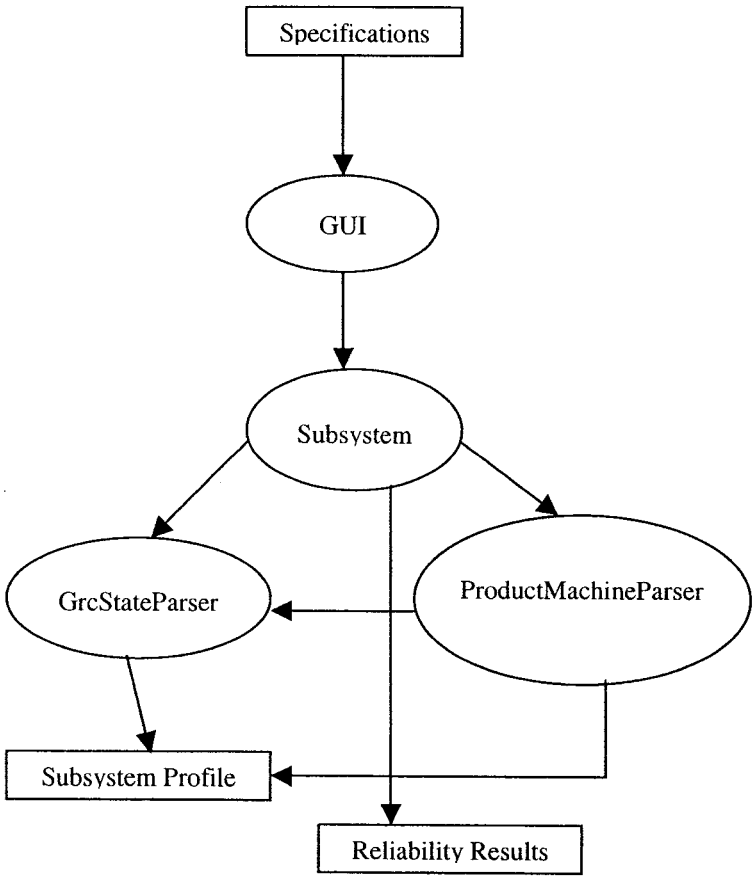

Figure 10 The Data Flow Diagram of TROM-SRMS

4.5 Class Diagram

The class diagram shown in Figure 11 illustrates the classes and their relationships in the context of TROM-SRMS.

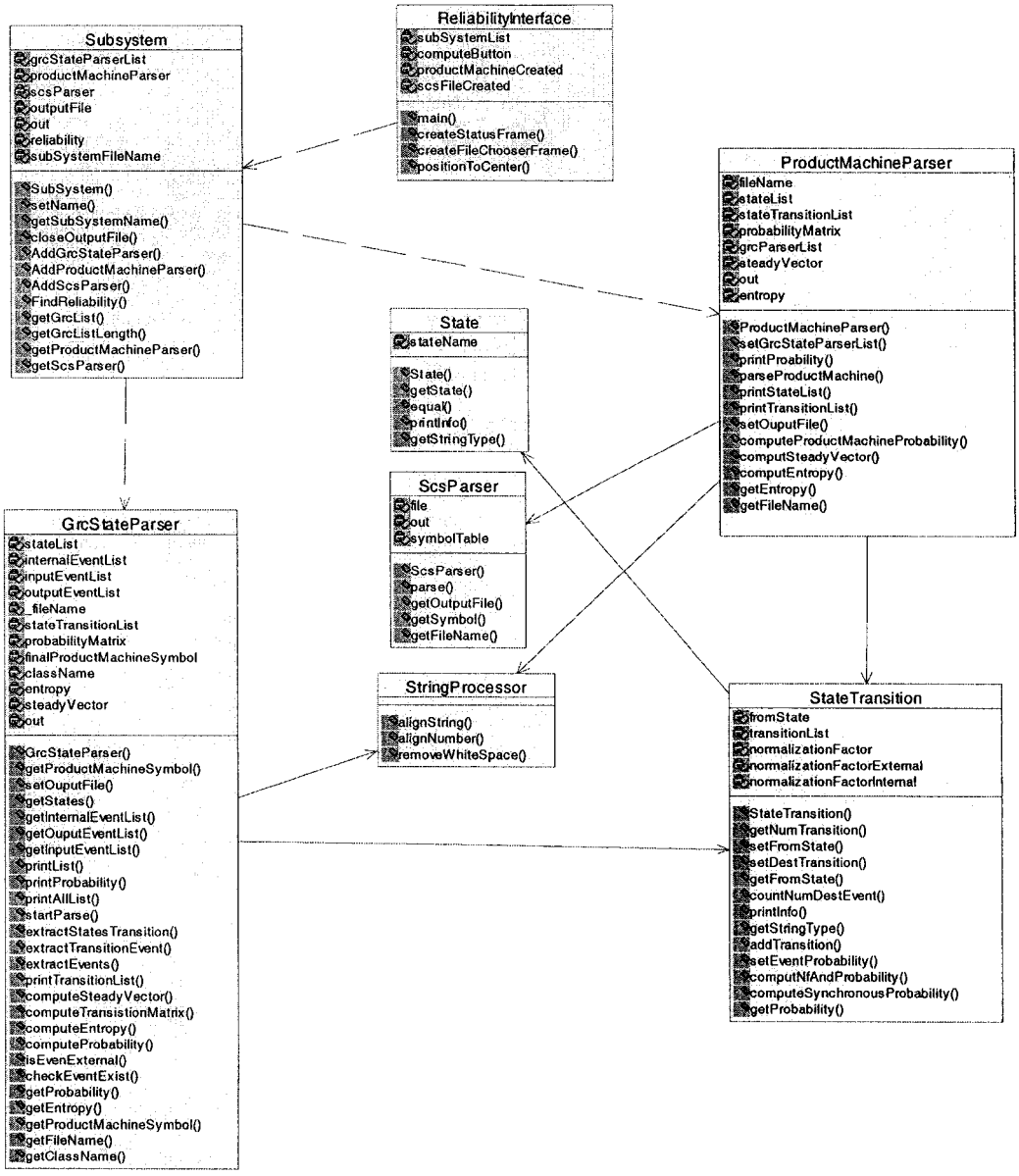

Figure 11: The Class Diagram of the TROM-SRMS

4.6 Sequence Diagram

The sequence diagram (Figure 12) depicts the dynamic aspect of the flow of control in the TROM-SRMS.

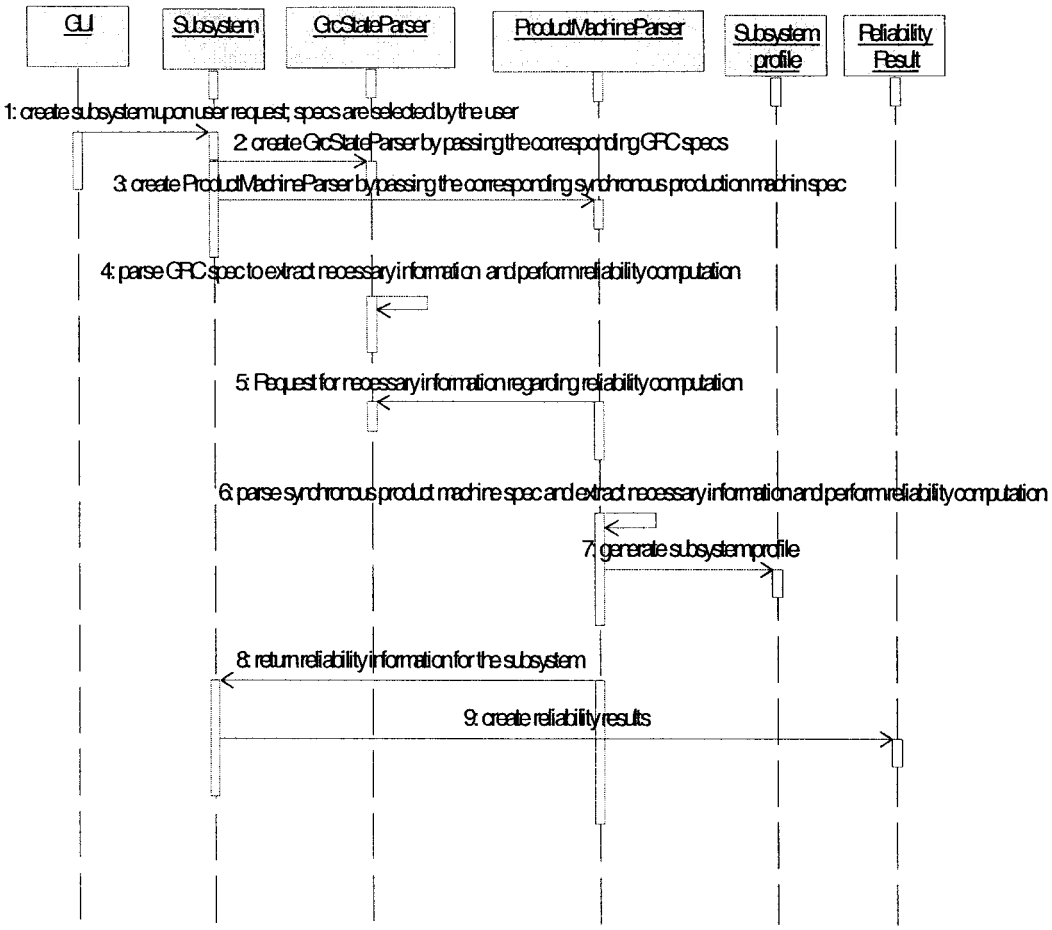

Figure 12: The Sequence Diagram for TROM-SRMS Reliability Computation

Chapter 5. Key Algorithms in TROM-SRMS

This chapter describes key algorithms used in the reliability computation process. The whole process can be divided up into three main algorithms; they are used in the computation of transition matrix of GRC, transition matrix of synchronous product machine, and the reliability. The algorithm of the transition matrix computation of a single GRC computes the probabilities of events. The algorithm of the transition matrix computation of the synchronous product machine uses events' probabilities. The reliability computation algorithm is based on the outputs from the algorithm of the transition matrix computation of the synchronous product machine.

5.1 Algorithm for Transition Matrix Computation of a single GRC

The algorithm accepts a TROM GRC specification as its input. The output of this algorithm includes the probability of each event and the transition matrix of the underlying GRC.

Assumption:

For a transition R where R is denoted as:
R:<Ss, Sd>; e(port condition); enabling condition => post condition
R.Ss is the source state of the transition R
R.Sd is the destination state of the transition R
R.e is the event of the transition

Step 1: *Initialize*

EXOE: external output event list of GRC;

EXIE: external input event list of GRC;

IE: internal event list

SS: states of GRC;

TS: transition specifications of GRC

TP: list of transition probability of a particular event and its associated source and destination state.

TM: transition matrix of GRC

Step 2: *Find events of each type and add them to EXOE, EXIE or IE depending on their type*

Events: event1!@port1, event2@port2, event3@port3, event4 ... eventN

Step 3: *Find all the states and add them to SS*

States: *state1, state2, state3 ... stateN

Step 4: *Find all the transitions and add them to TS*

Ri:<Ss, Sd>; e(port condition); enabling condition => post condition
...
Rn:<Ss, Sd>; e(port condition); enabling condition => post condition

Step 5: *Do the following changes to TP for each transition R in TS*

Step 5.1: compute W, the total number of transitions in TS such that the source states are the same;

Step 5.2: compute U, the total number of transitions in TS such that the source and destination states are the same;

Step 5.2: compute Q, the probability of transition R given by 1/W *U;

Step 5.3: Add R.Ss, R.Sd, R.e and 1/W in TP;

Step 5.4: Add Q in TM;

5.1.1 Pseudo Code

The function that implements the algorithm of transition matrix computation of GRC is *ComputeGrcProbability()*, which includes other sub-functions, whose pseudo code are given separately.

Pre-Condition

- The transition matrix is empty
- The input GRC files are in the correct format

Post-Condition

- The GRC files are parsed and events, transitions and their probabilities are computed
- The transition matrix is filled with transition probabilities

void ComputeGrcProbability()

```
Begin
    find events parseEvent();
    find states parseState();
    find transition parseTransition();
    find probability findTransitionProbability();
End
```

Find events paserEvent()

```
Begin
    event <- read the event list from GRC specification;
    write event into the event list;
End
```

Find states parseState()

Begin

state <- read the state list from GRC specification;

write state into the state list;

End

Find transitions parseTransition()

Begin

transition <- read the transition list from GRC specification;

write transition into the transition list;

End

Find probability findTransitionProbability()

Begin

For every Ti in the transition list Do

Find all transitions Tk such that source state of Ti == source state of Tk;

Total1 <- write the total number of transitions from previous step;

Find all transitions Tk such that the source state of Ti == source state of Tk and destination state of Ti == destination of Tk;

Total2 <- write the total number of transitions from previous step;

Probability <- write combination of Ti and its event probability (1/total);

TransitionProbability <- write the transition probability of Ti (1/total * total2);

End For

End begin

5.2 Algorithm for Transition Matrix Computation of Synchronous Product Machine

The algorithm makes use of the probability of each event computed from algorithm 4.1. The input to the algorithm is the synchronous product machine specification. The output of the algorithm is the transition matrix of the synchronous product machine.

Assumption:

a) For a transition R where R is denoted as:

CR: <<1.s1, 2.s2, 3.s3 ...N.sn>, <1.d1, 2.d2, 3.d3 ... N.dn>>: e;

CR.X.si is the source sub-state of the transition CR, where si is a state in GRC X;

CR.X.di is the destination sub-state of the transition R, where di is a state in GRC X;

CR.e is the transition event;

b) TP_X contains the probability of each event in GRC X. This is obtained from algorithm 4.1

c) NF = 0, NF' = 0, NF" = 0 where they represent normalization factor for external events, internal events and the GRC itself

Step 1: Initialize

TMS: transition matrix of synchronous product machine;

TS: transition specifications of synchronous product machine.

Step 2: Do the following changes to TMS for each transition CR in TS

If CR has an external transition event, perform step 2.1

Step 2.1: compute NF:

Find the transition probability of each sub-transition in CR:

sub-transition 1: GRC 1, s1 -> d1 with event e

sub-transition 2: GRC 2, s2 -> d2 with event e

.

.

.

sub-transition N: GRC N, sN -> dN with event e

Find the **product** P of the transition probabilities of each sub transition that is not equal to 0;

Add NF to the P.

Find all other transitions such that the source state is the same.

For each transition TR of such kind, find the transition probability of each sub-transition:

sub-transition 1: GRC 1, s1 -> d1 with event e

sub-transition 2: GRC 2, s2 -> d2 with event e

.

.

.

sub-transition N: GRC N, sN -> dN with event e

Find the **product** P of the transition probabilities of each sub transition that is not equal to 0;

Add NF to the P.

If CR has an internal transition event, perform step 2.2

Step 2.2: compute NF':

Find the transition probability of each sub-transition in CR:

sub-transition 1: GRC 1, s1 -> d1 with event e

sub-transition 2: GRC 2, s2 -> d2 with event e

.

.

.

sub-transition N: GRC N, sN -> dN with event e

Find the **sum** S of the transition probabilities of each sub-Transition;

Add NF' to the S.

Find all other transitions such that the source state is the same.

For each transition TR of such kind, find the transition probability of each sub-transition:

sub-transition 1: GRC 1, s1 -> d1 with event e

sub-transition 2: GRC 2, s2 -> d2 with event e

.

.

sub-transition N: GRC N, sN -> dN with event e

Find the **sum** S of the transition probabilities of each sub-transition;

Add NF' to the S.

Step 2.3: compute NF'' for transition:

Find NF'' by summing up NF and NF'.

Step 2.4: compute transition probability:

Find the transition probability depending on the type of its transition event:

For internal event, find the sum S of all the transition probability of the sub-states;

For external event, find the product P of all the transition probability of the sub-states that is not equal to 0;

Add S and P to get R;

Find all other transitions that have the same source state and destination state:

For each transition TR of such kind, find the associated transition probability for internal and external event transition:

For internal event, find the sum Su of all the transition probability of the sub-states;

For external event, find the product Pr of all the transition probability of the sub-states that is not equal to 0;

Add Su and Pr to R;

Compute Q, the probability of the transition by R/NF";

Add Q in TMS.

5.2.1 Pseudo Code

The function that implements the algorithm of transition matrix computation of the synchronous product machine is *CompSyncProductProb()*, which includes other sub-functions, whose pseudo code are given separately

Pre-Condition

- The transition matrix is empty
- The input synchronous product machine specification file is in the correct format
- The transition probability of every single transition in the GRCs which form the synchronous product machine is computed

Post-Condition

- The synchronous product machine specification file is parsed and events, transitions and their probabilities are computed
- The transition matrix is filled with transition probabilities

Find transition matrix CompSyncProductProb()

Begin

find transition parseTransition();

find probability findTransitionProbability();

End

Find events paserTransition()

```
Begin
        transition <- read the transition list from Synchronous Product Machine
        specification.
        write transition into the transition list;
End
```

Find probability findTransitionProbability()

```
begin
    while(st <- read in transition from the transition list)
        NF = 0, NF' = 0, NF" = 0, K = 0, J = 0;
        begin
          For every sub-state in st Do
            begin
              if(st.event == external)
                K = probExt(st.substate.source, st.substate.destination,
                    TP_st.Grc_class);
                NF += K;
              elseif(st.event  == internal)
                J = probInt(st.substate.source, st.substate.destination,
                    TP_st.Grc_class);
                NF' += J;
            end begin
          end for
            while(st1 <- read in transition from the rest of transition list)
```

```
begin
  if(st.source_state == st1.source_state)
    for every sub-state in st1 do
      begin
        if(st1.event == external)
          K = probExt(st1.substate.source, st1.substate.destination,
                      st1.Grc_class);
          NF += K;
        elseif(st1.event == internal)
          J = probInt(st1.substate.source, st1.substate.destination,
                      st1.Grc_class);
          NF' += J;
      end begin
      end for
end begin
NF'' = NF + NF';
P = 0;
If(st.event == external)
  P += probExt(st.substate.source, st.substate.destination,
       TP_st.Grc_class);
elseif(st.event  == internal)
  P += probInt(st.substate.source, st.substate.destination,
       TP_st.Grc_class);
```

```
        while(st2 <- read in transition from the rest of transition list)
          begin
            if((st.source_state == st2.source_state)&&
              (st2.destination_state == st2.destination_state))
             if(st.event == external)
               P += probExt(st2.substate.source, st2.substate.destination,
                    Transition_Probability_lists_of_all_the_GRC);
                    /* Transition_Probability_lists_of_all_the_GRC are lists of
                       of each transition in the GRC and its associated transition
                    */
              elseif(st.event  == internal)
               P += probInt(st2.substate.source, st2.substate.destination,
                    Transition_Probability_lists_of_all_the_GRC);
          end begin
        TP = P/NF;
        write TP into transition matrix
      end begin
end begin
/* This function computes the transition probability of a transition in a
synchronous product machine given an external event*/
Find transition probability due to an external event
probability probExt(source_state, dest_state, Grc_transition_prob_list)
begin
```

```
  while(tr <- read transition and its associated probability)
  begin
     if((tr.source_state == source_state) && (tr.dest_state == dest_state))
       if(tr.probability != 0)
         P *= tr.probability;
  end begin
  return P;
end begin

/* This function computes the transition probability of a transition in a
synchronous product machine given an internal event*/
```

Find transition probability due to an internal event

```
probability probInt(source_state, dest_state, Grc_transition_prob_list)
begin
  while(tr <- read transition and its associated probability)
  begin
     if((tr.source_state == source_state) && (tr.dest_state == dest_state))
       P += tr.probability;
  end begin
  return P;
end begin
```

5.3 Algorithm for Reliability Computation

The algorithm makes use of the transition matrix of each GRC and synchronous product machine. The inputs to the algorithm are the transition matrices of GRC and synchronous product machine. The output of the algorithm is the reliability of the subsystem.

Assumption:

a) TM_X contains the transition matrix of the GRC X obtained from algorithm 4.1

b) TMS contains the transition matrix of the synchronous product machine obtained from algorithm 4.2

Step 1: Find Steady Vector for each GRC

For each transition matrix in TM_X, compute the corresponding steady vector.

Step 2: Find Steady Vector for synchronous product machine

Compute steady vector for TMS.

Step 3: Find Entropy for each GRC

Based on steady vector, compute the entropy with

$$H = - \sum_{i=1}^{n} V_i \sum_{j=1}^{n} P_{ij} \log_2 P_{ij}$$

Note that when P_{ij} ***is equal to 0, the term*** $P_{ij} \log_2 P_{ij}$ ***is considered 0.***

Step 4: Find Entropy for Synchronous Product Machine

Based on steady vector, compute the entropy with

$$H = - V\sum_{j=1}^{n} P_{ij} log_2 P_{ij}$$

Note that when P_{ij} ***is equal to 0, the term*** $P_{ij} log_2 P_{ij}$ ***is considered 0.***

Step 5: Find Reliability for the Sub-System

Based on entropy of GRC and synchronous product machine, the reliability is computed as *Reliability(Sub-System)* = $\sum_{i=1}^{n} H_i - H$

5.3.1 Pseudo Code

The function that implements the algorithm of reliability computation is *ComputeReliability()* which includes other sub-functions, whose pseudo codes are given separately.

Pre-Condition

- The transition matrices of GRCs are computed
- The transition matrix of synchronous product machine is computed

Post-Condition

- The reliability of the subsystem is computed

ComputeReliability()

```
begin
    find GRC steady vector findGrcSteadyVector()
    find Synchronous Product Machine steady vector findSynchSteadyVector()
    find GRC entropy findGrcEntropy()
    find Synchronous Product Machine entropy findSynchEntropy()
    find reliability findReliability()
```

```
end

findGrcSteadyVector()

begin
        for each transition matrix TM that belongs to GRC X in TM_X do
                solve equation of the form x + y + z + . . . = 1;
                where  [x y z ...] TM = [x y z ...];
                steadyVectorList_X <- write the result in the corresponding GRC
                                        steady vector list
        end for
end begin

findSynchSteadyVector()

begin
        for TMS of the synchronous product machine do
                solve equation of the form x + y + z + . . . = 1;
                where  [x y z ...] TMS = [x y z ...];
                steadyVectorList_X <- write the result in the corresponding
                                        synchronous product machine steady vector
                                        list
        end for
end begin

findGrcEntropy()

begin
```

while(V <- read from GRC steady vector list)

begin

P = transition matrix of GRC associated to V, obtained from TM_X;

Entropy += $V \sum_{j=1}^{n} P_{ij} log_2 P_{ij}$;

end begin

E <- write entropy

End begin

findSynchEntropy()

begin

V <- read from Synchronous Product Machine steady vector list

P = transition matrix of the Synchronous Product Machine, obtained from TMS;

Entropy = $V \sum_{j=1}^{n} P_{ij} \log_2 P_{ij}$;

ES <- write entropy

End begin

findReliability()

begin

result = E – ES;

Reliability <- write result into reliability;

end begin

Chapter 6. Gate-Train-Controller Case Study

In this chapter, we introduce the Gate-Train-Controller problem that is considered as a benchmark example by researchers in the real-time system communities. The case study will demonstrate the theoretical computation results of the reliability computation algorithms introduced in Chapter 3. The results will be used to compare to the results generated by the software as a validation scheme.

6.1 Description of the Problem

For the purpose of algorithm and software validation, this section only presents a limited amount of information that suffices our purpose. For more detail description of the problem, please refer to [Orm02].

The Gate-Train-Controller is a real-time time constrained system consisting of Gate, Train and Controller subsystems. Each of these subsystems is mapped to a TROM GRC object and their corresponding GRC specifications. Statecharts and the synchronous product machine specification are shown in Figures 13, 14, 15, 16, 17 and 18:

```
Class Train [@R]
   Events: Near!@R, Out, Exit!@R, In
   States: *idle, cross, leave, toCross
   Attributes: cr:@C
   Traits:
   Attribute-Function:
      idle →  {}; cross →  {};
      leave →  {}; toCross →  {cr};
   Transition-Specifications:
      R1: <idle,toCross>; Near(true); true=>cr'=pid;
      R2: <cross,leave>; Out(true); true => true;
      R3: <leave,idle>; Exit(pid = cr); true => true;
      R4: <toCross,cross>; In(true); true => true;
   Time-Constraints:
      TCvar2: R1, Exit, [0, 6], {};
      TCvar1: R1, In, [2, 4], {};
end
```

Figure 13: Formal Specification of Class Train

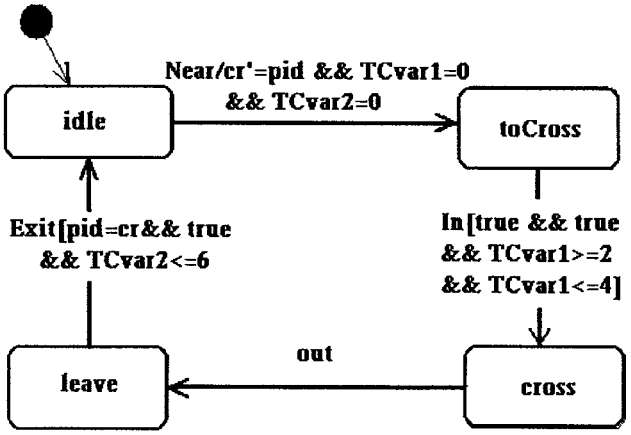

Figure 14: State Chart of Class Train

```
Class Controller [@P, @Y]
    Events: Lower!@Y, Near?@P, Raise!@Y, Exit?@P
    States: *idle, activate, deactivate, monitor
    Attributes: inSet:PSet
    Traits: Set[@P,PSet]
    Attribute–Function:
        activate →  {inSet}; deactivate →  {inSet};
        monitor →  {inSet}; idle →  {};
    Transition–Specifications:
        R1: <activate,monitor>; Lower(true);
            true => true;
        R2: <activate,activate>; Near(NOT(member(pid,inSet)));
            true => inSet' = insert(pid,inSet);
        R3: <deactivate,idle>; Raise(true);
            true => true;
        R4: <monitor,deactivate>; Exit(member(pid,inSet));
            size(inSet) = 1 => inSet' = delete(pid,inSet);
        R5: <monitor,monitor>; Exit(member(pid,inSet));
            size(inSet) > 1 => inSet' = delete(pid,inSet);
        R6: <monitor,monitor>; Near(!(member(pid,inSet)));
            true => inSet' = insert(pid,inSet);
        R7: <idle,activate>; Near(true);
            true => inSet' = insert(pid,inSet);
    Time–Constraints:
        TCvar1: R7, Lower, [0, 1], {};
        TCvar2: R4, Raise, [0, 1], {};
end
```

Figure 15: Formal Specification of Class Controller

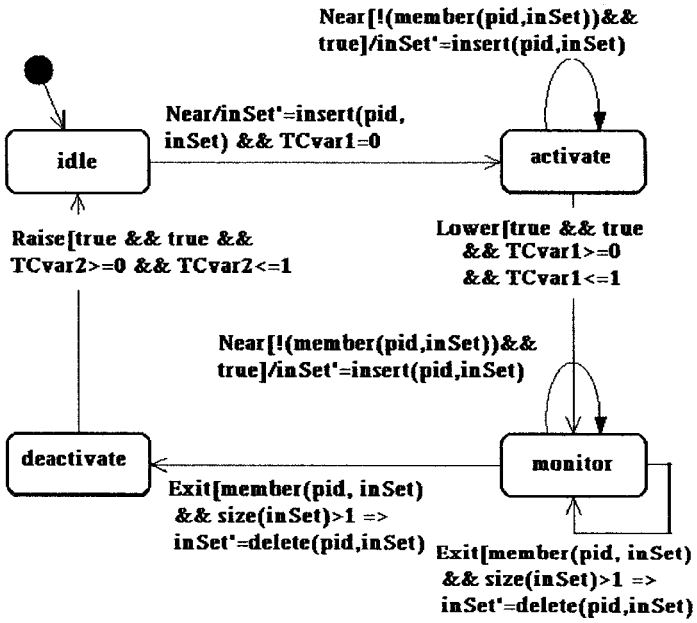

Figure 16: State Chart of Class Controller

```
Class Gate [@S]
    Events: Lower?@S, Down, Up, Raise?@S
    States: *opened, toClose, toOpen, closed
    Attributes:
    Traits:
    Attribute–Function:
        opened →  {}; toClose →  {};
        toOpen →  {}; closed →  {};
    Transition–Specifications:
        R1: <opened,toClose>; Lower(true); true => true;
        R2: <toClose,closed>; Down(true); true => true;
        R3: <toOpen,opened>; Up(true); true => true;
        R4: <closed,toOpen>; Raise(true); true => true;
    Time–Constraints:
        TCvar1: R1, Down, [0, 1], {};
        TCvar2: R4, Up, [1, 2], {};
end
```

Figure 17: Formal Specification of Class Gate

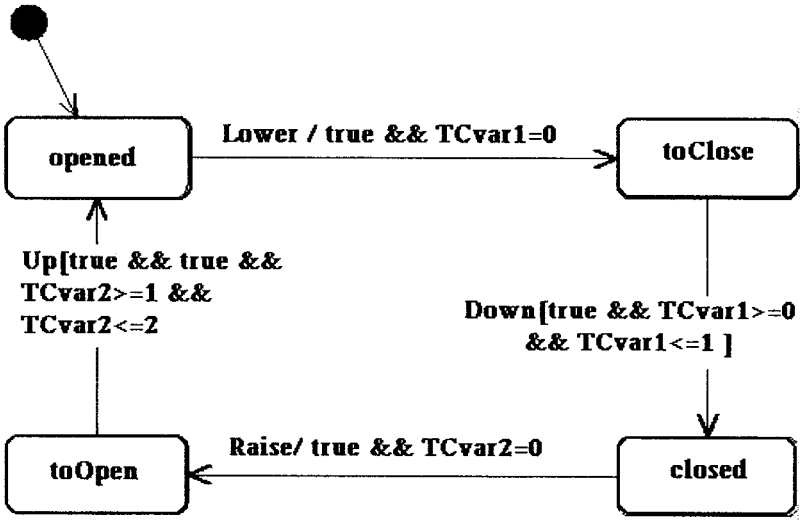

Figure 18: State Chart of Class Gate

The synchronous product machine specification is shown below:

Class Name : C_T_G

State List:

<<G.opened, C.idle, T.idle>, true>

<<G.opened, C.activate, T.toCross>, false>

<<G.toClose, C.monitor, T.toCross>, false>

<<G.opened, C.activate, T.cross>, false>

<<G.closed, C.monitor, T.toCross>, false>

<<G.toClose, C.monitor, T.cross>, false>

<<G.opened, C.activate, T.leave>, false>

<<G.closed, C.monitor, T.cross>, false>

<<G.toClose, C.monitor, T.leave>, false>

<<G.closed, C.monitor, T.leave>, false>

<<G.toClose, C.deactivate, T.idle>, false>

<<G.closed, C.deactivate, T.idle>, false>

<<G.toOpen, C.idle, T.idle>, false>

Transition Spec List:

CR-0 <<G.opened, C.idle, T.idle>, <G.opened, C.activate, T.toCross>> : C.Near;

CR-1 <<G.opened, C.activate, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C/G.Lower;

CR-2 <<G.opened, C.activate, T.toCross>, <G.opened, C.activate, T.cross>> : T.In;

CR-3 <<G.toClose, C.monitor, T.toCross>, <G.closed, C.monitor, T.toCross>> : G.Down;

CR-4 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.cross>> : T.In;

CR-5 <<G.opened, C.activate, T.cross>, <G.toClose, C.monitor, T.cross>> : C/G.Lower;

CR-6 <<G.opened, C.activate, T.cross>, <G.opened, C.activate, T.leave>> : T.Out;

CR-7 <<G.closed, C.monitor, T.toCross>, <G.closed, C.monitor, T.cross>> : T.In;

CR-8 <<G.toClose, C.monitor, T.cross>, <G.closed, C.monitor, T.cross>> : G.Down;

CR-9 <<G.toClose, C.monitor, T.cross>, <G.toClose, C.monitor, T.leave>> : T.Out;

CR-10 <<G.opened, C.activate, T.leave>, <G.toClose, C.monitor, T.leave>> : C/G.Lower;

CR-11 <<G.closed, C.monitor, T.cross>, <G.closed, C.monitor, T.leave>> : T.Out;

CR-12 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.deactivate, T.idle>> : C.Exit;

CR-13 <<G.toClose, C.monitor, T.leave>, <G.closed, C.monitor, T.leave>> : G.Down;

CR-14 <<G.closed, C.monitor, T.leave>, <G.closed, C.deactivate, T.idle>> : C.Exit;

CR-15 <<G.toClose, C.deactivate, T.idle>, <G.closed, C.deactivate, T.idle>> : G.Down;

CR-16 <<G.closed, C.deactivate, T.idle>, <G.toOpen, C.idle, T.idle>> : C/G.Raise;

CR-17 <<G.toOpen, C.idle, T.idle>, <G.opened, C.idle, T.idle>> : G.Up;

CR-18 <<G.opened, C.activate, T.toCross>, <G.opened, C.activate, T.toCross>> : C.Near;

CR-20 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C.Exit;

CR-21 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C.Near;

CR-22 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.monitor, T.leave>> : C.Exit;

CR-23 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.monitor, T.leave>> : C.Near;

Figure 19: Formal Specification for Synchronous Product Machine of Gate-Train-Controller

```
SCS tgc-collab
        Includes:
        Instantiate:
                G::Gate[@S:1];
                T::Train[@C:1];
                C::Controller[@G:1, @P:1];
        Configure:
                controller1. @G1:@G <-> gate1.@S1:@S;
                controller1. @P1:@P <-> train1.@C1:@C;
end
```

Figure 20: Formal Specification for one train – one gate – one controller subsystem (SCS)

6.2 GRC related computations

This section presents the computation of transition matrix of each GRC in the Gate-Train-Controller system based on the algorithm presented in Chapter 3.

6.2.1 Gate

6.2.1.1 Transition Matrix

GRC Gate has the following transitions:

R1: <opened,toClose>; Lower(true); true => true;

R2: <toClose,closed>; Down(true); true => true;

R3: <toOpen,opened>; Up(true); true => true;

R4: <closed,toOpen>; Raise(true); true => true;

- Since there's only one transition with the source state <opened>, transition probability of R1 is 1. All other transitions with source state <opened> in the transition matrix are 0.
- Since there's only one transition with the source state <toClose>, transition probability of R2 is 1. All other transitions with source state <toClose> in the transition matrix are 0.
- Since there's only one transition with the source state <toOpen>, transition probability of R3 is 1. All other transitions with source state <toOpen> in the transition matrix are 0.
- Since there's only one transition with the source state <closed>, transition probability of R4 is 1. All other transitions with source state <closed> in the transition matrix are 0.

The following table shows the transition matrix of the Gate class:

Table 1: Transition Matrix of Gate Class

	opened	toClose	ToOpen	closed
opened	0	1	0	0
toClose	0	0	0	1
toOpen	1	0	0	0
closed	0	0	1	0

The following table shows the transition probability of each event in Gate class:

Table 2. Probability of each transition in Gate Class

Transition : Event	Probability
<opened,toClose> : Lower	1
<toClose,closed> : Down	1
<toOpen,opened> : Up	1
<closed,toOpen> : Raise	1

6.2.1.2 Steady Vector

Need to find a vector such that

$$|wxyz| \begin{bmatrix} 0100 \\ 0001 \\ 1000 \\ 0010 \end{bmatrix} = |wxyz|$$

Converting the above formula to a set of equations give:

$0W + 0X + 1Y + 0Z = W$

$1W + 0X + 0Y + 0Z = X$

$0W + 0X + 0Y + 1Z = Y$

$0W + 1X + 0Y + 0Z = Z$

$W + X + Y + Z = 1$

By solving the equations we get:

$W = 0.25$

$X = 0.25$

$Y = 0.25$

$Z = 0.25$

[0.25 0.25 0.25 0.25] ... Steady Vector of Gate

6.2.1.3 Entropy

$$\mathbf{H}_{Gate} = -((0.25 * (0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0)) +$$

$$(0.25 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1)) +$$

$$(0.25 * (1 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$$

$$(0.25 * (0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0)))$$

= 0

6.2.2 Pseudo Code

6.2.2.1 Transition Matrix

GRC Train has the following transitions:

R1:<idle,toCross>; Near(true); true => cr'=pid;

R2:<cross,leave>; Out(true); true => true;

R3: <leave,idle>; Exit(pid=cr); true => true;

R4: <toCross,cross>; In(true); true => true;

- Since there's only one transition with the source state <idle>, transition Probability of R1 is 1. All other transitions with source state <idle> in the transition matrix are 0.
- Since there's only one transition with the source state <cross>, transition Probability of R2 is 1. All other transitions with source state <cross> in the transition matrix are 0.
- Since there's only one transition with the source state <leave>, transition Probability of R3 is 1. All other transitions with source state <leave> in the transition matrix are 0.
- Since there's only one transition with the source state <toCross>, transition Probability of R4 is 1. All other transitions with source state <toCross> in the transition matrix are 0.

The following table shows the transition matrix of the Train class:

Table 3: Transition Matrix of Train Class

	Idle	Cross	leave	toCross
Idle	0	0	0	1
cross	0	0	1	0
leave	1	0	0	0

toCross	0	1	0	0

The following table shows the transition probability of each event in Train class:

Table 4: Probability of each transition in Train Class

Transition : Event	Probability
<idle,toCross> : Near	1
<cross,leave> : Out	1
<leave,idle> : Exit	1
<toCross,cross> : In	1

6.2.2.2 Steady Vector

Need to find a vector such that

$$|wxyz| \begin{bmatrix} 0001 \\ 0010 \\ 1000 \\ 0100 \end{bmatrix} = |wxyz|$$

Converting the above formula to a set of equations give:

0W + 0X + 1Y + 0Z = W

0W + 0X + 0Y + 1Z = X

0W + 1X + 0Y + 0Z = Y

1W + 0X + 0Y + 0Z = Z

W + X + Y + Z = 1

By solving the equations we get:

W = 0.25

X = 0.25

Y = 0.25

Z = 0.25

[0.25 0.25 0.25 0.25] ... Steady Vector of Train

6.2.2.3 Entropy

$$H_{Train} = -((0.25 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1)) +$$

$$(0.25 * (0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0)) +$$

$$(0.25 * (1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$$

$$(0.25 * (0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0)))$$

$$= 0$$

6.2.3 Controller

6.2.3.1 Transition Matrix

GRC Controller has the following transitions:

R1:<activate,monitor>; Lower(true); true => true;

R2:<activate,activate>; Near(!(member(pid,inSet))); true => inSet'=insert(pid,inSet);

R3:<deactivate,idle>; Raise(true); true => true;

R4:<monitor,deactivate>; Exit(member(pid,inSet)); size(inSet)=1 =>

inSet'=delete(pid,inSet);

R5:<monitor,monitor>; Exit(member(pid,inSet)); size(inSet)>1 =>

inSet'=delete(pid,inSet);

R6:<monitor,monitor>; Near(!member(pid,inSet))); true => inSet'=insert(pid,inSet);

R7:<idle,activate>; Near(true); true => inSet'=insert(pid,inSet);;

R1 & R2:

Since these two transitions have the same source state <activate> but different destination state, the probability of each is 1/2. All other transitions with source state <activate> in the transition matrix are 0.

R3:

Since this is the only one transition with the source state <deactivate>, transition probability of R3 is 1. All other transitions with source state <deactivate> in the transition matrix are 0.

R4, R5 & R6:

Since these three transitions have the same the source state <monitor>, probability of each transition is 1/3. Due to the fact that R5 and R6 both have the same destination state, we sum up the transition probability of both in the transition matrix, which gives 2/3 for transition <monitor, monitor>.

R7:

Since there is only one transition with the source state <idle>, transition Probability of R7 is 1. All other transitions with source state <idle> in the transition matrix are 0.

The following table shows the transition matrix of the Controller class:

Table 5. Transition Matrix of Controller Class

	activate	deactivate	monitor	idle
activate	1/2	0	1/2	0
deactivate	0	0	0	1
monitor	0	1/3	2/3	0
idle	1	0	0	0

Table 6. Probability of each transition in Controller Class

Transition : Event	Probability
<activate,monitor> : Lower	1/2
<activate,activate> : Near	1/2
<deactivate,idle> : Raise	1
<monitor,deactivate> : Exit	1/3
<monitor,monitor> : Exit	1/3
<monitor,monitor> : Near	1/3
<idle,activate> : Near	1

6.2.3.2 Steady Vector

Need to find a vector such that

$$|wxyz| \begin{bmatrix} 1/2 & 0/1 & 1/2 & 0/1 \\ 0/1 & 0/1 & 0/1 & 1/1 \\ 0/1 & 1/3 & 2/3 & 0/1 \\ 1/1 & 0/1 & 0/1 & 0/1 \end{bmatrix} = |wxyz|$$

Converting the above formula to a set of equations give:

1/2W + 0X + 0Y + 1Z = W

0W + 0X + 1/3Y + 0Z = X

1/2W + 0X + 2/3Y + 0Z = Y

0W + 1X + 0Y + 0Z = Z

W + X + Y + Z = 1

By solving the equations we get:

W = 0.286

X = 0.143

Y = 0.429

Z = 0.143

[0.286 0.143 0.429 0.143] ... Steady Vector of Controller

6.2.3.3 Entropy

$$\mathbf{H}_{Controller} = -((0.286 * (0.5 * \log_2 0.5 + 0 * \log_2 0 + 0.5 * \log_2 0.5 + 0 * \log_2 0)) +$$

$$(0.143 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1)) +$$

$$(0.429 * (0 * \log_2 0 + 0.33 * \log_2 0.33 + 0.67 * \log_2 0.67 + 0 * \log_2 0)) +$$

$$(0.143 * (1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)))$$

$$= -(-0.286 + 0 + -0.3925 + 0) = -(-0.6785) = 0.6785$$

6.3 Synchronous Product Machine related computations

6.3.1 Transition Matrix

The synchronous product machine has the following transitions (see also Figure 19):

CR-0 <<G.opened, C.idle, T.idle>, <G.opened, C.activate, T.toCross>> : C.Near;

CR-1 <<G.opened, C.activate, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C/G.Lower;

CR-2 <<G.opened, C.activate, T.toCross>, <G.opened, C.activate, T.cross>> : T.In;

CR-3 <<G.toClose, C.monitor, T.toCross>, <G.closed, C.monitor, T.toCross>> : G.Down;

CR-4 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.cross>> : T.In;

CR-5 <<G.opened, C.activate, T.cross>, <G.toClose, C.monitor, T.cross>> : C/G.Lower;

CR-6 <<G.opened, C.activate, T.cross>, <G.opened, C.activate, T.leave>> : T.Out;

CR-7 <<G.closed, C.monitor, T.toCross>, <G.closed, C.monitor, T.cross>> : T.In;

CR-8 <<G.toClose, C.monitor, T.cross>, <G.closed, C.monitor, T.cross>> : G.Down;

CR-9 <<G.toClose, C.monitor, T.cross>, <G.toClose, C.monitor, T.leave>> : T.Out;

CR-10 <<G.opened, C.activate, T.leave>, <G.toClose, C.monitor, T.leave>> : C/G.Lower;

CR-11 <<G.closed, C.monitor, T.cross>, <G.closed, C.monitor, T.leave>> : T.Out;

CR-12 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.deactivate, T.idle>> : C.Exit;

CR-13 <<G.toClose, C.monitor, T.leave>, <G.closed, C.monitor, T.leave>> : G.Down;

CR-14 <<G.closed, C.monitor, T.leave>, <G.closed, C.deactivate, T.idle>> : C.Exit;

CR-15 <<G.toClose, C.deactivate, T.idle>, <G.closed, C.deactivate, T.idle>> : G.Down;

CR-16 <<G.closed, C.deactivate, T.idle>, <G.toOpen, C.idle, T.idle>> : C/G.Raise;

CR-17 <<G.toOpen, C.idle, T.idle>, <G.opened, C.idle, T.idle>> : G.Up;

CR-18 <<G.opened, C.activate, T.toCross>, <G.opened, C.activate, T.toCross>> : C.Near;

CR-20 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C.Exit;

CR-21 <<G.toClose, C.monitor, T.toCross>, <G.toClose, C.monitor, T.toCross>> : C.Near;

CR-22 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.monitor, T.leave>> : C.Exit;

CR-23 <<G.toClose, C.monitor, T.leave>, <G.toClose, C.monitor, T.leave>> : C.Near;

CR-0

Since there's only one transition with the source state <G.opened, C.idle, T.idle>, the transition probability of CR-0 is computed as the following:

P1 = Prob(<G.opened, G.opened>:Near) = 0

P2 = Prob(<C.idle, C.activate>:Near) = 1

P3 = Prob(<T.idle, T.toCross>:Near) = 1

NF Computation:

Event *Near* is an external event and triggers transition CR-0; therefore the NF is computed as the following:

P2 * P3 = 1 * 1 = 1

NF' Computation:

Since there is no internal event involved in the transition, NF' is equal to 0.

NF'' Computation:

NF'' = NF' + NF = 0 + 1 = 1

Transition Probability:

TransProb(CR-0) = (P2 * P3)/NF'' = (1 * 1)/1 = 1

CR-1, CR-2, CR-18

There are three transitions that start with the source state <G.opened, C.activate, T.toCross>.

The transition probabilities of CR-1, CR-2 and CR-18 are computed as the following:

The following probabilities are associated to CR-1:

P1 = Prob(<G.opened, G.toClose>:Lower) : 1

P2 = Prob(<C.activate, C.monitor>:Lower) : 1/2

P3 = Prob(<T.toCross, T.toCross>:Lower) : 0

The following probabilities are associated to CR-2:

P4 = Prob(<G.opened, G.opened>:In) : 0

P5 = Prob(<C.activate, C.activate>:In) : 0

P6 = Prob(<T.toCross, T.cross>:In) : 1

The following probabilities are associated to CR-18:

P7=Prob(<G.opened, G.opened>:Near) : 0

P8=Prob(<C.activate, C.activate>:Near) : 1/2

P9=Prob(<T.toCross, T.toCross>:Near) : 0

NF Computation:

Events *Lower* and *Near* are external events and trigger transition CR-1 and CR-18; therefore the NF is computed as the following:

(P1 * P2) + P8 = (1 * 1/2) + 1/2 = 1

NF' Computation:

Event *In* is an internal event and triggers transition CR-2; therefore the NF' is computed as the following:

P4 + P5 + P6 = 0 + 0 +1 = 1

NF'' Computation:

NF'' = NF' + NF = 1 + 1 = 2

Transition Probability:

TransProb(CR-1) = (P1*P2)/NF" = (1*1/2)/2 = 1/4

TransProb(CR-2) = (P4+P5+P6)/NF" = (0+0+1)/2 = 1/2

TransProb(CR-18) = (P8) /NF" = (1/2)/2 = ¼

CR-3, CR-4, CR-20 and CR-21

There are four transitions that start with the source state <G.toClose, C.monitor, T.toCross>.

The transition probabilities of CR-3, CR-4, CR-18 and CR-21 are computed as the following:

The following probabilities are associated to CR-3:

P1 = Prob(<G.toClose, G.Close>:Down) : 1

P2 = Prob(<C.monitor, C.monitor>:Down) : 0

P3 = Prob(<T.toCross, T.toCross>:Down) : 0

The following probabilities are associated to CR-4:

P4 = Prob(<G.toClose, G.toClose>:In) : 0

P5 = Prob(<C.monitor, C.monitor>:In) : 0

P6 = Prob(<T.toCross, T.cross>:In) :

The following probabilities are associated to CR-20:

P7 = Prob(<G.toClose, G.toClose>:Exit) : 0

P8 = Prob(<C.monitor, C.monitor>:Exit) : 1/3

P9 = Prob(<T.toCross, T.toCross>:Exit) : 0

The following probabilities are associated to CR-21:

P10 = Prob(<G.toClose, G.toClose>:Near) : 0

P11 = Prob(<C.monitor, C.monitor>:Near) : 1/3

P12 = Prob(<T.toCross, T.toCross>:Near) : 0

NF Computation:

Events *Exit* and *Near* are external events and trigger transition CR-20 and CR-21; therefore the NF is computed as the following:

P8 + P11 = 1/3 + 1/3 = 2/3

NF' Computation:

Events *In* and *Down* are internal events and trigger transition CR-3 and CR-4; therefore the NF' is computed as the following:

P1 + P6 = 1 + 1 = 2

NF'' Computation:

NF'' = NF' + NF = 2 + 2/3 = 8/3

Transition Probability:

TransProb(CR-3) = P1/NF'' = 1/(8/3) = 3/8

TransProb(CR-4) = P6/NF'' = 1/(8/3) = 3/8

TransProb(CR-20) = P8/NF'' = (1/3)/(8/3) = 1/8

TransProb(CR-21) = P11/NF'' = (1/3)/(8/3) = 1/8

Notice that CR-20 and CR-21 have the same transition states, therefore the transition probability in the transition matrix is the sum of both transitions, i.e.

TransMatrixProb(CR-20 and CR-21) = TransProb(CR-20) + TransProb(Cr-21) = 1/8 + 1/8 = 2/8

CR-5 and CR-6

There are two transitions that start with the source state <G.opened, C.activate, T.cross>.

The transition probabilities of CR-5 and CR-6 are computed as the following:

The following probabilities are associated to CR-5:

P1 = Prob(<G.opened, G.toClose>:Lower) : 1

P2 = Prob(<C.activate, C.monitor>:Lower) : 1/2

P3 = Prob(<T.cross, T.cross>:Lower) : 0

The following probabilities are associated to CR-6:

P4 = Prob(<G.opened, G.opened>:Out) : 0

P5 = Prob(<C.activate, C.activate>:Out) : 0

P6 = Prob(<T.cross, T.leave>:Out) : 1

NF Computation:

Event *Lower* is an external event and trigger transition CR-5; therefore the NF is computed as the following:

(P1 * P 2) = 1 * 1/2 = 1/2

NF' Computation:

Event *Out* is an internal event and triggers transition CR-6; therefore the NF' is computed as the following:

P4 + P5 + P6 = 0 + 0 + 1 = 1

NF'' Computation:

NF'' = NF' + NF = 1 + 1/2 = 3/2

Transition Probability:

TransProb(CR-5) = (P1*P2)/NF" = (1/2)/(3/2) = 1/3

TransProb(CR-6) = (P4+P5+P6)/NF" = (0+0+1)/(3/2) = 2/3

CR-7

There is only one transition that starts with the source state <G.closed, C.monitor, T.toCross>.

The transition probability of CR-7 is computed as follows:

The following probabilities are associated to CR-7:

P1 = Prob(<G.closed, G.closed>:In) : 0

P2 = Prob(<C.monitor, C.monitor>:In) : 0

P3 = Prob(<T.toCross, T.cross>:In) : 1

NF Computation:

There is no external event involved in the transition, therefore NF is equal to 0.

NF' Computation:

Event *In* is an internal event and triggers transition CR-7; therefore the NF' is computed as the following:

P1 + P 2 + P3 = 0 + 0 + 1 = 1

NF'' Computation:

NF" = NF' + NF = (P1+P2+P3) + 0 = (0+0+1) + 0 =1

Transition Probability:

TransProb(CR-7) = (P1+P2+P3)/NF" = (0+0+1)/1 = 1

CR-8 and CR-9

There are two transitions that start with the source state <G.toClose, C.monitor, T.cross>.

The transition probabilities of CR-8 and CR-9 are computed as the following:

The following probabilities are associated to CR-8:

P1 = Prob(<G.toClose, G.closed>:Down) : 1

P2 = Prob(<C.monitor, C.monitor>:Down) : 0

P3 = Prob(<T.cross, T.cross>:Down) : 0

The following probabilities are associated to CR-9:

P4 = Prob(<G.toClose, G.toClose>:Out) : 0

P5 = Prob(<C.monitor, C.monitor>:Out) : 0

P6 = Prob(<T.cross, T.leave>:Out) : 1

NF Computation:

There is no external event involved in the transitions, therefore NF is equal to 0.

NF' Computation:

Event *Down* and *Out* are internal events and trigger transition CR-8 and CR-9; therefore the NF' is computed as the following:

(P1 + P 2 + P3) + (P4 + P5 + P6) = (1 + 0 + 0) + (0 + 0 + 1) = 2

NF'' Computation:

NF'' = NF' + NF = ((P1+P2+P3) + (P4 + P5 + P6)) + 0 = ((1+0+0) + (0 + 0 + 1) + 0 =2

Transition Probability:

TransProb(CR-8) = (P1+P2+P3)/NF'' = (1+0+0)/2 = 1/2

TransProb(CR-9) = (P4+P5+P6)/NF'' = (0+0+1)/2 = 1/2

CR-10

There is only one transition that starts with the source state <G.opened, C.activate, T.leave>.

The transition probability of CR-10 is computed as the following:

The following probabilities are associated to CR-10:

P1 = Prob(<G.opened, G.toClose>:Lower) : 1

P2 = Prob(<C.activate, C.monitor>:Lower) : 1/2

P3 = Prob(<T.leave, T.leave>:Lower) : 0

NF Computation:

Event *Lower* is an external event and triggers transition CR-10; therefore the NF is computed as the following:

P1 * P2 = 1 * 1/2 = 1/2

NF' Computation:

There is no internal event involved in the transition, therefore NF' is equal to 0.

NF'' Computation:

NF'' = NF' + NF = 0 + (P1*P2) = 0 + 1/2 = 1/2

Transition Probability:

TransProb(CR-10) = (P1*P2)/NF'' = (1*1/2)/1/2 = 1

CR-11

There is only one transition that starts with the source state <G.closed, C.monitor, T.cross>.

The transition probability of CR-11 is computed as the following:

The following probabilities are associated to CR-11:

P1 = Prob(<G.closed, G.closed>:Out) : 0

P2 = Prob(<C.monitor, C.monitor>:Out) : 0

P3 = Prob(<T.cross, T.leave>:Out) : 1

NF Computation:

There is no external event involved in the transition, therefore NF is equal to 0.

NF' Computation:

Event *Out* is an internal event and triggers transition CR-11; therefore the NF' is computed as the following:

P1 + P2 + P3 = 0 + 0 + 1 = 1

NF'' Computation:

NF'' = NF' + NF = (P1+P2+P3) + 0 = 1 + 0 = 1

Transition Probability:

TransProb(CR-11) = (P1+P2+P3)/NF'' = (0+0+1)/1 = 1

CR-12, CR-13, CR22 and CR-23

There are four transitions that start with the source state <G.toClose, C.monitor, T.leave>.

The transition probabilities of CR-12, CR-13, CR-22 and CR-23 are computed as the following:

The following probabilities are associated to CR-12:

P1 = Prob(<G.toClose, G.toClose>:Exit) : 0

P2 = Prob(<C.monitor, C.deactivate>:Exit) : 1/3

P3 = Prob(<T.leave, T.idle>:Exit) : 1

The following probabilities are associated to CR-13:

P4 = Prob(<G.toClose, G.closed>:Down) : 1

P5 = Prob(<C.monitor, C.monitor>:Down) : 0

P6 = Prob(<T.leaave, T.leave>:Down) : 0

The following probabilities are associated to CR-22:

P7 = Prob(<G.toClose, G.toClose>:Exit) : 0

P8 = Prob(<C.monitor, C.monitor>:Exit) : 1/3

P9 = Prob(<T.leave, T.leave>:Exit) : 0

The following probabilities are associated to CR-23:

P10 = Prob(<G.toClose, G.toClose>:Near) : 0

P11 = Prob(<C.monitor, C.monitor>:Near) : 1/3

P12 = Prob(<T.leaave, T.leave>:Near) : 0

NF Computation:

Events *Exit* and *Near* are external events and trigger transition CR-12, CR-22 and CR-23; therefore the NF is computed as the following:

(P2 * P 3) + (P8) + (P11) = (1/3 * 1) + (1/3) + (1/3) = 3/3

NF' Computation:

Event *Down* is an internal event and triggers transition CR-13; therefore the NF' is computed as the following:

P4 + P5 + P6 = 1 + 0 + 0 = 1

NF'' Computation:

NF'' = NF' + NF = 1 + 3/3 = 6/3 = 2

Transition Probability:

TransProb(CR-12) = (P2*P3)/NF'' = (1/3*1)/2 = 1/6

TransProb(CR-13) = (P4+P5+P6)/NF'' = (1+0+0)/2 = 1/2

TransProb(CR-22) = (P8)/NF'' = (1/3)/2 = 1/6

TransProb(CR-23) = (P11)/NF'' = (1/3)/2 = 1/6

Notice that CR-22 and CR-23 have the same transition states, therefore the transition probability in the transition matrix is the sum of both transitions, i.e.

TransMatrixProb(CR-22 and CR-23) = TransProb(CR-22) + TransProb(Cr-23) = 1/6 + 1/6 = 2/6 = 1/3

CR-14

There is only one transition that starts with the source state <G.closed, C.monitor, T.leave>.

The transition probability of CR-14 is computed as the following:

The following probabilities are associated to CR-14:

P1 = Prob(<G.closed, G.closed>:Exit) : 0

P2 = Prob(<C.monitor, C.deactivate>:Exit) : 1/3

P3 = Prob(<T.leave, T.idle>:Exit) : 1

NF Computation:

Event *Exit* is an external event and triggers transition CR-14; therefore the NF is computed as the following

P2 * P3 = 1/3 * 1 = 1/3

NF' Computation:

There is no internal event involved in the transition, therefore NF' is equal to 0.

NF'' Computation:

NF'' = NF' + NF = 0 + (P2*P3) = 0 + (1/3*1) = 1/3

Transition Probability:

TransProb(CR-14) = (P2*P3)/NF'' = (1/3*1)/(1/3) = 1

CR-15

There is only one transition that starts with the source state <G.toClose, C.deactivate, T.idle>.

The transition probability of CR-15 is computed as the following:

The following probabilities are associated to CR-15:

P1 = Prob(<G.toClose, G.closed>:Down) : 1

P2 = Prob(<C.deactivate, C.deactivate>:Down) : 0

P3 = Prob(<T.idle, T.idle>:Down) : 0

NF Computation:

There is no external event involved in the transition, therefore NF is equal to 0.

NF' Computation:

Event *Down* is an internal event and triggers transition CR-15; therefore the NF' is computed as the following:

P1 + P2 + P3 = 1 + 0 + 0 = 1

NF'' Computation:

NF'' = NF' + NF = (P1+P2+P3) + 0 = (1 + 0 + 0) + 0 = 1

Transition Probability:

TransProb(CR-15) = (P1+P2+P3)/NF'' = (1+0+0)/1 = 1

CR-16

There is only one transition that starts with the source state <G.closed, C.deactivate, T.idle>.

The transition probability of CR-16 is computed as the following:

The following probabilities are associated to CR-16:

P1 = Prob(<G.closed, G.toOpen>:Raise) : 1

P2 = Prob(<C.deactivate, C.idle>:Raise) : 1

P3 = Prob(<T.idle, T.idle>:Exit) : 0

NF Computation:

Event *Raise* is an external event and triggers transition CR-16; therefore the NF is computed as the following:

P1 * P2 = 1 * 1 = 1

NF' Computation:

There is no internal event involved in the transition, therefore NF' is equal to 0.

NF'' Computation:

NF'' = NF' + NF = 0 + (P1*P2) = 0 + (1*1) = 1

Transition Probability:

TransProb(CR-16) = (P1*P2)/NF'' = (1*1)/(1) = 1

CR-17

There is only one transition that starts with the source state <G.toOpen, C.idle, T.idle>.

The transition probability of CR-17 is computed as the following:

The following probabilities are associated to CR-17:

P1 = Prob(<G.toOpen, G.opened>:Up) : 1

P2 = Prob(<C.idle, C.idle>:Up) : 0

P3 = Prob(<T.idle, T.idle>:Up) : 0

NF Computation:

There is no external event involved in the transition, therefore NF is equal to 0.

NF' Computation:

Event *Up* is an internal event and triggers transition CR-17; therefore the NF' is computed as the following:

P1 + P2 + P3 = 1 + 0 + 0 = 1

NF'' Computation:

NF'' = NF' + NF = (P1+P2+P3) + 0 = (1 + 0 + 0) + 0 = 1

Transition Probability:

TransProb(CR-17) = (P1+P2+P3)/NF'' = (1+0+0)/1 = 1

Table7 shows the transition matrix of the synchronous product machine of Gate-Train-Controller system.

State List:

S1 - <G.opened, C.idle, T.idle>

S2 - <G.opened, C.activate, T.toCross>

S3 - <G.toClose, C.monitor, T.toCross>

S4 - <G.opened, C.activate, T.cross>

S5 - <G.closed, C.monitor, T.toCross>

S6 - <G.toClose, C.monitor, T.cross>

S7 - <G.opened, C.activate, T.leave>

S8 - <G.closed, C.monitor, T.cross>

S9 - <G.toClose, C.monitor, T.leave>

S10 - <G.closed, C.monitor, T.leave>

S11 - <G.toClose, C.deactivate, T.idle>

S12 - <G.closed, C.deactivate, T.idle>

S13 - < G.toOpen, C.idle, T.idle>

Table 7. Transition Matrix of Synchronous Product Machine Gate-Train-Controller

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
S1	0	1	0	0	0	0	0	0	0	0	0	0	0
S2	0	1/4	1/4	1/2	0	0	0	0	0	0	0	0	0
S3	0	0	2/8	0	3/8	3/8	0	0	0	0	0	0	0
S4	0	0	0	0	0	1/3	2/3	0	0	0	0	0	0
S5	0	0	0	0	0	0	0	1	0	0	0	0	0
S6	0	0	0	0	0	0	0	1/2	½	0	0	0	0
S7	0	0	0	0	0	0	0	0	1	0	0	0	0
S8	0	0	0	0	0	0	0	0	0	1	0	0	0
S9	0	0	0	0	0	0	0	0	1/3	1/2	1/6	0	0
S10	0	0	0	0	0	0	0	0	0	0	0	1	0
S11	0	0	0	0	0	0	0	0	0	0	0	1	0
S12	0	0	0	0	0	0	0	0	0	0	0	0	1

S13	1	0	0	0	0	0	0	0	0	0	0	0	0

6.3.2 Steady Vector

Need to find a vector such that

$|MNPQRSTUVWXYZ|$

$$\begin{bmatrix}
0/1 & 1/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 1/4 & 1/4 & 1/2 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 2/8 & 0/1 & 3/8 & 3/8 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/3 & 2/3 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/2 & 1/2 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 & 0/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 & 0/1 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/3 & 1/2 & 1/6 & 0/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 & 0/1 \\
0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 1/1 \\
1/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1 & 0/1
\end{bmatrix}$$

$= |MNPQRSTUVWXYZ|$

Converting the above formula to a set of equations gives:

0M + 0N + 0P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 1Z = M

1/1M + 1/4N + 0P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = N

0M + 1/4N + 2/8P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = P

0M + 1/2N + 0P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = Q

0M + 0N + 3/8P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = R

0M + 0N + 3/8P + 1/3Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = S

0M + 0N + 0P + 2/3Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = T

$0M + 0N + 0P + 0Q + 1/1R + 1/2S + 0T + 0U + 0V + 0W + 0X + 0Y + 0Z = U$

$0M + 0N + 0P + 0Q + 0R + 1/2S + 1/1T + 0U + 1/3V + 0W + 0X + 0Y + 0Z = V$

$0M + 0N + 0P + 0Q + 0R + 0S + 0T + 1/1U + 1/2V + 0W + 0X + 0Y + 0Z = W$

$0M + 0N + 0P + 0Q + 0R + 0S + 0T + 0U + 1/6V + 0W + 0X + 0Y + 0Z = X$

$0M + 0N + 0P + 0Q + 0R + 0S + 0T + 0U + 0V + 1W + 1X + 0Y + 0Z = Y$

$0M + 0N + 0P + 0Q + 0R + 0S + 0T + 0U + 0V + 0W + 0X + 1Y + 0Z = Z$

By solving the equations we get:

$M = 0.114$

$N = 0.152$

$P = 0.051$

$Q = 0.076$

$R = 0.019$

$S = 0.044$

$T = 0.051$

$U = 0.041$

$V = 0.109$

$W = 0.096$

$X = 0.018$

$Y = 0.114$

$Z = 0.114$

[0.114 0.152 0.051 0.076 0.019 0.044 0.051 0.041 0.109 0.096 0.018 0.114

0.114] ... Steady Vector of Synchronous Product Machine Gate-Train-Controller

6.3.2.1 Entropy

$\mathbf{H}_{Gate-Train-Controller}$ =

$-((0.114 * (0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.152 * (0 * \log_2 0 + 1/4 * \log_2 1/4 + 1/4 * \log_2 1/4 + 1/2 * \log 1/2 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.051 * (0 * \log_2 0 + 0 * \log_2 0 + 2/8 * \log_2 2/8 + 0 * \log_2 0 + 3/8 \log_2 3/8 + 3/8 \log_2 3/8 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.076 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1/3 * \log_2 1/3 + 2/3 * \log_2 2/3 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.019 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.044 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1/2 * \log_2 1/2 + 1/2 * \log_2 1/2 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.051 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.041 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.109 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1/3 * \log_2 1/3 + 1/2 * \log_2 1/2 + 1/6 * \log_2 1/6 + 0 * \log_2 0 + 0 * \log_2 0)) +$

$(0.096 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0)) +$

$(0.018 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1 + 0 * \log_2 0)) +$

$(0.114 * (0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 1 * \log_2 1)) +$

$(0.114 * (1 * \log_2 1 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0 + 0 * \log_2 0)))$

$= -(-0.58045) = 0.58045$

6.4 Reliability Computation

Reliability(Gate-Train-Controller)

$= (H_{Gate} + H_{Train} + H_{Controller}) - H_{Gate-Train-Controller}$

$= (0 + 0 + 0.6785) - 0.58045 = 0.09805$

6.5 Conclusion of the results

The results generated from the TROM-SRMS are provided in appendix 1 and they conform to those computed in the previous sections.

Please note that due to rounding at different precision, there are slight differences between the results generated by the TROM-SRMS and the theoretically computed results in previous sections.

Chapter 7. Conclusion & Future Work

This project is about the implementation of TROM-SRMS, a software module in the TROMLAB environment for assessment of the reliability of a real-time reactive system based on the Markov model. Background theories on the reliability computation as well as the key algorithms are presented in this report. A case study is provided as a means to validate the results generated from the TROM-SRMS.

The future work should consider the following enhancements:

- Currently the user has to specify the system configuration file of the system, it would be more convenient and transparent to the user if there were an alternative way to incorporate the file automatically to the TROM-SRMS.
- Currently, in order to generate the synchronous product machine specification, the user needs to remove all the time-constraints from the GRC specifications and run the *system test* with TROM-SBTS. The suggestion is to find a way to generate the specification in an automatic fashion.

References

[AAM98] V.S.Alagar, R.Achuthan, D.Muthiayen. TROMLIB: *A Software Development Environment for Real-Time Reactive Systems.* (first version 1996, revised 2001), Submitted for Publication

[Ach95] R. Achuthan, *A Formal Model for Object-Oriented Development of Real-Time Reactive Systems*. PhD. thesis, Department of Computer Science, Concordia University, Montreal, Canada, October 1995

[Che02] M. Chen. *The Implementation of Specification-based Testing System for Real-time Reactive System in TROMLIB Framework*. Master Major Report, Department of Computer Science, Concordia University, Montreal, Canada, December 2002

[GH93] J.V. Guttag and J.J. Horning. *Larch: Language and Tools for Formal Specifications*. Springer Verlag. 1993.

[Hai99] G. Haidar. *Reasoning System for Real-Time Reactive Systems*. Master Thesis, Department of Computer Science, Concordia University, Montreal, Canada, December 1999

[HP85] D. Harel, A. Pnueli. *On the Development of Reactive Systems.* In Logic and Models of Concurrent Systems, NATO, Advanced Study Institute on Logics and Models for Verification and Specification of Concurrent Systems Spring Verlag, 1985

[Mut96] D.Muthiayen *Animation and Formal Verification of Real-Time Reactive Systems in an Object-Oriented Environment.* Master Thesis, Department of Computer Science, Concordia University, Montreal, Canada, October 1996

[Nag99] D.Muthiayen *Real-Time Reactive System Developemnt – A Formal approach based on UML and PVS.* PhD Thesis, Department of Computer Science, Concordia University, Montreal, Canada, January 2000

[Orm02] O. Ormandjieva, *Deriving New Measurements for Real-Time Reactive Systems.* PhD. Thesis, Department of Computer Science, Concordia University, Montreal, Canada, 2002

[Pom99] F. Pompeo *A Formal Verification Assistant for TROMLIB environment.* Master Thesis, Department of Computer Science, Concordia University, Montreial, Canada, November 1999

[Pop99] O. Popistas. Rose-GRC Translator: *Mapping UML Visual Models onto Formal Specifications.* Master Thesis, Department of Computer Science, Concordia University, Montreial, Canada, April 1999

[PR01] Pressman, Roger S. *Software Engineering: a Practitioner's approach* –5th ed.p. cm-(McGraw-Hill series in computer Science) ISBN: 0-07-3655778-3

[Sir99] V. Srinivasan. *Graphical User Interface for TROMLIB Environment.* Master Thesis, Department of Computer Science, Concordia University, Montreal, Canada, December 1999.

[SOM95] I. Sommerville, *Software Engineering* 5th edition, Addison-Wesley ISBN 0201-42765-6

[Tao96] H. Tao. *Static Analyzer: A Design Tool for TROM.* Master Thesis, Department of Computer Science, Concordia University, Montreal, Canada, August 1996

[Zhe02] M.Zheng. *Automated Generation of Test Suits from Formal Specifications of Real-Time Reactive Systems*. Ph.D. Thesis, Department of Computer Science, Concordia University, Montreal, Canada, 2002.

Appendix Results of running Gate-Train-Controller on TROM-SRMS

---------Grc controller_spec.txt---------
=>State Transition Matrix<=
Probability for <idle,idle>: 0.0/1.0
Probability for <idle,activate>: 1.0/1.0
Probability for <idle,deactivate>: 0.0/1.0
Probability for <idle,monitor>: 0.0/1.0
Probability for <activate,idle>: 0.0/2.0
Probability for <activate,activate>: 1.0/2.0
Probability for <activate,deactivate>: 0.0/2.0
Probability for <activate,monitor>: 1.0/2.0
Probability for <deactivate,idle>: 1.0/1.0
Probability for <deactivate,activate>: 0.0/1.0
Probability for <deactivate,deactivate>: 0.0/1.0
Probability for <deactivate,monitor>: 0.0/1.0
Probability for <monitor,idle>: 0.0/3.0
Probability for <monitor,activate>: 0.0/3.0
Probability for <monitor,deactivate>: 1.0/3.0
Probability for <monitor,monitor>: 2.0/3.0

	idle	activate	deactivate	monitor
idle	0.0/1.0	1.0/1.0	0.0/1.0	0.0/1.0
activate	0.0/2.0	1.0/2.0	0.0/2.0	1.0/2.0
deactivate	1.0/1.0	0.0/1.0	0.0/1.0	0.0/1.0
monitor	0.0/3.0	0.0/3.0	1.0/3.0	2.0/3.0

-----steady vector---
0.14285714285714285
0.2857142857142857
0.14285714285714288
0.4285714285714286

.....the entropy value is 0.6792696431662097
====>State Transition List<====
*****State Transition Info*****
From State :
activate
To State/Event
monitor/Lower prob: 1.0/2.0
activate/Near prob: 1.0/2.0

*****State Transition Info*****
From State :
deactivate
To State/Event
idle/Raise prob: 1.0/1.0

*****State Transition Info*****
From State :
monitor
To State/Event
deactivate/Exit prob: 1.0/3.0
monitor/Exit prob: 1.0/3.0
monitor/Near prob: 1.0/3.0

*****State Transition Info*****
From State :
idle
To State/Event
activate/Near prob: 1.0/1.0

State List
idle
activate
deactivate
monitor
Internal Event List
Ouput Event List
Lower
Raise
Input Event List
Near
Exit
---------Grc gate-spec.txt---------
=>State Transition Matrix<=
Probability for <opened,opened>: 0.0/1.0
Probability for <opened,toClose>: 1.0/1.0
Probability for <opened,toOpen>: 0.0/1.0
Probability for <opened,closed>: 0.0/1.0
Probability for <toClose,opened>: 0.0/1.0
Probability for <toClose,toClose>: 0.0/1.0
Probability for <toClose,toOpen>: 0.0/1.0
Probability for <toClose,closed>: 1.0/1.0
Probability for <toOpen,opened>: 1.0/1.0
Probability for <toOpen,toClose>: 0.0/1.0
Probability for <toOpen,toOpen>: 0.0/1.0
Probability for <toOpen,closed>: 0.0/1.0
Probability for <closed,opened>: 0.0/1.0
Probability for <closed,toClose>: 0.0/1.0
Probability for <closed,toOpen>: 1.0/1.0
Probability for <closed,closed>: 0.0/1.0

	opened	toClose	toOpen	closed
opened	0.0/1.0	1.0/1.0	0.0/1.0	0.0/1.0

```
toClose    0.0/1.0    0.0/1.0    0.0/1.0    1.0/1.0
 toOpen    1.0/1.0    0.0/1.0    0.0/1.0    0.0/1.0
 closed    0.0/1.0    0.0/1.0    1.0/1.0    0.0/1.0
-----steady vector---
0.25000000000000006
0.2500000000000001
0.24999999999999986
0.25000000000000006

.....the entropy value is -0.0
====>State Transition List<====
*****State Transition Info*****
From State :
opened
To State/Event
toClose/Lower prob: 1.0/1.0

*****State Transition Info*****
From State :
toClose
To State/Event
closed/Down prob: 1.0/1.0

*****State Transition Info*****
From State :
toOpen
To State/Event
opened/Up prob: 1.0/1.0

*****State Transition Info*****
From State :
closed
To State/Event
toOpen/Raise prob: 1.0/1.0

**State List**
opened
toClose
toOpen
closed
**Internal Event List**
Down
Up
**Ouput Event List**
**Input Event List**
Lower
```

```
Raise
---------Grc train_spec.txt---------
=>State Transition Matrix<=
Probability for <idle,idle>: 0.0/1.0
Probability for <idle,cross>: 0.0/1.0
Probability for <idle,leave>: 0.0/1.0
Probability for <idle,toCross>: 1.0/1.0
Probability for <cross,idle>: 0.0/1.0
Probability for <cross,cross>: 0.0/1.0
Probability for <cross,leave>: 1.0/1.0
Probability for <cross,toCross>: 0.0/1.0
Probability for <leave,idle>: 1.0/1.0
Probability for <leave,cross>: 0.0/1.0
Probability for <leave,leave>: 0.0/1.0
Probability for <leave,toCross>: 0.0/1.0
Probability for <toCross,idle>: 0.0/1.0
Probability for <toCross,cross>: 1.0/1.0
Probability for <toCross,leave>: 0.0/1.0
Probability for <toCross,toCross>: 0.0/1.0
            idle      cross      leave    toCross
  idle    0.0/1.0    0.0/1.0    0.0/1.0    1.0/1.0
 cross    0.0/1.0    0.0/1.0    1.0/1.0    0.0/1.0
 leave    1.0/1.0    0.0/1.0    0.0/1.0    0.0/1.0
toCross    0.0/1.0    1.0/1.0    0.0/1.0    0.0/1.0
-----steady vector---
0.25000000000000006
0.24999999999999994
0.24999999999999997
0.25000000000000006

.............the entropy value is -0.0
====>State Transition List<====
*****State Transition Info*****
From State :
idle
To State/Event
toCross/Near prob: 1.0/1.0

*****State Transition Info*****
From State :
cross
To State/Event
leave/Out prob: 1.0/1.0

*****State Transition Info*****
From State :
```

leave
To State/Event
idle/Exit prob: 1.0/1.0

*****State Transition Info*****
From State :
toCross
To State/Event
cross/In prob: 1.0/1.0

State List
idle
cross
leave
toCross
Internal Event List
Out
In
Ouput Event List
Near
Exit
Input Event List
----------Synchrnous product machine <<C_T_G-grid.txt>>----------
----------State List----------
G.opened,C.idle,T.idle
G.opened,C.activate,T.toCross
G.toClose,C.monitor,T.toCross
G.opened,C.activate,T.cross
G.closed,C.monitor,T.toCross
G.toClose,C.monitor,T.cross
G.opened,C.activate,T.leave
G.closed,C.monitor,T.cross
G.toClose,C.monitor,T.leave
G.closed,C.monitor,T.leave
G.toClose,C.deactivate,T.idle
G.closed,C.deactivate,T.idle
G.toOpen,C.idle,T.idle
----------State Transition List----------
*****State Transition Info*****
From State :
G.opened,C.idle,T.idle
To State/Event
G.opened,C.activate,T.toCross/Near prob: 1.0/1.0
NF External: 1.0/1.0
NF Internal: 0.0/1.0
NF : 1.0/1.0

*****State Transition Info*****
From State :
G.opened,C.activate,T.toCross
To State/Event
G.toClose,C.monitor,T.toCross/Lower prob: 1.0/2.0
G.opened,C.activate,T.cross/In prob: 1.0/1.0
G.opened,C.activate,T.toCross/Near prob: 1.0/2.0
NF External: 4.0/4.0
NF Internal: 1.0/1.0
NF : 8.0/4.0
*****State Transition Info*****
From State :
G.toClose,C.monitor,T.toCross
To State/Event
G.closed,C.monitor,T.toCross/Down prob: 1.0/1.0
G.toClose,C.monitor,T.cross/In prob: 1.0/1.0
G.toClose,C.monitor,T.toCross/Exit prob: 1.0/3.0
G.toClose,C.monitor,T.toCross/Near prob: 1.0/3.0
NF External: 6.0/9.0
NF Internal: 2.0/1.0
NF : 24.0/9.0
*****State Transition Info*****
From State :
G.opened,C.activate,T.cross
To State/Event
G.toClose,C.monitor,T.cross/Lower prob: 1.0/2.0
G.opened,C.activate,T.leave/Out prob: 1.0/1.0
NF External: 1.0/2.0
NF Internal: 1.0/1.0
NF : 3.0/2.0
*****State Transition Info*****
From State :
G.closed,C.monitor,T.toCross
To State/Event
G.closed,C.monitor,T.cross/In prob: 1.0/1.0
NF External: 0.0/1.0
NF Internal: 1.0/1.0
NF : 1.0/1.0
*****State Transition Info*****
From State :
G.toClose,C.monitor,T.cross
To State/Event
G.closed,C.monitor,T.cross/Down prob: 1.0/1.0
G.toClose,C.monitor,T.leave/Out prob: 1.0/1.0
NF External: 0.0/1.0
NF Internal: 2.0/1.0

NF : 2.0/1.0
*****State Transition Info*****
From State :
G.opened,C.activate,T.leave
To State/Event
G.toClose,C.monitor,T.leave/Lower prob: 1.0/2.0
NF External: 1.0/2.0
NF Internal: 0.0/1.0
NF : 1.0/2.0
*****State Transition Info*****
From State :
G.closed,C.monitor,T.cross
To State/Event
G.closed,C.monitor,T.leave/Out prob: 1.0/1.0
NF External: 0.0/1.0
NF Internal: 1.0/1.0
NF : 1.0/1.0
*****State Transition Info*****
From State :
G.toClose,C.monitor,T.leave
To State/Event
G.toClose,C.deactivate,T.idle/Exit prob: 1.0/3.0
G.closed,C.monitor,T.leave/Down prob: 1.0/1.0
G.toClose,C.monitor,T.leave/Exit prob: 1.0/3.0
G.toClose,C.monitor,T.leave/Near prob: 1.0/3.0
NF External: 27.0/27.0
NF Internal: 1.0/1.0
NF : 54.0/27.0
*****State Transition Info*****
From State :
G.closed,C.monitor,T.leave
To State/Event
G.closed,C.deactivate,T.idle/Exit prob: 1.0/3.0
NF External: 1.0/3.0
NF Internal: 0.0/1.0
NF : 1.0/3.0
*****State Transition Info*****
From State :
G.toClose,C.deactivate,T.idle
To State/Event
G.closed,C.deactivate,T.idle/Down prob: 1.0/1.0
NF External: 0.0/1.0
NF Internal: 1.0/1.0
NF : 1.0/1.0
*****State Transition Info*****
From State :

G.closed,C.deactivate,T.idle
To State/Event
G.toOpen,C.idle,T.idle/Raise prob: 1.0/1.0
NF External: 1.0/1.0
NF Internal: 0.0/1.0
NF : 1.0/1.0
*****State Transition Info*****
From State :
G.toOpen,C.idle,T.idle
To State/Event
G.opened,C.idle,T.idle/Up prob: 1.0/1.0
NF External: 0.0/1.0
NF Internal: 1.0/1.0
NF : 1.0/1.0
----------Probability----------
1. <G.opened,C.idle,T.idle> to <G.opened,C.idle,T.idle> with probability : 0.0
2. <G.opened,C.idle,T.idle> to <G.opened,C.activate,T.toCross> with probability : 1.0
3. <G.opened,C.idle,T.idle> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
4. <G.opened,C.idle,T.idle> to <G.opened,C.activate,T.cross> with probability : 0.0
5. <G.opened,C.idle,T.idle> to <G.closed,C.monitor,T.toCross> with probability : 0.0
6. <G.opened,C.idle,T.idle> to <G.toClose,C.monitor,T.cross> with probability : 0.0
7. <G.opened,C.idle,T.idle> to <G.opened,C.activate,T.leave> with probability : 0.0
8. <G.opened,C.idle,T.idle> to <G.closed,C.monitor,T.cross> with probability : 0.0
9. <G.opened,C.idle,T.idle> to <G.toClose,C.monitor,T.leave> with probability : 0.0
10. <G.opened,C.idle,T.idle> to <G.closed,C.monitor,T.leave> with probability : 0.0
11. <G.opened,C.idle,T.idle> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
12. <G.opened,C.idle,T.idle> to <G.closed,C.deactivate,T.idle> with probability : 0.0
13. <G.opened,C.idle,T.idle> to <G.toOpen,C.idle,T.idle> with probability : 0.0
14. <G.opened,C.activate,T.toCross> to <G.opened,C.idle,T.idle> with probability : 0.0
15. <G.opened,C.activate,T.toCross> to <G.opened,C.activate,T.toCross> with probability : 0.25
16. <G.opened,C.activate,T.toCross> to <G.toClose,C.monitor,T.toCross> with probability : 0.25
17. <G.opened,C.activate,T.toCross> to <G.opened,C.activate,T.cross> with probability : 0.5
18. <G.opened,C.activate,T.toCross> to <G.closed,C.monitor,T.toCross> with probability : 0.0
19. <G.opened,C.activate,T.toCross> to <G.toClose,C.monitor,T.cross> with probability : 0.0
20. <G.opened,C.activate,T.toCross> to <G.opened,C.activate,T.leave> with probability : 0.0
21. <G.opened,C.activate,T.toCross> to <G.closed,C.monitor,T.cross> with probability : 0.0
22. <G.opened,C.activate,T.toCross> to <G.toClose,C.monitor,T.leave> with probability : 0.0

23. <G.opened,C.activate,T.toCross> to <G.closed,C.monitor,T.leave> with probability : 0.0
24. <G.opened,C.activate,T.toCross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
25. <G.opened,C.activate,T.toCross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
26. <G.opened,C.activate,T.toCross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
27. <G.toClose,C.monitor,T.toCross> to <G.opened,C.idle,T.idle> with probability : 0.0
28. <G.toClose,C.monitor,T.toCross> to <G.opened,C.activate,T.toCross> with probability : 0.0
29. <G.toClose,C.monitor,T.toCross> to <G.toClose,C.monitor,T.toCross> with probability : 0.25
30. <G.toClose,C.monitor,T.toCross> to <G.opened,C.activate,T.cross> with probability : 0.0
31. <G.toClose,C.monitor,T.toCross> to <G.closed,C.monitor,T.toCross> with probability : 0.375
32. <G.toClose,C.monitor,T.toCross> to <G.toClose,C.monitor,T.cross> with probability : 0.375
33. <G.toClose,C.monitor,T.toCross> to <G.opened,C.activate,T.leave> with probability : 0.0
34. <G.toClose,C.monitor,T.toCross> to <G.closed,C.monitor,T.cross> with probability : 0.0
35. <G.toClose,C.monitor,T.toCross> to <G.toClose,C.monitor,T.leave> with probability : 0.0
36. <G.toClose,C.monitor,T.toCross> to <G.closed,C.monitor,T.leave> with probability : 0.0
37. <G.toClose,C.monitor,T.toCross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
38. <G.toClose,C.monitor,T.toCross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
39. <G.toClose,C.monitor,T.toCross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
40. <G.opened,C.activate,T.cross> to <G.opened,C.idle,T.idle> with probability : 0.0
41. <G.opened,C.activate,T.cross> to <G.opened,C.activate,T.toCross> with probability : 0.0
42. <G.opened,C.activate,T.cross> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
43. <G.opened,C.activate,T.cross> to <G.opened,C.activate,T.cross> with probability : 0.0
44. <G.opened,C.activate,T.cross> to <G.closed,C.monitor,T.toCross> with probability : 0.0
45. <G.opened,C.activate,T.cross> to <G.toClose,C.monitor,T.cross> with probability : 0.3333333333333333
46. <G.opened,C.activate,T.cross> to <G.opened,C.activate,T.leave> with probability : 0.6666666666666666
47. <G.opened,C.activate,T.cross> to <G.closed,C.monitor,T.cross> with probability : 0.0

48. <G.opened,C.activate,T.cross> to <G.toClose,C.monitor,T.leave> with probability : 0.0
49. <G.opened,C.activate,T.cross> to <G.closed,C.monitor,T.leave> with probability : 0.0
50. <G.opened,C.activate,T.cross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
51. <G.opened,C.activate,T.cross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
52. <G.opened,C.activate,T.cross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
53. <G.closed,C.monitor,T.toCross> to <G.opened,C.idle,T.idle> with probability : 0.0
54. <G.closed,C.monitor,T.toCross> to <G.opened,C.activate,T.toCross> with probability : 0.0
55. <G.closed,C.monitor,T.toCross> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
56. <G.closed,C.monitor,T.toCross> to <G.opened,C.activate,T.cross> with probability : 0.0
57. <G.closed,C.monitor,T.toCross> to <G.closed,C.monitor,T.toCross> with probability : 0.0
58. <G.closed,C.monitor,T.toCross> to <G.toClose,C.monitor,T.cross> with probability : 0.0
59. <G.closed,C.monitor,T.toCross> to <G.opened,C.activate,T.leave> with probability : 0.0
60. <G.closed,C.monitor,T.toCross> to <G.closed,C.monitor,T.cross> with probability : 1.0
61. <G.closed,C.monitor,T.toCross> to <G.toClose,C.monitor,T.leave> with probability : 0.0
62. <G.closed,C.monitor,T.toCross> to <G.closed,C.monitor,T.leave> with probability : 0.0
63. <G.closed,C.monitor,T.toCross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
64. <G.closed,C.monitor,T.toCross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
65. <G.closed,C.monitor,T.toCross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
66. <G.toClose,C.monitor,T.cross> to <G.opened,C.idle,T.idle> with probability : 0.0
67. <G.toClose,C.monitor,T.cross> to <G.opened,C.activate,T.toCross> with probability : 0.0
68. <G.toClose,C.monitor,T.cross> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
69. <G.toClose,C.monitor,T.cross> to <G.opened,C.activate,T.cross> with probability : 0.0
70. <G.toClose,C.monitor,T.cross> to <G.closed,C.monitor,T.toCross> with probability : 0.0
71. <G.toClose,C.monitor,T.cross> to <G.toClose,C.monitor,T.cross> with probability : 0.0
72. <G.toClose,C.monitor,T.cross> to <G.opened,C.activate,T.leave> with probability : 0.0

73. <G.toClose,C.monitor,T.cross> to <G.closed,C.monitor,T.cross> with probability : 0.5
74. <G.toClose,C.monitor,T.cross> to <G.toClose,C.monitor,T.leave> with probability : 0.5
75. <G.toClose,C.monitor,T.cross> to <G.closed,C.monitor,T.leave> with probability : 0.0
76. <G.toClose,C.monitor,T.cross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
77. <G.toClose,C.monitor,T.cross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
78. <G.toClose,C.monitor,T.cross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
79. <G.opened,C.activate,T.leave> to <G.opened,C.idle,T.idle> with probability : 0.0
80. <G.opened,C.activate,T.leave> to <G.opened,C.activate,T.toCross> with probability : 0.0
81. <G.opened,C.activate,T.leave> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
82. <G.opened,C.activate,T.leave> to <G.opened,C.activate,T.cross> with probability : 0.0
83. <G.opened,C.activate,T.leave> to <G.closed,C.monitor,T.toCross> with probability : 0.0
84. <G.opened,C.activate,T.leave> to <G.toClose,C.monitor,T.cross> with probability : 0.0
85. <G.opened,C.activate,T.leave> to <G.opened,C.activate,T.leave> with probability : 0.0
86. <G.opened,C.activate,T.leave> to <G.closed,C.monitor,T.cross> with probability : 0.0
87. <G.opened,C.activate,T.leave> to <G.toClose,C.monitor,T.leave> with probability : 1.0
88. <G.opened,C.activate,T.leave> to <G.closed,C.monitor,T.leave> with probability : 0.0
89. <G.opened,C.activate,T.leave> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
90. <G.opened,C.activate,T.leave> to <G.closed,C.deactivate,T.idle> with probability : 0.0
91. <G.opened,C.activate,T.leave> to <G.toOpen,C.idle,T.idle> with probability : 0.0
92. <G.closed,C.monitor,T.cross> to <G.opened,C.idle,T.idle> with probability : 0.0
93. <G.closed,C.monitor,T.cross> to <G.opened,C.activate,T.toCross> with probability : 0.0
94. <G.closed,C.monitor,T.cross> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
95. <G.closed,C.monitor,T.cross> to <G.opened,C.activate,T.cross> with probability : 0.0
96. <G.closed,C.monitor,T.cross> to <G.closed,C.monitor,T.toCross> with probability : 0.0
97. <G.closed,C.monitor,T.cross> to <G.toClose,C.monitor,T.cross> with probability : 0.0

98. <G.closed,C.monitor,T.cross> to <G.opened,C.activate,T.leave> with probability : 0.0
99. <G.closed,C.monitor,T.cross> to <G.closed,C.monitor,T.cross> with probability : 0.0
100. <G.closed,C.monitor,T.cross> to <G.toClose,C.monitor,T.leave> with probability : 0.0
101. <G.closed,C.monitor,T.cross> to <G.closed,C.monitor,T.leave> with probability : 1.0
102. <G.closed,C.monitor,T.cross> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
103. <G.closed,C.monitor,T.cross> to <G.closed,C.deactivate,T.idle> with probability : 0.0
104. <G.closed,C.monitor,T.cross> to <G.toOpen,C.idle,T.idle> with probability : 0.0
105. <G.toClose,C.monitor,T.leave> to <G.opened,C.idle,T.idle> with probability : 0.0
106. <G.toClose,C.monitor,T.leave> to <G.opened,C.activate,T.toCross> with probability : 0.0
107. <G.toClose,C.monitor,T.leave> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
108. <G.toClose,C.monitor,T.leave> to <G.opened,C.activate,T.cross> with probability : 0.0
109. <G.toClose,C.monitor,T.leave> to <G.closed,C.monitor,T.toCross> with probability : 0.0
110. <G.toClose,C.monitor,T.leave> to <G.toClose,C.monitor,T.cross> with probability : 0.0
111. <G.toClose,C.monitor,T.leave> to <G.opened,C.activate,T.leave> with probability : 0.0
112. <G.toClose,C.monitor,T.leave> to <G.closed,C.monitor,T.cross> with probability : 0.0
113. <G.toClose,C.monitor,T.leave> to <G.toClose,C.monitor,T.leave> with probability : 0.3333333333333333
114. <G.toClose,C.monitor,T.leave> to <G.closed,C.monitor,T.leave> with probability : 0.5
115. <G.toClose,C.monitor,T.leave> to <G.toClose,C.deactivate,T.idle> with probability : 0.16666666666666666
116. <G.toClose,C.monitor,T.leave> to <G.closed,C.deactivate,T.idle> with probability : 0.0
117. <G.toClose,C.monitor,T.leave> to <G.toOpen,C.idle,T.idle> with probability : 0.0
118. <G.closed,C.monitor,T.leave> to <G.opened,C.idle,T.idle> with probability : 0.0
119. <G.closed,C.monitor,T.leave> to <G.opened,C.activate,T.toCross> with probability : 0.0
120. <G.closed,C.monitor,T.leave> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
121. <G.closed,C.monitor,T.leave> to <G.opened,C.activate,T.cross> with probability : 0.0
122. <G.closed,C.monitor,T.leave> to <G.closed,C.monitor,T.toCross> with probability : 0.0

123. <G.closed,C.monitor,T.leave> to <G.toClose,C.monitor,T.cross> with probability : 0.0
124. <G.closed,C.monitor,T.leave> to <G.opened,C.activate,T.leave> with probability : 0.0
125. <G.closed,C.monitor,T.leave> to <G.closed,C.monitor,T.cross> with probability : 0.0
126. <G.closed,C.monitor,T.leave> to <G.toClose,C.monitor,T.leave> with probability : 0.0
127. <G.closed,C.monitor,T.leave> to <G.closed,C.monitor,T.leave> with probability : 0.0
128. <G.closed,C.monitor,T.leave> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
129. <G.closed,C.monitor,T.leave> to <G.closed,C.deactivate,T.idle> with probability : 1.0
130. <G.closed,C.monitor,T.leave> to <G.toOpen,C.idle,T.idle> with probability : 0.0
131. <G.toClose,C.deactivate,T.idle> to <G.opened,C.idle,T.idle> with probability : 0.0
132. <G.toClose,C.deactivate,T.idle> to <G.opened,C.activate,T.toCross> with probability : 0.0
133. <G.toClose,C.deactivate,T.idle> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
134. <G.toClose,C.deactivate,T.idle> to <G.opened,C.activate,T.cross> with probability : 0.0
135. <G.toClose,C.deactivate,T.idle> to <G.closed,C.monitor,T.toCross> with probability : 0.0
136. <G.toClose,C.deactivate,T.idle> to <G.toClose,C.monitor,T.cross> with probability : 0.0
137. <G.toClose,C.deactivate,T.idle> to <G.opened,C.activate,T.leave> with probability : 0.0
138. <G.toClose,C.deactivate,T.idle> to <G.closed,C.monitor,T.cross> with probability : 0.0
139. <G.toClose,C.deactivate,T.idle> to <G.toClose,C.monitor,T.leave> with probability : 0.0
140. <G.toClose,C.deactivate,T.idle> to <G.closed,C.monitor,T.leave> with probability : 0.0
141. <G.toClose,C.deactivate,T.idle> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
142. <G.toClose,C.deactivate,T.idle> to <G.closed,C.deactivate,T.idle> with probability : 1.0
143. <G.toClose,C.deactivate,T.idle> to <G.toOpen,C.idle,T.idle> with probability : 0.0
144. <G.closed,C.deactivate,T.idle> to <G.opened,C.idle,T.idle> with probability : 0.0
145. <G.closed,C.deactivate,T.idle> to <G.opened,C.activate,T.toCross> with probability : 0.0
146. <G.closed,C.deactivate,T.idle> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
147. <G.closed,C.deactivate,T.idle> to <G.opened,C.activate,T.cross> with probability : 0.0

148. <G.closed,C.deactivate,T.idle> to <G.closed,C.monitor,T.toCross> with probability : 0.0
149. <G.closed,C.deactivate,T.idle> to <G.toClose,C.monitor,T.cross> with probability : 0.0
150. <G.closed,C.deactivate,T.idle> to <G.opened,C.activate,T.leave> with probability : 0.0
151. <G.closed,C.deactivate,T.idle> to <G.closed,C.monitor,T.cross> with probability : 0.0
152. <G.closed,C.deactivate,T.idle> to <G.toClose,C.monitor,T.leave> with probability : 0.0
153. <G.closed,C.deactivate,T.idle> to <G.closed,C.monitor,T.leave> with probability : 0.0
154. <G.closed,C.deactivate,T.idle> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
155. <G.closed,C.deactivate,T.idle> to <G.closed,C.deactivate,T.idle> with probability : 0.0
156. <G.closed,C.deactivate,T.idle> to <G.toOpen,C.idle,T.idle> with probability : 1.0
157. <G.toOpen,C.idle,T.idle> to <G.opened,C.idle,T.idle> with probability : 1.0
158. <G.toOpen,C.idle,T.idle> to <G.opened,C.activate,T.toCross> with probability : 0.0
159. <G.toOpen,C.idle,T.idle> to <G.toClose,C.monitor,T.toCross> with probability : 0.0
160. <G.toOpen,C.idle,T.idle> to <G.opened,C.activate,T.cross> with probability : 0.0
161. <G.toOpen,C.idle,T.idle> to <G.closed,C.monitor,T.toCross> with probability : 0.0
162. <G.toOpen,C.idle,T.idle> to <G.toClose,C.monitor,T.cross> with probability : 0.0
163. <G.toOpen,C.idle,T.idle> to <G.opened,C.activate,T.leave> with probability : 0.0
164. <G.toOpen,C.idle,T.idle> to <G.closed,C.monitor,T.cross> with probability : 0.0
165. <G.toOpen,C.idle,T.idle> to <G.toClose,C.monitor,T.leave> with probability : 0.0
166. <G.toOpen,C.idle,T.idle> to <G.closed,C.monitor,T.leave> with probability : 0.0
167. <G.toOpen,C.idle,T.idle> to <G.toClose,C.deactivate,T.idle> with probability : 0.0
168. <G.toOpen,C.idle,T.idle> to <G.closed,C.deactivate,T.idle> with probability : 0.0
169. <G.toOpen,C.idle,T.idle> to <G.toOpen,C.idle,T.idle> with probability : 0.0

	G.opened,C.idle,T.idle	G.opened,C.activate,T.toCross
G.toClose,C.monitor,T.toCross		G.opened,C.activate,T.cross
G.closed,C.monitor,T.toCross		G.toClose,C.monitor,T.cross
G.opened,C.activate,T.leave		G.closed,C.monitor,T.cross
G.toClose,C.monitor,T.leave		G.closed,C.monitor,T.leave
G.toClose,C.deactivate,T.idle		G.closed,C.deactivate,T.idle
G.toOpen,C.idle,T.idle		
G.opened,C.idle,T.idle	0.0/1.0	1.0/1.0
0.0/1.0	0.0/1.0	0.0/1.0
0.0/1.0	0.0/1.0	0.0/1.0
0.0/1.0	0.0/1.0	0.0/1.0
G.opened,C.activate,T.toCross	0.0/8.0	4.0/16.0
4.0/16.0	4.0/8.0	0.0/8.0
0.0/8.0	0.0/8.0	0.0/8.0
0.0/8.0	0.0/8.0	0.0/8.0

G.toClose,C.monitor,T.toCross 0.0/24.0 0.0/24.0
54.0/216.0 0.0/24.0 9.0/24.0 9.0/24.0
0.0/24.0 0.0/24.0 0.0/24.0 0.0/24.0
0.0/24.0 0.0/24.0 0.0/24.0
G.opened,C.activate,T.cross 0.0/3.0 0.0/3.0
0.0/3.0 0.0/3.0 0.0/3.0 2.0/6.0
2.0/3.0 0.0/3.0 0.0/3.0 0.0/3.0
0.0/3.0 0.0/3.0 0.0/3.0
G.closed,C.monitor,T.toCross 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 1.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0
G.toClose,C.monitor,T.cross 0.0/2.0 0.0/2.0
0.0/2.0 0.0/2.0 0.0/2.0 0.0/2.0
0.0/2.0 1.0/2.0 1.0/2.0 0.0/2.0
0.0/2.0 0.0/2.0 0.0/2.0
G.opened,C.activate,T.leave 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 2.0/2.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0
G.closed,C.monitor,T.cross 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 1.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0
G.toClose,C.monitor,T.leave 0.0/54.0 0.0/54.0
0.0/54.0 0.0/54.0 0.0/54.0 0.0/54.0
0.0/54.0 0.0/54.0 162.0/486.0 27.0/54.0
27.0/162.0 0.0/54.0 0.0/54.0
G.closed,C.monitor,T.leave 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 3.0/3.0 0.0/1.0
G.toClose,C.deactivate,T.idle 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 1.0/1.0 0.0/1.0
G.closed,C.deactivate,T.idle 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 1.0/1.0
G.toOpen,C.idle,T.idle 1.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0 0.0/1.0
0.0/1.0 0.0/1.0 0.0/1.0
-----steady vector---
0.11410459587955626

0.1521394611727417
0.050713153724247256
0.07606973058637087
0.019017432646592704
0.04437400950871637
0.050713153724247256
0.04120443740095084
0.10935023771790811
0.09587955625990487
0.01822503961965134
0.1141045958795562
0.11410459587955625

.....the entropy value is 0.5811732270874608
.....The Reliability of the Sub-System is 0.09809641607874897