**CHAPTER 1**

**PROBLEM DESCRIPTION**

**1.1 Problem Statement**

In the field of computers, it is a demanding task to comply with safety requirements. Verification of digital hardware designs is becoming an increasingly complex task as the designs are incorporating more functionality, becoming complex and growing larger in size. This gives rise to model checking which is a method for checking whether a finite state model of a system meets the given specification. A model checking tool accepts system requirements, design or a property that a final system is expected to satisfy. The idea is that by ensuring that the model satisfies enough system properties, we increase our support in the correctness of the model. The system requirements are called models because they represent requirements or design. Most real-time embedded or safety-critical systems are control-oriented rather than data-oriented indicating the dynamic behavior is much more important than business logic. Syntax checking plays a vital role in identifying and safeguarding the business logic and more importantly control logic.

* 1. **Problem Solution**

In this application we have implemented a model checking analysis tool for verification of properties defined in CTL temporal logic. Computational Tree Logic(CTL) is a branching-time logic implying that the model of time is tree structure in which the future is not determined. In the context of our application we analyze and check for the correctness of Kripke Structure’s property which is defined by the CTL formula. Kripke is widely used in model checking, it is a variation of transition system which is basically a graph whose nodes represent the reachable states of the system and whose edges represent transitions. We focus on the Boolean Satisfiability(SAT) based formal verification, In formal verification, the conformance of a design to a given set of specifications is proven mathematically, thereby leaving less room for unexplored search spaces. In this Java standalone application which takes the definition of Kriple structure as its input, accepts the state ID for whose property is defined by the CTL formula. The application thoroughly performs syntax checking and provides messages if a Kripke definition can be parsed. The definition language for the CTL follows CTL syntax. For instance, operators and, or, not,-> EX, AX, EF, AF, EG, AG, E[ p U q ], A[ p U q ]]

**CHAPTER 2**

**PROBLEM VISUALIZATION**

**2.1 UML Diagram for CTL Model**

State

Transition

CTL Formula

ExpressionsHolder

KripkeStructure

View

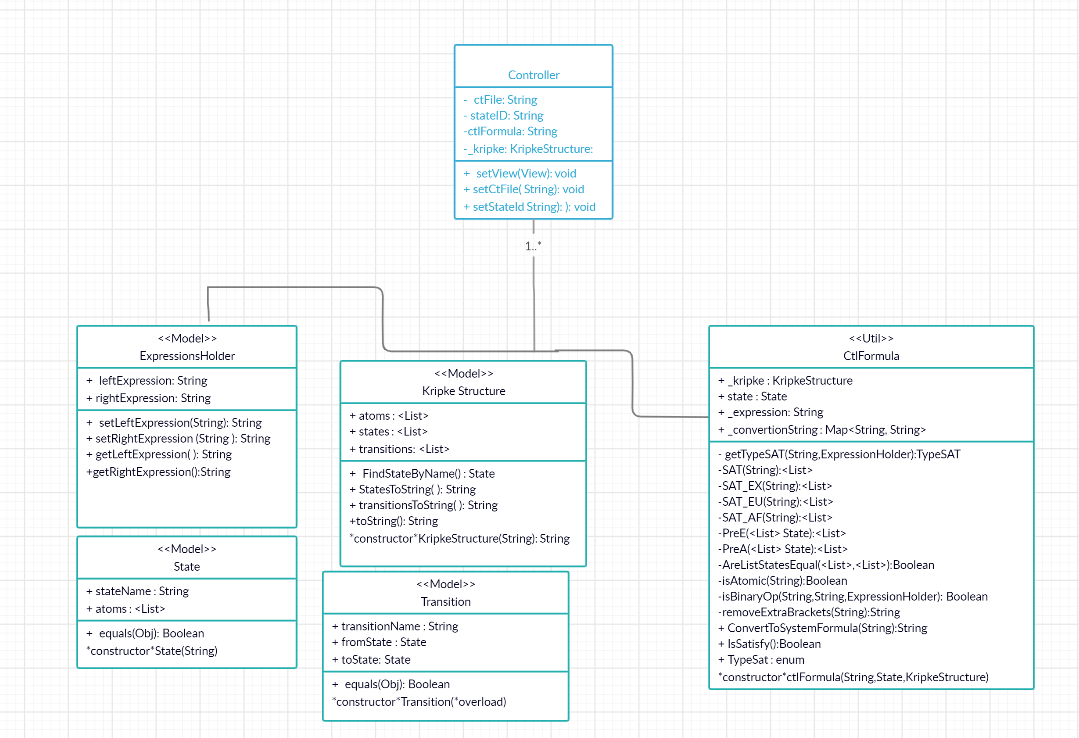
Controller

UTIL

MODEL

VIEW

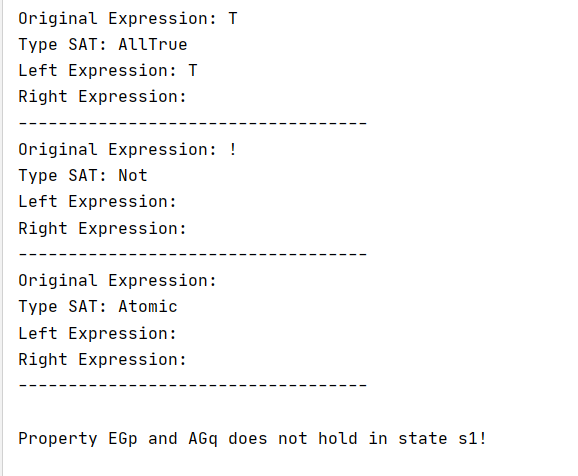
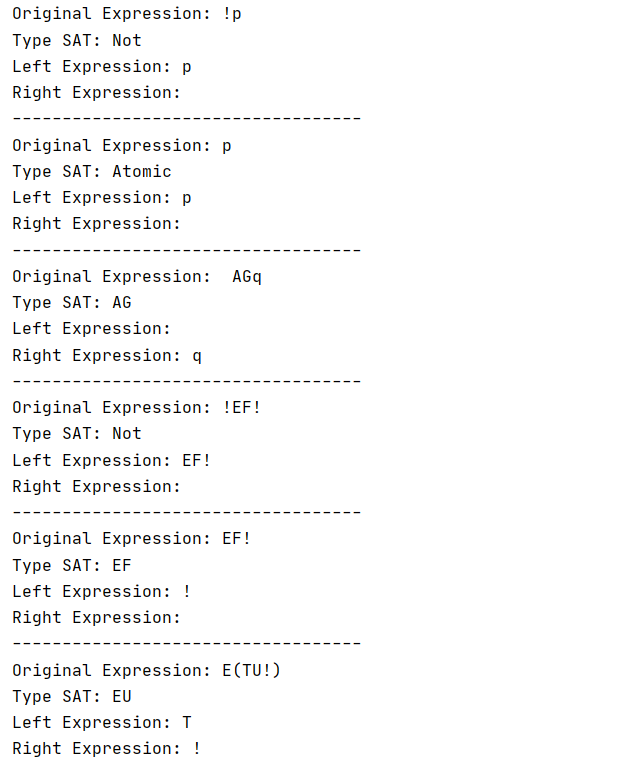
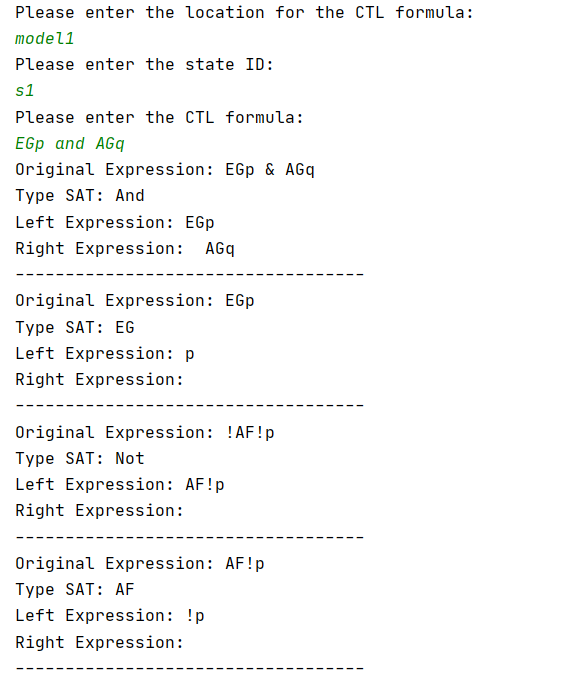
CONTROLLER

**2.2 Class Diagram for CTL Model**

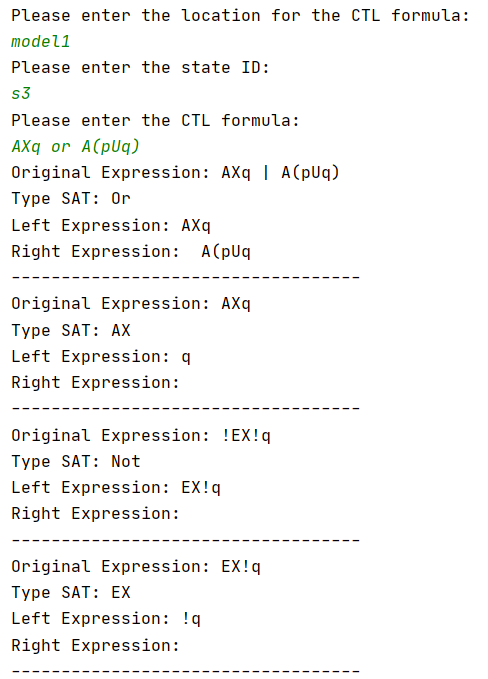
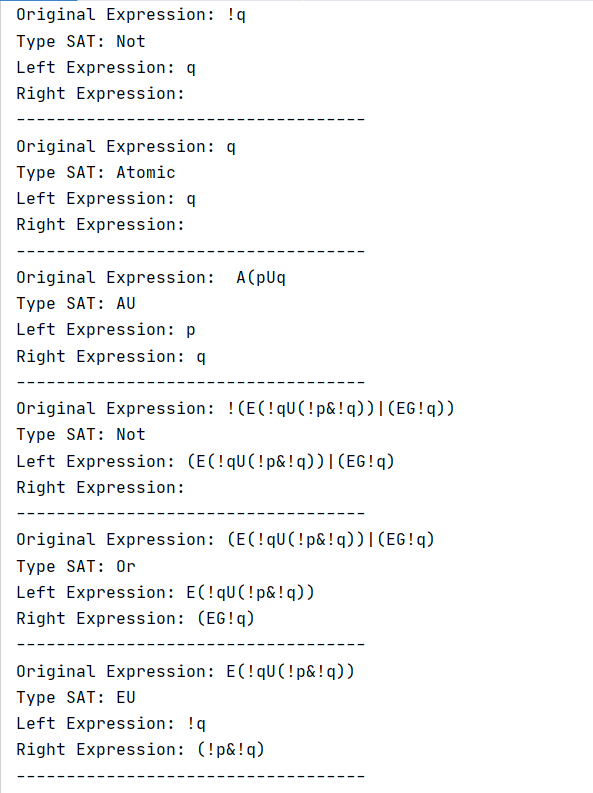
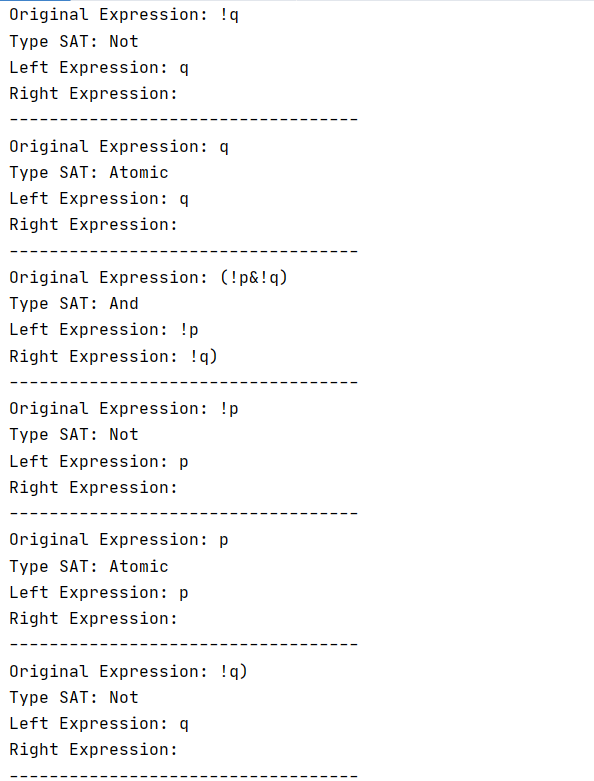
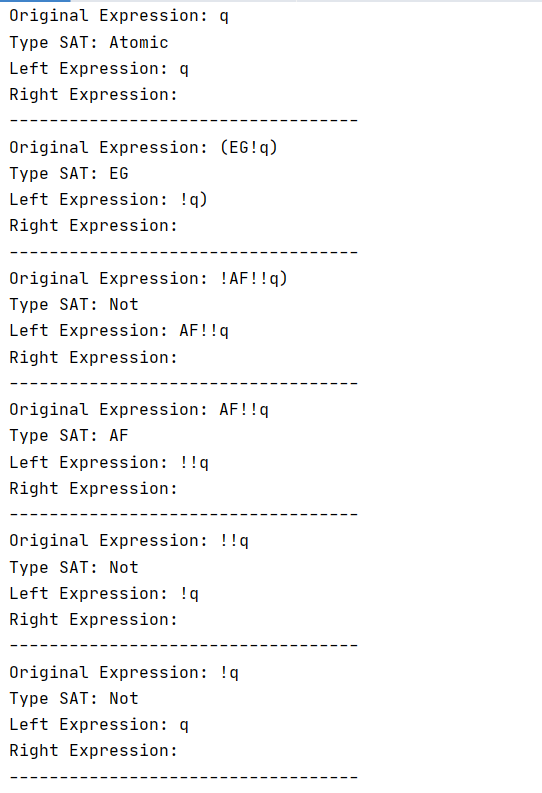
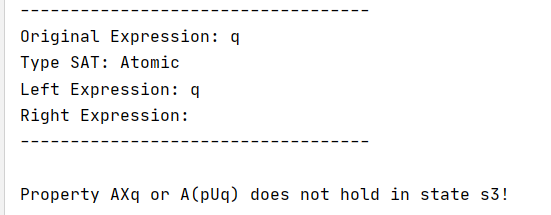
**CHAPTER 3**

**PROBLEM OUTPUT**

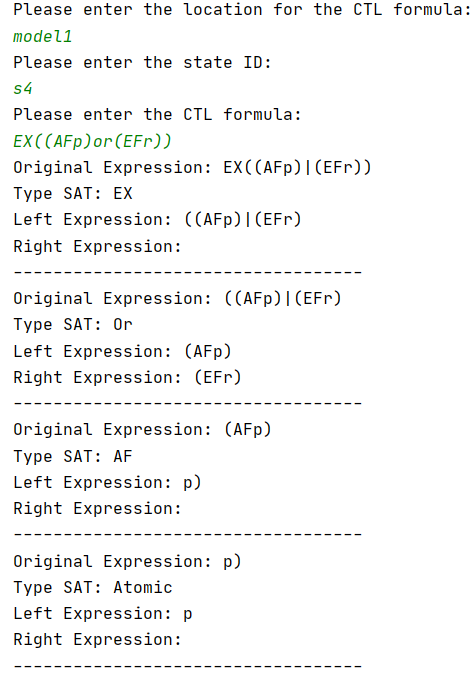
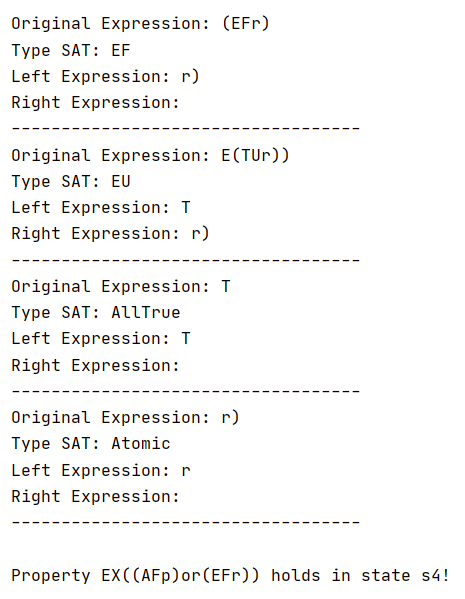
**Output 1: EGp and AGp in State 1 and Model 1**



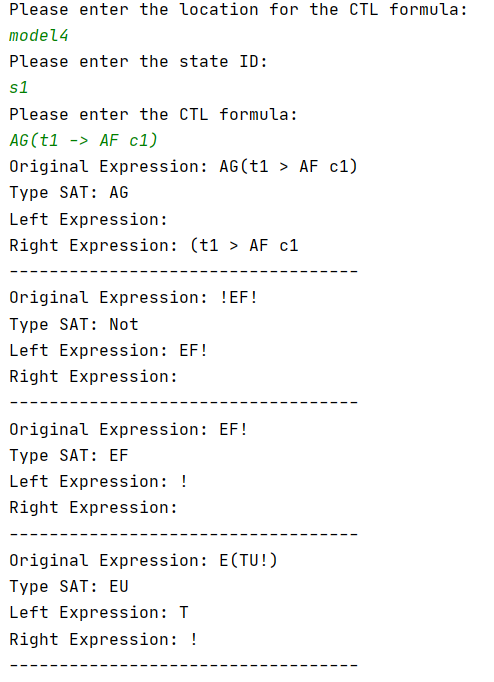
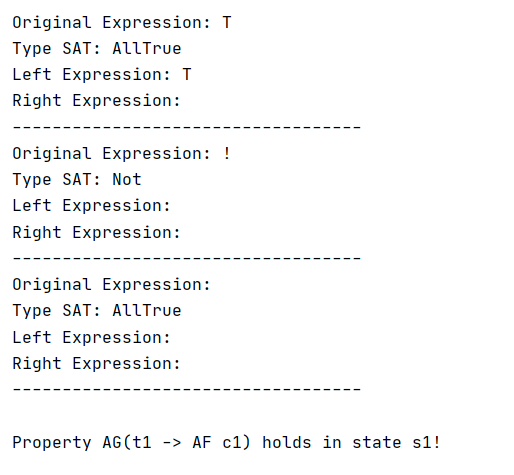
**Output 2: AXq or A(pUq) in State 3 and Model 1**

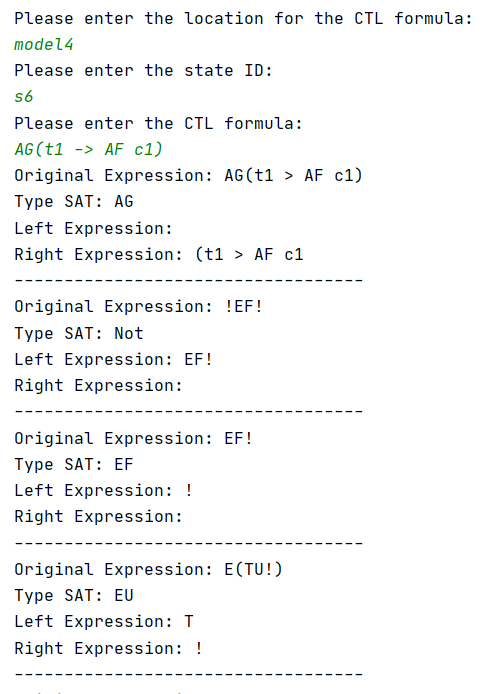
**Output 3: AXq or A(pUq) in State 3 and Model 1**

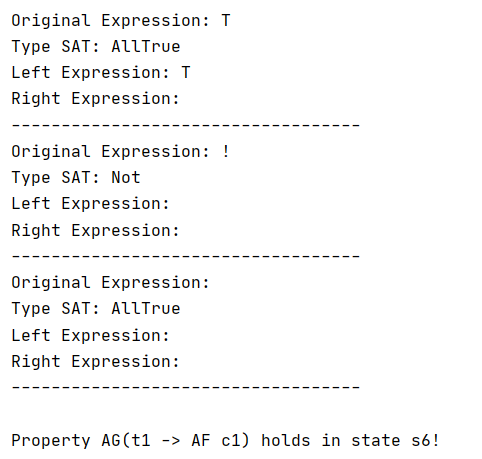
 

**Output 4: AG(t1 -> AF c1) in model 4 and state 1**

**Output 5: AG(t1 -> AF c1) in model 4 and state 6**





**Output 6: EFe and !Axe in modelmicrowave and state 1**

Please enter the location for the CTL formula:

modelmicrowave

Please enter the state ID:

s1

Please enter the CTL formula:

EFe and !AXe

Original Expression: EFe & !AXe

Type SAT: And

Left Expression: EFe

Right Expression: !AXe

-----------------------------------

Original Expression: EFe

Type SAT: EF

Left Expression: e

Right Expression:

-----------------------------------

Original Expression: E(TUe)

Type SAT: EU

Left Expression: T

Right Expression: e

-----------------------------------

Original Expression: T

Type SAT: AllTrue

Left Expression: T

Right Expression:

-----------------------------------

Original Expression: e

Type SAT: Atomic

Left Expression: e

Right Expression:

-----------------------------------

Original Expression: !AXe

Type SAT: Not

Left Expression: AXe

Right Expression:

-----------------------------------

Original Expression: AXe

Type SAT: AX

Left Expression: e

Right Expression:

-----------------------------------

Original Expression: !EX!e

Type SAT: Not

Left Expression: EX!e

Right Expression:

-----------------------------------

Original Expression: EX!e

Type SAT: EX

Left Expression: !e

Right Expression:

-----------------------------------

Original Expression: !e

Type SAT: Not

Left Expression: e

Right Expression:

-----------------------------------

Original Expression: e

Type SAT: Atomic

Left Expression: e

Right Expression:

-----------------------------------

Property EFe and !AXe holds in state s1!

**Output 7: EXh in modelmicrowave and state 2**

Please enter the location for the CTL formula:

modelmicrowave

Please enter the state ID:

s2

Please enter the CTL formula:

EXh

Original Expression: EXh

Type SAT: EX

Left Expression: h

Right Expression:

-----------------------------------

Original Expression: h

Type SAT: Atomic

Left Expression: h

Right Expression:

-----------------------------------

Property EXh does not hold in state s2!

**Output 8: AF(c or h) in modelmicrowave and state 7**

Please enter the location for the CTL formula:

modelmicrowave

Please enter the state ID:

s7

Please enter the CTL formula:

AF(c or h)

Original Expression: AF(c | h)

Type SAT: AF

Left Expression: (c | h

Right Expression:

-----------------------------------

Original Expression: (c | h

Type SAT: Or

Left Expression: c

Right Expression: h

-----------------------------------

Original Expression: c

Type SAT: Atomic

Left Expression: c

Right Expression:

-----------------------------------

Original Expression: h

Type SAT: Atomic

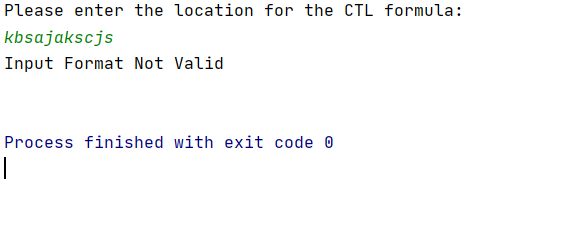
Left Expression: h

Right Expression:

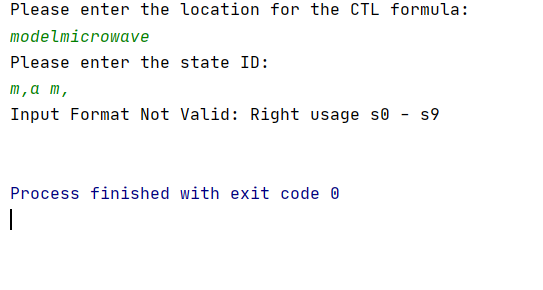
-----------------------------------

Property AF(c or h) holds in state s7!

**Output 9: Custom Validation for Input**



**Output 10: Custom Validation for Input**



**APPENDIX**

This chapter lists contents which are submitted for CTL Model Checker project which is written in Java using intellij IDE.

* ModelCheckCTL\documentation-sum20

sum20\TestFiles\CTLModelCheckerTestCases\CTLModelCheckerTestCases.docx

This file contains description of test cases in detail.

* ModelCheckCTL\documentation-sum20\ClassDiagram.png

This PNG file has the class diagram.

* ModelCheckCTL\ModelCheckCTL

This folder contains all project source code.