USING MODEL CHECKING FOR VERIFICATION OF REDUNDANCY AND INCONSISTENCY IN MARITIME LAWS

by

Muzammil Sagheer
A Thesis
Submitted to the Graduate Faculty of George Mason University
In Partial Fulfillment of The Requirements for the Degree of Master of Science Computer Science

Committee:

______________________________  Dr. Syed Abbas K. Zaidi, Thesis Director
______________________________  Dr. Alexander H. Levis, Committee Member
______________________________  Dr. Gheorghe Tecuci, Committee Member
______________________________  Dr. Hassan Gomaa, Department Chair
______________________________  Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date:____________________________ Summer Semester 2010
George Mason University
Fairfax, VA
Using Model Checking for Verification of Redundancy and Inconsistency in Maritime Laws

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Muzammil Sagheer
Bachelor of Science
Mohammad Ali Jinnah University, Karachi, Pakistan, 2004

Director: Syed Abbas K. Zaidi, Research Professor
Department of Electrical and Computer Engineering

Summer Semester 2010
George Mason University
Fairfax, VA
ACKNOWLEDGEMENTS

First of all I would like to thank God Almighty for giving me strength, wisdom and perseverance during this research and indeed throughout my life. May His blessings be upon all of us in all of the times to come.

I owe my deepest gratitude to my advisor Dr. Syed Abbas K. Zaidi for his guidance, support, insightful feedback and constructive criticism throughout this thesis. He has been an invaluable mentor during the entire period of this research. His personal advices have taught me innumerable lessons in the area of academic research.

I am also thankful to my other committee members, Dr. Alexander Levis and Dr. Gheorghe Tecuci, for their support, co-operation and their contributions to the success of this thesis.

I express my heartfelt gratitude to my parents for their constant support and encouragement during my studies. This thesis was simply impossible without the support and love of my brothers, my fiancée and her family.

Last but not the least; I would like to thank all of my friends, especially my colleagues at System Architectures Laboratory who were always there to help me whenever I needed it.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>ix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1: INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Goals and Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.3. Related Work</td>
<td>4</td>
</tr>
<tr>
<td>1.4. Thesis Outline</td>
<td>6</td>
</tr>
<tr>
<td>CHAPTER 2: PROBLEM DEFINITION</td>
<td>8</td>
</tr>
<tr>
<td>2.1. Introduction</td>
<td>8</td>
</tr>
<tr>
<td>2.2. Verifiable Characteristics of Maritime Laws</td>
<td>9</td>
</tr>
<tr>
<td>2.3. Examples of Various Cases Within Maritime Laws</td>
<td>15</td>
</tr>
<tr>
<td>CHAPTER 3: FOUNDATIONS</td>
<td>22</td>
</tr>
<tr>
<td>3.1. Introduction</td>
<td>22</td>
</tr>
<tr>
<td>3.2. Petri Net Theory</td>
<td>23</td>
</tr>
<tr>
<td>3.3. Model Checking</td>
<td>33</td>
</tr>
<tr>
<td>CHAPTER 4: PETRI NET REPRESENTATION OF RULES</td>
<td>40</td>
</tr>
<tr>
<td>4.1. Introduction</td>
<td>40</td>
</tr>
<tr>
<td>4.2. Rule Normalization</td>
<td>41</td>
</tr>
<tr>
<td>4.3. Transformation of Normalized Rules into Petri Nets</td>
<td>42</td>
</tr>
<tr>
<td>4.4. Pattern of Inconsistent and Redundant Rules in Occurrence Graphs</td>
<td>47</td>
</tr>
<tr>
<td>4.5. Conclusion</td>
<td>55</td>
</tr>
<tr>
<td>CHAPTER 5: SOLUTION TO THE PROBLEM</td>
<td>56</td>
</tr>
<tr>
<td>5.1. Introduction</td>
<td>56</td>
</tr>
<tr>
<td>5.2. Structural Analysis</td>
<td>56</td>
</tr>
</tbody>
</table>
5.3. Behavioral Analysis.................................................................................................................. 57
5.4. Conclusion............................................................................................................................... 70

CHAPTER 6: REASONING WITH MARITIME LAWS................................................................. 71
6.1. Introduction............................................................................................................................. 71
6.2. Petri Net Models for Reasoning ............................................................................................. 72
6.3. Forward Reasoning................................................................................................................ 77
6.4. Backward Reasoning............................................................................................................. 82
6.5. Use of Verification Results in Decision Making ................................................................. 84
6.6. Conclusion............................................................................................................................. 86

CHAPTER 7: APPLICATION: MARITIME LAW MANAGER AND ANALYZER ................ 87
7.1. Introduction............................................................................................................................. 87
7.2. Maritime Law Management System ..................................................................................... 88
7.3. RULER (Rule Evaluation Routine) ....................................................................................... 96
7.4. RulEx (Rule Execution Engine) ......................................................................................... 99
7.5. Conclusion............................................................................................................................. 101

CHAPTER 8: CONCLUSION AND FUTURE WORK ................................................................. 102
8.1. Conclusion............................................................................................................................. 102
8.2. Future Work........................................................................................................................... 103

APPENDIX A ............................................................................................................................... 106
A.1. Temporal Logic ...................................................................................................................... 106
A.2. Linear-time Temporal Logic (LTL) ..................................................................................... 107
A.3. Branching-time Logic ......................................................................................................... 110

APPENDIX B ............................................................................................................................... 117
B.1. Scenario Overview .............................................................................................................. 117
B.2. Sample Rules .................................................................................................................... 118
B.3. Formal Representation ....................................................................................................... 120
B.4. Shorthand Representation of Formal Rules ....................................................................... 121
B.5. Sample Queries .................................................................................................................. 123

APPENDIX C ............................................................................................................................... 124
REFERENCES.............................................................................................................................. 132
LIST OF TABLES

Table                                      Page
TABLE 2.1: SYMBOLS CORRESPONDING TO PROPOSITIONS OF QUERY 5 IN APPENDIX B…16
TABLE 2.2: SYMBOLS CORRESPONDING TO PROPOSITIONS OF QUERY 6 IN APPENDIX B…18
TABLE 2.3: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 1(A) OF OPERATION
GOOD SAMARITAN ROE, RULE 2 OF SEACOM ROE AND RULE 2 OF SOFA FOR
INDONESIA IN APPENDIX B.................................................................19
TABLE 3.1: DERIVED OPERATORS.................................................................35
TABLE 4.1: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 1(B) OF OPERATION
GOOD SAMARITAN ROE IN APPENDIX B................................................43
TABLE 4.2: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 2 AND RULE 3 OF
SOFA FOR INDONESIA IN APPENDIX B................................................45
TABLE 4.3: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 2 OF SOFA FOR
INDONESIA AND RULE 2 OF SEACOM ROE IN APPENDIX B.......................46
TABLE 4.4: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULES 1(B) OF OPERATION
GOOD SAMARITAN ROE AND SEACOM ROE IN APPENDIX B........................50
TABLE 4.5: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULES 1(B) OF JOINT
EXERCISE ROE AND SEACOM ROE IN APPENDIX B..................................53
TABLE 5.1: SYMBOLS CORRESPONDING TO PROPOSITIONS OF MUTUALLY EXCLUSIVE
SETS......................................................................................................59
TABLE 5.2: SYMBOLS CORRESPONDING TO PROPOSITIONS OF SAMPLE QUERY IN APPENDIX
B.............................................................................................................63
TABLE 5.3: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULES 1(B) OF OPERATION
GOOD SAMARITAN ROE AND SEACOM ROE IN APPENDIX B..................66
TABLE 5.4: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 2 OF SOFA FOR
INDONESIA AND RULE 2 OF SEACOM ROE IN APPENDIX B....................67
TABLE 5.5: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULES 1(B) OF JOINT
EXERCISE ROE AND SEACOM ROE IN APPENDIX B.................................68
TABLE 6.1: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE 1(B) OF OPERATION
GOOD SAMARITAN ROE IN APPENDIX B................................................72
TABLE 6.2: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULES 2 AND 3(A) OF
SEACOM ROE IN APPENDIX B..............................................................76
TABLE 6.3: SYMBOLS CORRESPONDING TO PROPOSITIONS OF RULE2 OF SEACOM ROE, RULE 1(A) OF OPERATION GOOD SAMARITAN ROE AND RULE 1(B) OF JOINT EXERCISE ROE IN APPENDIX B

TABLE B.1: SET OF PROPOSITIONS USED IN THE FORMAL REPRESENTATION OF MARITIME LAWS
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1: OVERVIEW OF COMPUTER-AIDED APPROACH</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3.1: AN ORDINARY PETRI NET</td>
<td>24</td>
</tr>
<tr>
<td>Figure 3.2: ORDINARY PETRI NET WITH MARKING</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3.3: PETRI NET WITH FIRING SEQUENCE T1, T2</td>
<td>28</td>
</tr>
<tr>
<td>Figure 3.4: PETRI NET WITH FIRING SEQUENCE T2, T1</td>
<td>28</td>
</tr>
<tr>
<td>Figure 3.5: PETRI NET after FIRING T1 and T2 CONCURRENTLY</td>
<td>29</td>
</tr>
<tr>
<td>Figure 3.6: PETRI NET after FIRING T3</td>
<td>29</td>
</tr>
<tr>
<td>Figure 3.7: OCCURRENCE GRAPH (OG) for PETRI NET in Figure 3.2</td>
<td>31</td>
</tr>
<tr>
<td>Figure 3.8: AN EXAMPLE OF A CONFLICT</td>
<td>32</td>
</tr>
<tr>
<td>Figure 3.9: AN EXAMPLE OF A CONFLICT-FREE PETRI NET</td>
<td>32</td>
</tr>
<tr>
<td>Figure 3.10: OCCURRENCE GRAPH (OG) SATISFYING EV (P4 ∧ P5)</td>
<td>38</td>
</tr>
<tr>
<td>Figure 3.11: OCCURRENCE GRAPH (OG) SATISFYING EX(P4 ∨ P5)</td>
<td>39</td>
</tr>
<tr>
<td>Figure 4.1: PETRI NET REPRESENTATION OF RULE 1(b) OF OPERATION GOOD SAMARITAN ROE</td>
<td>43</td>
</tr>
<tr>
<td>Figure 4.2: PETRI NET REPRESENTATION FOR DISJUNCTION IN CONCLUSION</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4.3: PETRI NET REPRESENTATION OF TWO MUTUALLY EXCLUSIVE CONCEPTS</td>
<td>45</td>
</tr>
<tr>
<td>Figure 4.4: PETRI NET BEFORE AND AFTER MERGING COMMON PLACES</td>
<td>47</td>
</tr>
<tr>
<td>Figure 4.5: PETRI NET REPRESENTATION OF MARITIME LAWS</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4.6: PETRI NET REPRESENTATION OF TWO INCONSISTENT RULES</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.7(a): PETRI NET after FIRING OGS_1b</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.7(b): PETRI NET after FIRING SEACOM_1b</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.8: OCCURRENCE GRAPH CORRESPONDING TO PETRI NET of Figure 4.6</td>
<td>52</td>
</tr>
<tr>
<td>Figure 4.9: PETRI NET REPRESENTATION OF TWO REDUNDANT RULES</td>
<td>54</td>
</tr>
<tr>
<td>Figure 4.10(a): PETRI NET after FIRING JE_1b</td>
<td>54</td>
</tr>
<tr>
<td>Figure 4.10(b): PETRI NET after FIRING SEACOM_1b</td>
<td>54</td>
</tr>
<tr>
<td>Figure 4.11: OCCURRENCE GRAPH CORRESPONDING TO PETRI NET of Figure 4.9</td>
<td>55</td>
</tr>
<tr>
<td>Figure 5.1: MODIFIED PETRI NET of MARITIME LAWS (HAVING NO CIRCULAR RULES)</td>
<td>64</td>
</tr>
<tr>
<td>Figure 5.2: OCCURRENCE GRAPH CORRESPONDING TO PETRI NET of Figure 5.1</td>
<td>69</td>
</tr>
<tr>
<td>Figure 6.1: PETRI NET MODEL FOR FORWARD REASONING</td>
<td>74</td>
</tr>
<tr>
<td>Figure 6.2: PETRI NET MODEL FOR BACKWARD REASONING</td>
<td>74</td>
</tr>
</tbody>
</table>
Figure 6.3: Petri Net with Bi-directional Arcs .................................................. 74
Figure 6.4: Conversion of Petri Net Having Conflict into a Conflict-free Net .... 77
Figure 6.5(a): Petri Net with Tokens (Corresponding to Initial Facts) .......... 79
Figure 6.5(b): Petri Net (After First Execution Step) ................................. 80
Figure 6.5(c): Petri Net (After Second Execution Step) .............................. 80
Figure 6.6(d): Petri Net (After Third Execution Step) ............................... 81
Figure 6.6(a): Initialized Petri Net for Backward Reasoning ..................... 83
Figure 6.6(b): Petri Net (After First Execution Step) ................................. 83
Figure 6.6(c): Petri Net (After Second Execution Step) .............................. 84
Figure 7.1: Static Entity Model (Data Model) of Maritime Law Management
System ................................................................................................. 89
Figure 7.2: User Interface of Maritime Law Management System.................. 91
Figure 7.3: Various Components within Maritime Law Management System.... 92
Figure 7.4: GUI of Maritime Law Management System Showing Search
Results ................................................................................................. 94
Figure 7.5: Generated Report for the Selected Rules .................................... 95
Figure 7.6: Input Rules from Maritime Law Management System ................ 97
Figure 7.7: Results Showing Identified Redundant Cases ............................ 98
Figure 7.8: Results Showing Identified Inconsistent Cases ......................... 99
Figure 7.9: Example of Rule Execution Process ........................................ 100
Figure 7.10: Example of Input Conditions Identification Process ............... 101
Figure A.1: A directed graph representing state-transition system .......... 108
Figure A.2: A state-transition system satisfying Xϕ .................................... 109
Figure A.3: A state-transition system satisfying Gϕ .................................... 109
Figure A.4: A state-transition system satisfying Fϕ .................................... 109
Figure A.5: A state-transition system satisfying ϕ1 U ϕ2 ............................ 110
Figure A.6: State-transition systems satisfying ϕ1 R ϕ2 ............................... 110
Figure A.7: A state-transition system whose starting state satisfies AFϕ .... 113
Figure A.8: A state-transition system whose starting state satisfies AGϕ ... 114
Figure A.9: A state-transition system whose starting state satisfies EFϕ .... 114
Figure A.10: A state-transition system whose starting state satisfies EGϕ ... 115
Figure A.11: A state-transition system whose starting state satisfies AXϕ . 115
Figure A.12: A state-transition system whose starting state satisfies E[ϕ1 U ϕ
2] ......................................................................................................... 116
ABSTRACT

USING MODEL CHECKING FOR VERIFICATION OF REDUNDANCY AND INCONSISTENCY IN MARITIME LAWS

Muzammil Sagheer, MS

George Mason University, 2010

Thesis Director: Syed Abbas K. Zaidi

A computer-aided solution for the task performed by a Maritime Lawyer is presented in this thesis. It facilitates the process of searching a set of rules from across different Maritime Laws that are applicable in a certain given situation and provides analytical capabilities that include verification of the selected rules against inconsistency as well as reasoning with them. The analyses are meant to decide the applicability of actions from the selected rule set after a careful consideration of possible inconsistencies that may appear in the applicable rules. The analytical techniques require that Maritime Laws be represented as Production Systems. The production rules are first normalized and transformed into an equivalent Petri Net representation. The structural and behavioral analysis of the Petri Net is then performed to look for properties of the net corresponding to useless, incomplete, cyclic, redundant and inconsistent cases. The structural analysis is done by an already existing approach to identify the cases of useless, incomplete and circular rules. The behavioral analysis is done by using a formal methodology of model
checking that explores the state space of the Petri Net to verify the properties corresponding to redundant and inconsistent rules. A reasoning mechanism is also proposed, which uses the Petri Net representation, to answer queries about the applicability of actions from the selected set of rules. The combined use of the verification results and the reasoning mechanism will help a Maritime Lawyer in identifying a course of action for a given situation of interest that is supported by the applicable Maritime Laws.
CHAPTER 1: INTRODUCTION

This thesis focuses on the formulation of a technique for verification of rules, selected from different Maritime Laws, against properties like consistency, completeness and redundancy. It is similar to the problem of validation and verification (V&V) of rule-bases proposed in the literature [1-13]. The objective of the approach presented in this thesis is to help a Maritime Lawyer to decide the suitability and applicability of actions in a particular situation. It, therefore, differs from the other V&V approaches [1-13] by not attempting to correct or update the rule set as a result of its application.

1.1. Motivation

A ship or, a fleet of ships, operating at sea has to abide by different Maritime Laws, e.g., Admiralty Law, Law of the Sea, etc. These laws come from different sources (e.g., international agencies, nation states, etc.) and their applicability changes from one geographical location to the other and also with time. A Maritime Lawyer is required to look into these law books to identify the applicability of action(s) in a given situation.

At present, this process is done manually by a Maritime Lawyer, who searches for all the rules that are relevant to answer queries about applicability of action(s), considering the geographical boundaries, temporal aspects, and others. From the set of identified rules, he/she figures out the applicability/suitability of action(s) and presents the results to the commander of the ship or fleet.
The entire process of information foraging and deciding the course of action from the applicable rules is a time-consuming activity. It becomes even more difficult for some time-critical decisions that need to be made, especially in a rapidly changing situation. Also, there is no guarantee that the rules are complete and consistent across all the law books, resulting in an additional task on the Maritime Lawyer to verify the completeness and consistency of the rules before arriving at a conclusion. Consequently, a computer-aided approach is needed that facilitates the process of information foraging and checking for possible conflicting cases across disparate law books. The approach can also be supplemented by a reasoning mechanism that a Maritime Lawyer can use for deciding the appropriateness or suitability of inquired action(s).

1.2. Goals and Contributions

A new technique is proposed for the identification of inconsistency and redundancy within rules selected from different law sources. It extends an earlier approach proposed in [1, 2] to overcome the issues and limitations of that methodology. A discussion on some weaknesses in the previous approach, namely the algorithms for inconsistent and redundant rules, is presented in the next section. The proposed verification technique makes use of Production Systems, where pre- and post-conditions of a production rule are given as first-order expressions, as the formal knowledge representation scheme for Maritime Laws. The formal representation is the basis for all the algorithms developed to assist a Maritime Lawyer while deciding the applicability of a course of actions. A reasoning mechanism is also proposed that can be used for
decision-making. The reasoning mechanism uses the same representation of the rules as used by the verification technique.

An overview of the computer-aided approach is shown in Figure 1.1. The formally represented set of rules, applicable in a certain given situation, is first selected from various Maritime Laws. Different analyses are then performed on these selected set of rules to verify certain properties within them. The approach first transforms the formally represented rules into an equivalent Petri Net representation, which is then initialized based on the given input situation. The structural and behavioral exploration of the Petri Net is then performed to look for properties of the net corresponding to useless, incomplete, cyclic, redundant and/or inconsistent cases. The structural analysis is done by the approach proposed in [1, 2]. The behavioral analysis is done by the new technique that employs a formal model checking methodology for the verification task. Finally, the reasoning engine is used to decide the appropriateness of inquired action or course of actions.

It should be noted that the set of verifiable properties are not considered errors in the problem addressed in this thesis. The reported cases are intended as a support provided to the Maritime Lawyer in identifying possible problematic situations where a law differs from the other applicable ones. In other words, they are supposed to provide assistance to the Maritime Lawyer in deciding the suitability of action(s) while using the reasoning engine.
1.3. **Related Work**

There have been a number of tools and techniques for validation and verification (V&V) of rule-based systems that employ production rules as the representation scheme. A large number of these proposed approaches make use of Petri Nets for the V&V task, since they provide exactly the same operational formalism as Production Systems. Zisman [3] was the first to note this similarity and use Petri Nets for the verification task. Other techniques that utilize Petri Nets were proposed by Giordanna and Saitta [4], Zhang and Nguyen [5-6], Liu and Dillon [7], Agarwal and Tanniru [8]. Zaidi and Levis [1] discuss the weaknesses of these earlier approaches. The technique proposed by them divides the set of properties into two classes: structural and behavioral. The behavioral set of properties includes redundancy and inconsistency. Their approach requires construction of a state space graph (also known as reachability graph or occurrence graph).
graph) for the Petri Net representation of a rule-base, and a search within it for identification of these properties. The proposed methodology is intended for the case where the initial marking of the Petri Net is unknown. A heuristic is used to overcome the combinatorial enumeration problem that results from the unknown input assumption for generation of the state space graph and its analysis. However, the approach proposed for behavioral analysis is, in general, too complex to be used in practice. A detailed discussion of their approach is also provided in [2]. A number of other techniques was proposed afterwards, but most of them have certain problems of their own, which are presented in the following discussion.

Yang et al. [9] propose an approach that uses the incidence matrix of the Petri Net representing a rule-base to verify properties within it. The approach is based on an ordering of rules according to some criteria defined by them before representing it with a Petri Net and using its corresponding incidence matrix for the verification task. Furthermore, the technique proposed by them works with a restricted form of rules known as Horn clauses. Also, their approach does not have the capability for verification of properties in chains of rules.

He et al. [10] and Yang et al. [11] propose an approach that uses ω-net (a special type of low-level Petri Net) for rule-base representation and the analysis of the corresponding reachability graph of ω-net for verification of properties. The issue in both the proposed techniques is that the structure of the reachability graph for a ω-net ignores paths to reachable markings, resulting in an additional work for the accurate detection of those properties [12].
Wu et al. [13] presented the use of Colored Petri Nets for rule-base representation and its reachability graph analysis for the verification task.

Ding et al. [12] recently proposed a technique that extends the approach of He et al. [10] by generating a backward reasoning forest of reachable markings in reachability graph. Their purpose of doing this is to explicitly present and record the missing reachable path information within the reachability graph of a $\omega$-net. However, their approach has the limitation of working with only those rule-bases where rules are expressed as Horn clauses.

Some other tools, reported in the literature, include MELODIA [14] that uses Boolean techniques for verification of inconsistency and redundancy in production rules, KHEOPS [15], which is a real-time rule-based system having a forward-chaining interpreter and allows for checking some formal properties, CLIPS [16] and Algernon [17], which are expert system shells that include features for verification and validation of expert systems.

A new methodology is proposed in this thesis that overcomes the weaknesses of previous approaches. The approach extends the work proposed by Zaidi and Levis [1, 2] by presenting a formal technique of model checking to capture redundant and inconsistent cases. The set of useless, incomplete and circular rules is identified by using static analysis methodology proposed by them.

1.4. Thesis Outline

The thesis is organized as follows: Chapter 2 describes the problem addressed in this thesis by providing a formal description of the verifiable properties (or
characteristics) within rules along with illustrative examples taken from some sample Maritime Laws. Chapter 3 provides background information on the essential concepts used throughout the thesis. Chapter 4 discusses the Petri Net representation to model the set of rules from law sources. Chapter 5 provides the solution to the verification problem. It briefly describes the already existing approach for identification of incomplete, useless and circular rules; and presents a detailed discussion on the new technique that employs formal model checking methodology to verify the cases of inconsistency and redundancy. Chapter 6 discusses the reasoning mechanism that has been developed for deciding the appropriateness of actions from the set of applicable rules. It also shows the usage of results from the verification to assist in decision-making. Chapter 7 presents a suite of tools that has been developed to provide a computer-aided support to a Maritime Lawyer. The suite consists of a Management System, a Rule Evaluation Routine and a Rule Execution Engine providing the functionalities of management of Maritime Laws, automated verification of rules, and reasoning with them, respectively. Finally, Chapter 8 concludes the thesis and suggests some directions for future research.
CHAPTER 2: PROBLEM DEFINITION

This chapter presents the details of the problem that is addressed in the thesis. It defines the various properties in a rule set, or across several rule sets, that a Maritime Lawyer might be interested in exploring to address a situation of interest. The formal description of the set of properties, presented in this chapter, has been compiled from different V&V approaches that were reported in the literature.

2.1. Introduction

Maritime Laws are, in essence, a set of rules expressed in natural language. If these rules can be converted into some formal representation using an appropriate knowledge representation language, then they can be used for analysis by machines. The analyses may include verification of rules for completeness and correctness using automated formal techniques as well as reasoning with them using a reasoning engine.

The rules in Maritime Laws can be regarded as Condition-Action rules which help a Maritime Lawyer to determine when, where, and how certain actions should be taken. This suggests the use of Production Systems to be a better choice compared to other formalisms for encoding Maritime Laws, since such a representation is useful in action selection problems.

A Production System consists of a set of Condition-Action rules, also known as production rules, a working memory that contains the current state of knowledge, and a
rule interpreter that executes the rules, consequently, making changes in the working memory [18]. The conditions and actions of production rules are expressed as First-Order Logic (FOL) formulas. Such types of Production Systems are referred to as First-Order Production Systems [18]. The reason for choosing FOL to encode the conditions and actions of production rules is because of its adequacy of representation (i.e. expressive power) [19]. It has already been mentioned in Chapter 1 that the technique proposed, in this thesis, is an extension of a previous work presented in [1, 2]. However, the methodology proposed in [2] uses different Petri Net representations for the rules in Propositional Logic (PL) and for the rules expressed in First-Order Logic (FOL). This thesis presents a technique that uses the same Petri Net representation for both the expressions. Since every FOL formula having a finite set of constant symbols can be converted to an equivalent Propositional formula [20], so the FOL statements in production rules are transformed into PL statements by instantiating their variables with known constant values. This process of transforming FOL statements into PL statements is known as propositionalization [20], which makes it possible to exploit the Petri Net representation for PL expressions that has been presented in [1, 2] to deal with both the cases. A small sample of rules from different Maritime Law sources, supporting a hypothetical maritime scenario [21], is provided in Appendix B. The set of corresponding propositionalized production rules are given in Section B.3.

2.2. **Verifiable Characteristics of Maritime Laws**

The Production System representing the set of rules from across different Maritime Laws is likely to have inconsistency, redundancy, etc., since the constituent
rules come from different law sources. A rule from one Maritime Law may allow an action, whereas another rule addressing the same situation from a different Maritime Law may prohibit it. There can also be cases in which two Maritime Laws, applicable in a given situation, may say the same thing. Under these circumstances, a Maritime Lawyer has to suggest if an action can be taken or not. In other words, he/she has to be certain about his/her decisions after a careful consideration of the various characteristics that may appear in the set of applicable Maritime Laws. This section outlines such characteristics and provides a formal description of them by demonstrating them using Production System as representation technique for rules. Such systems work by using forward-chaining mechanism that executes the rule by matching conditions of a rule to the current set of facts (state of knowledge), thereby, deducing new set of facts, which get added to existing set of knowledge [18]. It should be noted that the discussion in the remaining section makes use of a hypothetical set of production rules for describing various properties. Examples of such properties in Maritime Laws are provided in Section 2.3.

2.2.1. Inconsistent Rules

Inconsistent rules result in conflicting facts [10]. The execution of rule set having inconsistent rules result in new facts that either conflict with one another or conflict with original fact(s).

Example 1:

Consider the two rules shown below. Given A, B, C and D (as facts), these two rules become applicable, leading to two conflicting conclusions (P and ¬P) if executed. It
should be noted that the formal representation used in this thesis makes use of the symbol ‘∧’ to represent the logical AND and ‘¬’ to represent the negation (i.e. NOT) operator.

Given facts: \( A \land B \land C \land D \)

Rule 1: \( A \land B \rightarrow P \)

Rule 2: \( C \land D \rightarrow \neg P \)

Example 2:

Consider another set of facts and rules presented below. Based on the given information, the first rule becomes applicable leading to a conclusion which makes the second rule applicable as well that leads to a conclusion (\( \neg A \)) contradicting with original information (\( A \)).

Given facts: \( A \land B \land D \)

Rule 1: \( A \land B \rightarrow P \)

Rule 2: \( P \land D \rightarrow \neg A \)

To put it another way, a rule set free of inconsistent rules will never infer conflicting information. It should be noted that the inconsistent case presented in the second example was not caused because of two rules concluding inconsistent information. In fact, the cause of inconsistency was the conclusion from an applicable rule contradicting with the original set of facts.

2.2.2. Redundant Rules

Redundancy refers to the presence of multiple copies of the same rule or the presence of set of rules that lead to the same effect (output), from the same input conditions [1, 2].
**Example 1:**

Consider the set of rules shown below. The two provided rules are identical except that their input conditions are arranged differently. Based on the given set of facts (A and B), which makes their input conditions true, same conclusion is derived by their execution.

Given facts: \( A \land B \)

Rule 1: \( A \land B \rightarrow P \)

Rule 2: \( B \land A \rightarrow P \)

**Example 2:**

The rule set shown below is another example of redundancy in which the first three rules have the same effect as the last rule while processing the same input conditions (A, B, C and D).

Given facts: \( A \land B \land C \land D \)

Rule 1: \( A \land B \rightarrow P \)

Rule 2: \( C \land D \rightarrow Q \)

Rule 3: \( P \land Q \rightarrow R \)

Rule 4: \( A \land B \land C \land D \rightarrow R \)

### 2.2.3. Incompleteness

Incompleteness refers to a situation when there is not enough information in the rule-base to answer a query of interest [22]. Some of the previous approaches in V&V literature have further categorized this case into Rules with Unknown Conditions and Rules with Useless Conclusions i.e. both the categories determine incompleteness in a
The incompleteness in this thesis is defined as the situation when the system either does not have rules that can reach the goal (i.e. answer a query) or it has a set of rules that can arrive at some desired goal, however, their inputs are unknown to the system. The example below illustrates the concept.

**Example:**

Consider the set of rules shown below. Based on the given set of facts (A, B, C and D), it is not possible for the system to make an assertion R, since one of the input conditions (S) of the last rule, which can assert R, is unknown. This situation may have originated because of some missing rule that can make assertion S from the input information or due to an insufficient amount of information provided in the original set of facts.

**Given facts:**  \( A \land B \land C \land D \)

**Rule 1:**  \( A \land B \rightarrow P \)

**Rule 2:**  \( C \land D \rightarrow Q \)

**Rule 3:**  \( P \land Q \land S \rightarrow R \)

### 2.2.4. Useless Rules

This case has already been mentioned in the previous discussion. They refer to the rules that do not produce useful results. In other words, these are the rules whose output is not of interest to the user and there is no rule or set of rules leading from their outputs to useful conclusions [1, 2].
Example:

Consider a case, where the set of facts and rules are as provided below and a user is interested to know if assertion P can be made or not. It should be noted that the formal representations in this thesis follow Prolog notation to express such queries, which are denoted as ?proposition. From the given information, it can be seen that the system can reach to a decision about P using the first rule, however, the remaining rules are applicable as well, whose outputs are not relevant to the query in this case.

Given facts: \[ A \land B \land C \land D \]

Rule 1: \[ A \land B \rightarrow P \]

Rule 2: \[ C \land D \rightarrow Q \]

Rule 3: \[ A \land B \land C \land D \rightarrow R \]

Goal: \[ ?P \]

2.2.5. Circular Rules

Circularity refers to the presence of rules that will lead a system to get trapped in executing them again and again. The following example illustrates such a case.

Example 1:

Consider the two rules shown below. Given A (as fact), the first rule becomes applicable leading to a conclusion which makes the second rule applicable as well. The conclusion of the second rule leads back to the first rule to become applicable again.

Given facts: \[ A \]

Rule 1: \[ A \rightarrow B \]

Rule 2: \[ B \rightarrow A \]
Another case of circularity is defined as the presence of rules that are circularly dependent on each other resulting in a deadlock [1, 2].

**Example 2:**

The set of rules shown below illustrates the case of a deadlock. Given A and B as facts, the first rule cannot execute since it requires an input condition R that can be obtained by executing the second rule, which in turn requires the execution of first rule to make an assertion P to become applicable in the given situation.

Given facts:  \( A \land B \)

Rule 1:  \( A \land R \rightarrow P \)

Rule 2:  \( B \land P \rightarrow R \)

### 2.3. Examples of Various Cases Within Maritime Laws

This section uses some of the rules from a sample of Maritime Laws and test queries, provided in Appendix B, to show the properties that have been described in the previous section within those rules. The textual as well as formal representation of the rules is provided, whenever they are referred in the following discussion. It is assumed that the Maritime Laws have already been transformed into a Production System. A shorthand representation using symbols for long propositional statements of production rules is used in the entire thesis. The table listing these symbols and their corresponding propositions is provided in Appendix B.4.

**Example 1:**

This example uses a test query (from Appendix B.5), to show a case in which conflicting conclusions are reached from two different applicable Maritime Laws. The
textual as well as formal representation of the query is provided below. The symbols used in the logical representation and their corresponding propositions are listed in Table 2.1.

“Is it allowed to follow a pirate vessel in territorial waters, if it commits a hostile act?” (Query 5, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

Formally,

\[
\text{Given Facts: } P_5 \land P_9 \land P_{10} \land P_{14} \\
\text{Goal: } ?P_{13}, ?P_{15}
\]

Table 2.1: Symbols Corresponding to Propositions of Query 5 in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P_9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P_{10}</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P_{13}</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P_{14}</td>
<td>hostile_act(V1)</td>
</tr>
<tr>
<td>P_{15}</td>
<td>not_pursue(V1, C1)</td>
</tr>
</tbody>
</table>
Some of the rules (from Appendix B) relevant to answer this query are stated below.

“It is not permitted to pursue a pirate ship into territorial waters unless the hostile act was committed against U.S. forces.” (Rule 1b, SEACOM ROE)

Formally,

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \]  
(Rule 1b, SEACOM ROE)

“Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.” (Rule 1b, Operation Good Samaritan ROE)

Formally,

\[ P_5 \land P_9 \land P_{10} \rightarrow P_{15} \]  
(Rule 1b, Operation Good Samaritan ROE)

The above mentioned rules are applicable in the provided situation; however, they totally contradict one another since one rule prohibits following pirates in territorial waters (P15), whereas the other allows such an action (P13) in the given condition i.e. when the vessel commits a hostile act against U.S. forces.

\[(P_5 \land P_9 \land P_{10} \land P_{14})\]

\[ P_5 \land P_9 \land P_{10} \rightarrow P_{15} \]  
(Rule 1b, Operation Good Samaritan ROE)

\[ P_{15} \]

\[(P_5 \land P_9 \land P_{10} \land P_{14})\]

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \]  
(Rule 1b, SEACOM ROE)

\[ P_{13} \]
**Example 2:**

This example presents an instance of incompleteness i.e. a situation in which a certain inquired action cannot be answered from the available set of input conditions (facts). It uses the following query to illustrate the incompleteness.

“Is it permitted to stop and search a vessel that commits a hostile act?” (Query 6, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

Formally,

Given facts: P1 ∧ P14

Goal: ?P6

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>normal_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>

The rules relevant to this query are: Rule 2 from SOFA for Indonesia; part of Rule 1(a) from Operation Good Samaritan ROE, and Rule 2 from SEACOM ROE. The textual as well as logical representation of these rules is provided below.
“Only ships that have been declared pirate ships by the policing entity are allowed to be searched.” (Rule 2, SOFA for Indonesia)

Formally,
\[ P5 \rightarrow P6 \quad \text{(Rule 2, SOFA for Indonesia)} \]

“A ship shall be considered a pirate vessel if it commits a hostile act against U.S. forces.” (Rule 1a rephrased, Operation Good Samaritan ROE)

Formally,
\[ P1 \land P14 \rightarrow P5 \quad \text{(Rule 1a, Operation Good Samaritan ROE)} \]

“U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.” (Rule 2, SEACOM ROE)

Formally,
\[ P5 \rightarrow P6 \quad \text{(Rule 2, SEACOM ROE)} \]

Table 2.3: Symbols Corresponding to Propositions of Rule 1(a) of Operation Good Samaritan ROE, Rule 2 of SEACOM ROE and Rule 2 of SOFA for Indonesia in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>normal_vessel(V1)</td>
</tr>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>
Suppose that Rule 1(a) of Operation Good Samaritan ROE was missing from the provided set of rules. Although, Rule 2 from SOFA for Indonesia and Rule 2 from SEACOM ROE authorizes such an action (P6) for pirate vessels (P5), but they cannot be applied since there is no information to determine if the vessel is a pirate vessel or not. However, it would have been possible to figure this out from the given set of facts, if the missing rule was also present.

Without Rule 1(a) from Operation Good Samaritan ROE,

\[(P1 \land P14)\]

\[P5 \rightarrow P6\]  (Rule 2, SOFA for Indonesia), (Rule 2, SEACOM ROE)

\[
\begin{align*}
\end{align*}
\]

{} \[
\begin{align*}
\end{align*}
\]

With Rule 1(a) from Operation Good Samaritan ROE,

\[(P1 \land P14)\]

\[P1 \land P14 \rightarrow P5\]  (Rule 1a, Operation Good Samaritan ROE)

\[P5 \rightarrow P6\]  (Rule 2, SOFA for Indonesia), (Rule 2, SEACOM ROE)

\[
\begin{align*}
P6
\end{align*}
\]

**Example 3:**

This example uses the same query and set of rules, provided in the previous example, to present a case where rules from different Maritime Laws say the same thing in a given situation (i.e. an instance of redundancy).

From the three rules presented in previous example, it can be seen that Rule 2 from SOFA for Indonesia and Rule 2 from SEACOM ROE are identical (in terms of both
input condition and conclusion). The two rules conclude that U.S. forces are allowed to search any vessel, which has been declared pirate.

(P1 \land P14)

\[ \text{P1 \land P14 \rightarrow P5} \quad \text{(Rule 1a, Operation Good Samaritan ROE)} \]

\[ \text{P5 \rightarrow P6} \quad \text{(Rule 2, SOFA for Indonesia)} \]

\[ \text{P6} \]

(P1 \land P14)

\[ \text{P1 \land P14 \rightarrow P5} \quad \text{(Rule 1a, Operation Good Samaritan ROE)} \]

\[ \text{P5 \rightarrow P6} \quad \text{(Rule 2, SEACOM ROE)} \]

\[ \text{P6} \]
CHAPTER 3: FOUNDATIONS

This chapter provides information on the foundational concepts used throughout the thesis. It includes discussion on Petri Nets and Formal Model Checking theory. The material presented in this chapter on Petri Nets has been taken from [23] and on Formal Model Checking from [24, 25] with minor editorial changes.

3.1. Introduction

Petri Nets, whether colored or ordinary, have been used by several researchers for the verification of rule-based systems [1-13]. The advantage of using Petri Nets, for the problem addressed in this thesis, can be summarized as follows:

- Petri Nets have well-established formal mechanisms for modeling and checking the properties of a concurrent system, which can be used to model the applicable set of rules and verify them against incompleteness, inconsistency and/or redundancy.

- The execution nature of Petri Nets makes it possible to perform reasoning from the applicable set of rules, once they have been transformed into a Petri Net model.

- The graphical structure of Petri Nets can help visualize the transition firing sequence of the net, which can serve as an explanation facility for a Maritime Lawyer.
As a result, the use of Petri Net formalism to model the rules within Maritime Laws provides a way to approach the overall problem, which includes verification of applicable rules from Maritime Laws and reasoning with them.

The next section presents a discussion on Ordinary Petri Nets, which are used to model the rules within Maritime Laws in this thesis, and some other fundamental concepts from Petri Net theory that are used in the approach being proposed.

3.2. Petri Net Theory

3.2.1. Ordinary Petri Net

Definition (Ordinary Petri Net):- An Ordinary Petri Net is a directed graph that consists of:

- a finite set of places \( P \),
- a finite set of transitions \( T \) such that \( P \cap T = \emptyset \), and
- a set of directed arcs \( E \) from places to transitions or from transitions to places, such that there exists at most one arc between a place and a transition (i.e. the multiplicity of arcs is 1)

\[ E \subseteq (P \times T) \cap (T \times P) \]

The two sets \( P \) and \( T \) constitute the entire set of nodes \( N \) of the Petri Net.

In the graphical representation of Petri Net, places are denoted by elliptical nodes, transitions by square nodes and arcs by arrows.

Definition (Pre-set and Post-set):- For an element \( x \) of \( N (= P \cup T) \), its pre-set \( \bullet x \) is defined by:

\[ \bullet x = \{ y \in N \mid (y, x) \in E \} \]
and its post-set $x^*$ is defined by:

$$x^* = \{ y \in N \mid (x, y) \in E \}$$

Since the arcs in a Petri Net connect places to transitions or transitions to places, the pre-set of a transition ($\cdot t$) can be defined as the set of all places that are input to it and the post-set of a transition ($t \cdot$) as the set of its output places. The pre-set and post-set of a place ($\cdot p$ and $p \cdot$) can be defined in a similar way.

**Example:**

Consider the Petri Net shown in Figure 3.1.

![Figure 3.1: An Ordinary Petri Net](image)

The set of places $P$, the set of transitions $T$ and the set of directed arcs $E$ for this Petri Net are:

$P = \{P1, P2, P3, P4, P5, P6\}$

$T = \{T1, T2, T3\}$

$E = \{(P1, T1), (P2, T1), (T1, P4), (P3, T2), (T2, P5), (P4, T3), (P5, T3), (T3, P6)\}$
The pre-sets and post-sets of places and transitions for this Petri Net are:

- Pre-set of P1 = {}  
- Pre-set of P2 = {}  
- Pre-set of P3 = {}
- Pre-set of P4 = {T1}  
- Pre-set of P5 = {T2}  
- Pre-set of P6 = {T3}
- Post-set of P1 = {T1}  
- Post-set of P2 = {T1}  
- Post-set of P3 = {T2}
- Post-set of P4 = {T3}  
- Post-set of P5 = {T3}  
- Post-set of P6 = {}
- Pre-set of T1 = {P1, P2}  
- Pre-set of T2 = {P3}  
- Pre-set of T3 = {P4, P5}
- Post-set of T1 = {P4}  
- Post-set of T2 = {P5}  
- Post-set of T3 = {P6}

### 3.2.2. States and Behavior of a Petri Net

The states of a Petri Net are defined by its markings.

**Definition (Marking):** A marking $M$ of a Petri Net is a mapping that assigns a non-negative number to each place of the Petri Net.

$$M: P \rightarrow \mathbb{N} \text{ where } \mathbb{N} = \{0, 1, 2, \ldots\}$$

A place $p$ is said to be marked by a marking $M$ if $M(p) > 0$.

Graphically, a marked place is denoted by the presence of a token (•) or some tokens in that place.

**Example:**

Figure 3.2 shows a Petri Net with the indicated marking. It is the same Petri Net shown in Figure 3.1 but with tokens. Places with tokens (i.e. P1, P2 and P3) correspond to the ones that are marked by the given marking $M$. The marking of a Petri Net can also be expressed as a vector with a specified ordering of the places. The indicated marking $M$ of Figure 3.2 can be represented as:

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T$$
corresponding to the ordering P1, P2, P3, P4, P5 and P6.

\[
M(P1) = M(P2) = M(P3) = 1 \quad M(P4) = M(P5) = M(P6) = 0
\]

**Figure 3.2: Ordinary Petri Net with Marking**

A marking defines the state of a Petri Net. Changes in state of the Petri Net are caused by the occurrence of its transitions. The set of states and the events that cause the change from one state to another describe the dynamic behavior of a Petri Net.

**Definition (Marked Petri Net):** A Marked Petri Net is a Petri Net equipped with a marking called initial marking \( M_0 \).

The example Petri Net shown in Figure 3.2 is a marked Petri Net with initial marking \( M_0 = [1 \ 1 \ 1 \ 0 \ 0 \ 0]^T \).

**Definition (Transition Enablement):** A transition \( t \) is enabled by a marking \( M \) if all the places in the pre-set of transition \( (\bullet t) \) are marked by \( M \).

**Definition (Transition Firing):** If a transition \( t \) is enabled, it can fire or execute, which transforms the marking \( M \) into a new Marking \( M' \).
$$M \xrightarrow{t} M'$$

The new marking $M'$ is defined for each place $p$ as:

$$M'(p) = \begin{cases} 
  M(p) - 1 & \text{if } p \in \cdot t - t \\
  M(p) + 1 & \text{if } p \in t - \cdot t \\
  M(p) & \text{otherwise.}
\end{cases}$$

The firing of a transition may lead to another transition becoming enabled. Consider the same Petri Net shown in Figure 3.2, based on the given marking, transitions $T_1$ and $T_2$ are enabled concurrently. The enabled transitions are shown highlighted in the figure. With two enabled transitions at the same time, there are two possibilities of transition firing; either one of them fires first followed by the execution of the other, or both of them fire simultaneously. Figures 3.3, 3.4 and 3.5 show the three firing sequences. In all situations, the resulting marking of the Petri Net is $[0 \ 0 \ 0 \ 1 \ 1 \ 0]^T$, which makes another transition $T_3$ enabled.

(a) After Firing $T_1$
Figure 3.3: Petri Net with Firing Sequence T1, T2

Figure 3.4: Petri Net with Firing Sequence T2, T1
**Definition** (Occurrence Sequence): A finite sequence of transitions $t_1, \ldots, t_k$ enabled at marking $M$ is called finite occurrence sequence, if they transform the marking $M$ into $M_k$, with $M_1, M_2, M_{k-1}$ as intermediate markings.

$$M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_k} M_k$$

The expression can also be written as $M \xrightarrow{\sigma} M_k$, where $\sigma = t_1, \ldots, t_k$. 
It should be noted that transitions enabled concurrently at a marking, result in multiple occurrence sequences. The Petri Net of Figures 3.2 to 3.6 shows such a case, which has two occurrence sequences T1, T2, T3 and T2, T1, T3 resulting in the same end marking.

Since, the occurrence sequence σ transforms one marking M into another marking M', it can also be said that M' is reachable from M for such a sequence σ. The behavior of a Petri Net is characterized by these changes of markings. This implies that the overall behavior of a Petri Net, for a given marking M, is specified by the set of all occurrence sequences enabled at M.

**Definition (Occurrence Graph):** An Occurrence Graph (OG), also known as state space, for a Petri Net with an initial marking (also called a Petri Net System) is a directed-graph that consists of:

- a set of nodes corresponding to the set of reachable markings from M₀,
- a set of arcs between nodes that correspond to transitions from one marking to another.

In other words, it is a graphical representation of all the set of occurrence sequences from M₀ and all the reachable markings reachable from M₀, depicting the overall behavior of a Petri Net System. Figure 3.7 shows the Occurrence Graph (OG) corresponding to the Petri Net of Figure 3.2.

The dynamic nature of Petri Nets introduces a number of behavioral properties that are closely related to the structure of the net [2]. The following discussion presents one such property known as Conflict.
**Definition (Conflict):** The transitions of a Petri Net are said to be in a conflict at a marking $M$ if and only if $M$ enables these transitions but not concurrently i.e. firing of one transition will disable the others.

The cause of such a conflict is the presence of place(s) in the Petri Net, which are input to multiple transitions, resulting in a case of contention between their output transitions once they get marked.

---

**Figure 3.7: Occurrence Graph (OG) for Petri Net in Figure 3.2**
**Example:**

Figure 3.8 shows an example of a conflict with the help of a marked Petri Net. The placement of token in P1 corresponding to the initial marking enables both the transitions in its post-set (i.e. T1 and T2); however, the execution of any of the two enabled transitions (e.g. T1) will cause the other (T2) to get disabled.

\[
M_0 = [1 \ 0 \ 0]^T
\]

**Figure 3.8: An Example of a Conflict**

**Definition** (Conflict-free Net):- A Petri Net is said to be conflict-free, if \( \forall p \in P, |p^*| \leq 1 \).

**Example:**

Figure 3.9 shows an example of a conflict-free Petri Net in which each place has at most one transition in its post-set.

**Figure 3.9: An Example of a Conflict-free Petri Net**
3.3. Model Checking

Model Checking is an approach for formal verification of finite state concurrent systems. It starts by specifying a property of a system as a formal logical expression and then interprets this logical expression over the state space of the system to establish if the given property holds in the system’s behavior or not. In other words, it establishes using an exhaustive search of the state space if the system’s behavior is a model of the input logical expression or not. The process always ends with a yes or no answer indicating the presence or absence of the specified property within the system. The technique was initially proposed by Clarke and Emerson [26, 27] and by Sifakis and Queille [28].

The model checking method requires the following steps:

a) Modeling: The first step is to represent the system by a model accepted by the model checker.

b) Specification: The second step requires specification of properties that needs to be checked in the system. The properties are normally expressed by temporal logic, which can assert how the behavior of a system changes over time.

c) Verification: The verification step is automated. It exhaustively searches the state space of modeled system to check if the specified property exists in the system. It provides an error trace in case of a property violation that can be used to identify the point of error.

3.3.1. ASK-CTL

ASK-CTL is an extension of Computation Tree Logic (CTL), a class of temporal logics, which is used to model check Petri Nets. In other words, it is used for
specification of properties that need to be checked in systems represented by Petri Net models. The specified properties are checked against the state space (or Occurrence Graph) of a Petri Net. An implementation of ASK-CTL model checking algorithm is available in CPN Tools [29], which is a tool for creating, simulating and analyzing Petri Nets.

Since the Occurrence Graph of a Petri Net model carries information on nodes as well as edges, the CTL extension in ASK-CTL allows specification of these properties with both state and transition information. A quick reference on CTL is provided in Appendix A.

An ASK-CTL expression, as defined in [30], is a state or a transition formula. The definition of both the categories is as follows:

**Definition (ASK-CTL State Formula):** An ASK-CTL state formula \( A \) is defined by the following Backus Naur Form:

\[
A ::= \text{tt} \mid \neg A \mid A \lor A \mid A \land A \mid EU(A, A) \mid AU(A, A) \mid \alpha | <B>
\]

where \( \text{tt} \) is interpreted as true, \( \alpha \) is a function mapping from markings to Boolean values, \( B \) is a transition formula (defined below), \( U \) (until) is the standard temporal operator, \( A \) and \( E \) (for-all and exist respectively) are the path quantifiers. The temporal operator \( U \) is used in combination with one of the path quantifiers \( A \) and \( E \), e.g. the formula \( EU(A_I, A_2) \) expresses the existence of a path from a given state (marking) where \( A_I \) holds to a state where \( A_2 \) holds. Similarly, the formula \( AU(A_I, A_2) \) requires the property to hold along all paths from a given initial state [30].
**Definition** (ASK-CTL Transition Formula):- An ASK-CTL transition formula $B$ is defined by the following Backus Naur Form:

$$B ::= tt | \neg B | B \lor B | B \land B | \text{EU}(B, B) | \text{AU}(B, B) | \beta | <A>$$

where $\beta$ is a function mapping from transitions to Boolean values and $A$ is a state formula.

The functions $\alpha$ and $\beta$, mentioned in the formulae, are required to be implemented in Standard ML programming language [31, 32]. They return a Boolean value after checking some property of the state and transition, respectively. In addition to the above mentioned operators, ASK-CTL library offers a number of other derived operators to construct complex formulae for checking properties of a system. A list of some of these operators, along with their description, is provided in Table 3.1.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos $(A) \equiv \text{EU}(tt, A)$</td>
<td>It is <em>possible</em> to reach a state where $A$ holds.</td>
</tr>
<tr>
<td>Inv $(A) \equiv \neg\text{Pos} (\neg A)$</td>
<td>$A$ holds in every reachable state; $A$ is <em>invariant</em>.</td>
</tr>
<tr>
<td>Ev $(A) \equiv \text{AU}(tt, A)$</td>
<td>For all paths, $A$ holds within a finite number of steps; $A$ is <em>eventually</em> true.</td>
</tr>
<tr>
<td>Along $(A) \equiv \neg\text{Ev} (\neg A)$</td>
<td>There exist a path which is either infinite or ends in a dead state, along which $A$ holds in every state.</td>
</tr>
<tr>
<td>$&lt;B&gt; A \equiv &lt;B \land &lt;A&gt;&gt;$</td>
<td>There exist an immediate successor state $M$ satisfying $A$ and $B$ holds on the transition between the current state and $M$.</td>
</tr>
</tbody>
</table>
With this brief introduction to ASK-CTL, the model checking problem using ASK-CTL can be described as:

Given an Occurrence Graph of a Petri Net and an ASK-CTL property (formula) $p$, determine if $p$ holds in the initial state of OG, i.e. if $\text{OG} \vDash p$.

As per this definition, a Petri Net model is said to satisfy a property if the ASK-CTL formula that describes the property can be shown to hold in the initial state of the OG corresponding to the Petri Net being examined. The complexity of ASK-CTL model checking algorithm, implemented in CPN Tools, is linear in the product of the size of formula and the size of state space – $O(N(|V| + |E|))$, where $N$ is the length of the formula, $|V|$ is the number of nodes (or states) and $|E|$ is the number of edges in the Occurrence Graph.

### 3.3.2. Examples of ASK-CTL

This section presents some examples of ASK-CTL expressions interpreted over the Occurrence Graph (OG) of Figure 3.7. The given Occurrence Graph corresponds to the Petri Net model of Figure 3.2.

| EX($A$) $\equiv <t> A$ | There exists an immediate successor state in which $A$ holds. Read ‘Exist Next’. |
| AX($A$) $\equiv \neg\text{EX} \neg A$ | $A$ holds in all immediate successor states, if any. Read ‘For All Next’. |
**Example 1:**

Consider a case that requires verification of the property defined by the following ASK-CTL formula within the given Petri Net model (of Figure 3.2).

\[ \text{Ev } (P4 \land P5) \]

**Description:** In all possible futures, P4 and P5 hold together.

It has already been mentioned that the verification process in model checking methodology performs a search to establish if the specified property exists in the state space. A search of this type in the OG of a Petri Net System is demonstrated in Figure 3.10. It shows that the given property exists in the example Petri Net, since there is a reachable state along all paths from the initial state of OG that has P4 and P5 true at the same time. The resulting state is shown highlighted in the figure along with all the paths leading to it.
Example 2:

Consider another example that demonstrates the presence of a property defined by the following ASK-CTL formula in the given Petri Net System.

\[
\text{EX}(P4 \lor P5)
\]

Description:- It is possible to have P4 or P5 to hold in some of the next states.

The result of state space exploration process for the given ASK-CTL expression is illustrated in Figure 3.11. It shows the existence of an immediate node from the initial
state of the Occurrence Graph which satisfies the given ASK-CTL formula indicating the presence of the specified property in the Petri Net model.

Figure 3.11: Occurrence Graph (OG) Satisfying EX(P4 ∨ P5)
CHAPTER 4: PETRI NET REPRESENTATION OF RULES

This chapter explains the process of transforming production rules to a Petri Net model that will be used for verification of certain properties within it.

4.1. Introduction

A production rule, also known as a Condition-Action rule (or a Situation-Action rule) [20], is of the form:

\[ \alpha \rightarrow \beta \] 

(4.1)

where the left-hand side of the expression (or antecedent) represents the condition \( \alpha \) and the right-hand side (or consequent) represents the corresponding action \( \beta \). It states that whenever the condition \( \alpha \) is satisfied, then the action \( \beta \) can be taken as a consequence.

It has already been mentioned in Chapter 2 that the verification technique, proposed in this thesis, makes use of the Petri Net representation defined for Propositional Logic Systems in [1, 2]. The discussion that follows in the next two sections is taken from [1, 2]. It describes a methodology of transforming formally represented rules (in this case, production rules) into an equivalent Petri Net representation.
4.2. Rule Normalization

The technique works with rules, expressed in the form (4.1), whose antecedents are organized in Disjunctive Normal Form (DNF) and the conclusions in Conjunctive Normal Form (CNF). This assumption does not impose any restriction, since any schema constructed with the help of logical connectives and operators can be converted to an equivalent Disjunctive Normal Form (DNF) or an equivalent Conjunctive Normal Form (CNF). An example of such a rule is:

\[
\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n \rightarrow \beta_1 \land \beta_2 \land \ldots \land \beta_m
\]  

(4.2)

where \( \alpha \) and \( \beta \) are conjunctions and disjunctions of atomic propositions respectively.

The rule specified above (in 4.2) can be split into the following set of rules:

\[
\begin{align*}
\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n & \rightarrow \beta_1 \\
\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n & \rightarrow \beta_2 \\
\ldots & \\
\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n & \rightarrow \beta_m
\end{align*}
\]

It can also be split into the following set of rules:

\[
\begin{align*}
\alpha_1 & \rightarrow \beta_1 \land \beta_2 \land \ldots \land \beta_m \\
\alpha_2 & \rightarrow \beta_1 \land \beta_2 \land \ldots \land \beta_m \\
\ldots & \\
\alpha_n & \rightarrow \beta_1 \land \beta_2 \land \ldots \land \beta_m
\end{align*}
\]

It follows that the rule expressed in form (4.2) can be split into the following \((n \times m)\) rules:

\[
\alpha_1 \rightarrow \beta_1
\]
This leads to a set of rules, expressed in a form, in which the antecedents are conjunction of one or more propositions and the consequents are disjunction of one or more action propositions.

### 4.3. Transformation of Normalized Rules into Petri Nets

The normalized set of rules is then transformed into an equivalent Petri Net representation. For each rule, a Petri Net is generated where the conditions of a rule are represented by input places, satisfying the definition of conjunctive operator, and the conclusions by output places. The labels on transitions correspond to the rules they represent. An example of Petri Net representation for Rule 1(b) of Operation Good Samaritan ROE (taken from Appendix B) is shown in Figure 4.1. The referred rule is described as follows:

“Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.” (Rule 1b, Operation Good Samaritan ROE)

Formally,

\[ P5 \land P9 \land P10 \rightarrow P15 \]  

(Rule 1b, Operation Good Samaritan ROE)
Table 4.1: Symbols Corresponding to Propositions of Rule 1(b) of Operation Good Samaritan ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P15</td>
<td>not_pursue(V1, C1)</td>
</tr>
</tbody>
</table>

Figure 4.1: Petri Net Representation of Rule 1(b) of Operation Good Samaritan ROE

The consequent of a rule may have more than one proposition connected by disjunctive connective. For such cases, each output $\beta_i$ with $k$ individual propositions (connected by disjunction), is required to be represented by additional $(2^k-1)$ output transitions leading to $k$ places (representing those individual propositions), that corresponds to all possible combinations of propositions according to the definition of
disjunctive connective. Figure 4.2 shows a representation for such a case. It should be noted that the Petri Net model shown in Figure 4.2 corresponds to a hypothetical rule, which is not a part of sample scenario (of Appendix B) and is presented to explain the concept.

![Figure 4.2: Petri Net Representation for Disjunction in Conclusion](image)

The negation of a proposition (i.e., \( \neg P \) or not \( P \)) is represented by a place different from the place representing the proposition (i.e., \( P \)). Figure 4.3 shows an example of such a case using two rules from the sample scenario, which are stated as follows:

“Only ships that have been declared pirate ships by the policing entity are allowed to be searched.” (Rule 2, SOFA for Indonesia)

Formally,

\[ P_5 \rightarrow P_6 \]  
(Rule 2, SOFA for Indonesia)

“Labeling a ship as non-compliant does not authorize U.S. forces to search vessel.” (Rule 3, SOFA for Indonesia)
Formally,

$$P8 \rightarrow P7 \quad \text{(Rule 3, SOFA for Indonesia)}$$

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P7</td>
<td>not_search(V1)</td>
</tr>
<tr>
<td>P8</td>
<td>non_compliant_vessel(V1)</td>
</tr>
</tbody>
</table>

Table 4.2: Symbols Corresponding to Propositions of Rule 2 and Rule 3 of SOFA for Indonesia in Appendix B

$$P7 \equiv \neg P6$$

![Petri Net Representation](image)

Figure 4.3: Petri Net Representation of Two Mutually Exclusive Concepts

A token in a place represents the truth assignment of a proposition. If all the input places of a transition, representing the input conditions of a rule, have tokens then the
transition (rule) is enabled and can fire (execute), making the output place (consequent) true.

Finally, a single Petri Net representation for the entire rule set is obtained by merging (fusing) all the common places of Petri Nets representing individual rules. An example of such a process using Rule 2 of SOFA for Indonesia and Rule 2 of SEACOM ROE is shown in Figure 4.4. The textual as well as logical representations for the two rules are provided below.

“Only ships that have been declared pirate ships by the policing entity are allowed to be searched.” (Rule 2, SOFA for Indonesia)

Formally,

\[ P_5 \rightarrow P_6 \]  
(Rule 2, SOFA for Indonesia)

“U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.” (Rule 2, SEACOM ROE)

Formally,

\[ P_5 \rightarrow P_6 \]  
(Rule 2, SEACOM ROE)

Table 4.3: Symbols Corresponding to Propositions of Rule 2 of SOFA for Indonesia and Rule 2 of SEACOM ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_5 )</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>( P_6 )</td>
<td>search(V1)</td>
</tr>
</tbody>
</table>
Figure 4.4: Petri Net before and after merging common places

The Petri Net representation for the entire set of sample rules, presented in Appendix B.3, was generated based on the defined mapping. The resulting Petri Net model is shown in Figure 4.5, which will be used for verification of properties described in Chapter 2.

4.4. **Pattern of Inconsistent and Redundant Rules in Occurrence Graphs**

A discussion on Occurrence Graph (OG) has already been presented in Chapter 3. This section presents the patterns of inconsistent and redundant rules within the Occurrence Graphs of Petri Nets. Such patterns can help in formulating the two properties using ASK-CTL, which can be checked against the state space of a Petri Net, representing Maritime Laws. The patterns for incomplete, useless and circular rules within a Petri Net model have already been presented in [1, 2].
IS: SOFA for Indonesia
SEACOM: South East Asia Command ROE
OGS: Operation Good Samaritan ROE
JE: Joint Exercise ROE

Figure 4.5: Petri Net Representation of Maritime Laws
4.4.1. Inconsistent Rules

As per the definition of inconsistency, such a case occurs when two conflicting concepts become true in a given situation. In a Petri Net, these cases appear in the form of two conflicting places getting marked by a token in the set of occurrence sequences from $M_0$ (where $M_0$ represents the initial marking corresponding to a given situation). As a result, the two conflicting places will get marked in the corresponding OG as well. An example of such a case appearing in a Petri Net and the corresponding OG is shown in Figures 4.6, 4.7 and 4.8. The Petri Net model shown in Figures 4.6 and 4.7 corresponds to Rules 1(b) of Operation Good Samaritan ROE and SEACOM ROE, which are stated below.

“Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.” (Rule 1b, Operation Good Samaritan ROE)

Formally,

$$P5 \land P9 \land P10 \rightarrow P15$$

(Rule 1b, Operation Good Samaritan ROE)

“It is not permitted to pursue a pirate ship into territorial waters unless the hostile act was committed against U.S. forces.” (Rule 1b, SEACOM ROE)

Formally,

$$P5 \land P9 \land P10 \land P14 \rightarrow P13$$

(Rule 1b, SEACOM ROE)
Table 4.4: Symbols Corresponding to Propositions of Rules 1(b) of Operation Good Samaritan ROE and SEACOM ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
<tr>
<td>P15</td>
<td>not_pursue(V1, C1)</td>
</tr>
</tbody>
</table>

Assume that the input conditions of both these rules are true, which makes both of them applicable. The two occurrence sequences, corresponding to the enabled transitions, result in conflicting places (P13 and P15) getting marked, implying a contradiction in conclusion. Figures 4.7(a) and (b) show such a case by the presence of tokens in P13 and P15 for the two occurrence sequences. This conflicting behavior is represented by the corresponding OG as well (refer to Figure 4.8), which shows contradiction in the form of P13 and P15 getting marked in two different reachable states. The states having P13 and P15 marked are shown highlighted in the figure.
Figure 4.6: Petri Net Representation of Two Inconsistent Rules

Figure 4.7(a): Petri Net after Firing OGS_1b

Figure 4.7(b): Petri Net after Firing SEACOM_1b
4.4.2. Redundant Rules

In a Petri Net, redundancy appears in the form of several paths from place(s) with multiple outputs to place(s) with multiple inputs [1, 2]. In an Occurrence Graph of a Petri Net model, such cases appear in the form of multiple distinct paths from one state (marking) to another state [1, 2]. An example of a Petri Net and the OG representation for a redundant case is given in Figures 4.9, 4.10 and 4.11. The Petri Net model shown in Figures 4.9 and 4.10 corresponds to Rules 1(b) of Joint Exercise ROE and SEACOM ROE, which are stated below.

“It is prohibited to follow any suspected pirate ship into territorial waters unless the pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.” (Rule 1b, Joint Exercise ROE)

Formally,

\[ P5 \wedge P9 \wedge P10 \wedge P14 \rightarrow P13 \]  

(Rule 1b, Joint Exercise ROE)
“It is not permitted to pursue a pirate ship into territorial waters unless the hostile act was committed against U.S. forces.” (Rule 1b, SEACOM ROE)

Formally,

\[ P5 \land P9 \land P10 \land P14 \rightarrow P13 \]  

(Rule 1b, SEACOM ROE)

Table 4.5: Symbols Corresponding to Propositions of Rules 1(b) of Joint Exercise ROE and SEACOM ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>

The two rules mentioned above, are identical in terms of both input conditions and conclusion. Assume a situation in which these input conditions are true, in which case both the rules will lead to same conclusion on execution. The transition firing process for the two rules is shown in Figures 4.10(a) and (b). An Occurrence Graph representation corresponding to the Petri Net is provided in Figure 4.11, which shows redundancy in the form of two distinct paths from one state to another.
Figure 4.9: Petri Net Representation of Two Redundant Rules

Figure 4.10(a): Petri Net after Firing JE_1b

Figure 4.10(b): Petri Net after Firing SEACOM_1b
4.5. Conclusion

A method for transforming production rules into a Petri Net representation has been presented in this chapter. The Petri Net representation for the set of rules will be used for different analyses to verify them against the properties outlined in Chapter 2. A discussion on the composition of inconsistent and redundant cases within an Occurrence Graph of a Petri Net has also been presented. Such patterns will serve in formulating the behavioral analysis technique for the two cases that will be presented in the next chapter.
CHAPTER 5: SOLUTION TO THE PROBLEM

This chapter presents the solution to the verification problem that was defined in Chapter 2.

5.1. Introduction

Once the set of rules have been converted into a Petri Net representation, various analyses can be used to explore the properties of the Petri Net. This chapter presents the ones for the verification of properties corresponding to the set of cases discussed in Chapter 2. The set of properties can be divided into two classes: structural and behavioral. The Petri Net verification process using the structural properties has already been presented in [1, 2]. This chapter presents a new technique for behavioral analysis of Petri Nets to capture properties that correspond to redundancy and inconsistency in a rule-base.

5.2. Structural Analysis

The verification of rules using structural analysis, also known as static analysis, is used to reveal portions of a Petri Net structure that correspond to some of the rule patterns, mentioned in Chapter 2. The rule types identified by the structural analysis include circular, useless and incomplete rules.
The concept that makes structural analysis different from behavioral analysis is that it is independent of a marking. It explores the structure of the graph that represents a Petri Net model instead of an Occurrence Graph that corresponds to the dynamics or execution of the Petri Net, to identify certain characteristics. The structural exploration of a Petri Net, presented in [1, 2], looks at the invariants in the Petri Net to identify possible loops or dead ends corresponding to circularity in rules. In addition, it checks for the directed paths from input places (i.e. places corresponding to input situation) to output places (places corresponding to actions in question) to verify the cases of useless and incomplete rules. This evaluation of Petri Net structure is done by making use of incidence matrix of the Petri Net. The incidence matrix is a mathematical representation of a Petri Net, defining connectivity among its nodes, thus making it possible to verify the above-mentioned properties within its structure.

5.3. Behavioral Analysis

Behavioral analysis, also known as dynamic analysis, makes use of an Occurrence Graph to capture properties that correspond to redundant and inconsistent cases. It uses the state space of a Petri Net system to look for patterns among states or inside a state description that correspond to the two cases.

The new approach proposed in this thesis makes use of formal model checking with the state space of a Petri Net system representing a rule set. An introduction of formal model checking and ASK-CTL has already been presented in Chapter 3. This section presents the use of ASK-CTL for verification of inconsistent and redundant rules. The technique uses ASK-CTL formulae, representing certain behavioral properties, to
look for patterns in a state space that correspond to redundancy and inconsistency. The patterns corresponding to the two cases have already been discussed and presented in Chapter 4.

The approach presented in this thesis requires that circular instances of rules should be identified and isolated prior to applying the state space analysis. This can be easily achieved by the application of structural analysis as presented in [1, 2]. This is due to the fact that the presence of circular instances of rules in a Petri Net will result in cycles within the corresponding Occurrence Graph, which may lead to an infinite execution of the model checking algorithm, when applied to the Occurrence Graph of such a Petri Net.

5.3.1. ASK-CTL Formulae for Verification of Inconsistency and Redundancy

The construction of an Occurrence Graph, as mentioned in its definition, depends on the initial marking $M_0$ of a Petri Net. The working memory of the Production System representing a set of rules can be used to mark the conditions (i.e., places) that correspond to an initial marking of the Petri Net. An Occurrence Graph can then be constructed with the Petri Net representation of the rule set, marked by the tokens in the places that correspond to the conditions in the working memory. It is assumed for this approach that the sets of mutually exclusive concepts $\mu$ are also provided along with the set of rules. The sets of mutually exclusive concepts will be used in identification of inconsistent cases. Some examples of mutually exclusive sets from the sample rules (of Appendix B) are given below.
\[ \mu_1 = \{P6, P7\} \]

\[ \mu_2 = \{P13, P15\} \]

Table 5.1: Symbols Corresponding to Propositions of Mutually Exclusive Sets

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P7</td>
<td>not_search(V1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P15</td>
<td>not_pursue(V1, C1)</td>
</tr>
</tbody>
</table>

The rest of the section presents the ASK-CTL formulae for verification of redundancy and inconsistency and provides an explanation as to what they mean. It also shows the soundness and completeness of both the formulae i.e. they can be used to capture only and every instance of the two cases. An implementation of the two formulae (provided as a pseudo code) is presented in Appendix C.

(a) Verification of Inconsistency

Given an Occurrence Graph of a Petri Net with an initial marking \( M_0 \), the following ASK-CTL formula can be used to verify if the rule set, represented by the Petri Net, has inconsistent cases in it.

\[ \text{Pos } (\alpha_1) \land \text{Pos } (\alpha_2) \]  

(5.1)
where

\( \alpha_1 \) is a state function that checks if \( u \) is true,

\( \alpha_2 \) is a state function that checks if \( v \) is true,

such that \( \{ u, v \} \in \mu \)

**Description:-** It is possible that in the entire state space, two mutually exclusive concepts belonging to a set \( \mu_i \) are true.

Since the list of mutually exclusive concepts is maintained in this approach, the presented ASK-CTL formula is checked for every mutually exclusive set to identify every possible case of inconsistency within the rule set.

**Soundness:-** It has been mentioned in Chapter 2 that a rule set free of inconsistent rules will never result in conflicting knowledge. In other words, it can be said that the Occurrence Graph generated for such a conflict-free rule set will never have two mutually exclusive places from a set \( \mu_i \) as marked. This implies that the ASK-CTL formula, given by expression 5.1, does not capture any case other than inconsistency, since it checks for the truth values of the mutually exclusive concepts (propositions) in the state space.

**Completeness:-** The marked places in the states of an Occurrence Graph correspond to the propositions that are true in a certain given situation. Since the ASK-CTL formula, presented in expression 5.1, is checked for every mutually exclusive set in an Occurrence Graph, it guarantees that every possible conflicting case that can occur in a certain given situation will be identified. It is imperative that the list of mutually exclusive concepts be exhaustive and complete.
(b) Verification of Redundancy

Given an Occurrence Graph of a Petri Net with an initial marking $M_0$, the following ASK-CTL formula can be used to identify if the rule set, represented by Petri Net system, has redundancy.

$$\text{Pos } (\alpha_1 \land \text{Pos } (\alpha_2) \land (\alpha_3)) \tag{5.2}$$

where

$\alpha_1$ is a state function that checks if state $i$ has multiple output paths,

$\alpha_2$ is a state function that checks if state $j$ has multiple reachable paths from state $i$,

and

$\alpha_3$ is a function that checks if there are at least two occurrence sequences between $i$ and $j$ having uncommon transitions.

**Description:** It is possible to reach a state that has multiple output paths from which there exists another reachable state having multiple input paths such that at least two of these paths between them are distinct (i.e. the occurrence sequences correspond to different sets of transitions).

**Soundness:** The pattern for redundancy has already been presented in Chapter 4, which describes it as a set of multiple distinct paths from one state to another state within an OG. In fact, such a composition within an OG only corresponds to a redundant case. This implies that the ASK-CTL formula, given by expression 5.2, is guaranteed to capture
only redundant cases, since it looks for such patterns of multiple distinct paths from one state to another state.

Completeness: The ASK-CTL formula, given by expression 5.2, only identifies a single redundant case when applied to an Occurrence Graph. However, it is still possible to capture the entire set of redundant cases. Since an ASK-CTL formula is required to be implemented in SML programming language, the programming constructs of looping structures can be used to capture the entire set of redundant cases. The idea is to place the formula within a looping structure (e.g. while, do while) that captures a new redundant case, not captured before, in an iterative fashion. The pseudo code of an SML program implementing this function is presented in Appendix C. Such a modification will, however, increase the time complexity of the process by a factor of ‘r’ where r is the total number of redundant cases in the rule set.

5.3.2. Application of ASK-CTL Formulae on the Example Petri Net

This section presents an example to illustrate the new proposed technique. It uses the rules from Maritime Laws (provided in Appendix B). The input situation for the example is described by the following query, which has been taken from sample scenario.

“What are the permissible actions that can be taken against a hostile act of pirate vessel located in territorial waters?” (Rephrased Query 5, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

The given query can be formally represented as,

\[ P5 \land P9 \land P10 \land P14 \]
Table 5.2: Symbols Corresponding to Propositions of Sample Query in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>

The Petri Net representation for the sample Maritime Laws has already been presented in Chapter 4 (see Figure 4.5). There was a circular instance of a rule present in the given Petri Net representation, which was identified by the static analysis approach. The given Petri Net representation was modified by removing the identified circular instance comprising of second rule from Rule 1(a) of Operation Good Samaritan ROE. The modified version of Petri Net model is shown in Figure 5.1. Tokens were added to the places of Petri Net that correspond to the given situation (i.e. P5, P9, P10 and P14). An Occurrence Graph of the Petri Net corresponding to the given marking was generated. The resulting Occurrence Graph is shown in Figure 5.2.

The execution of the two SML programs (presented in Appendix C), which are implementations of the defined ASK-CTL formulae, for the given OG revealed the presence of inconsistency and redundancy as well as their cause in the set of rules for the given input.
Figure 5.1: Modified Petri Net of Maritime Laws (having No Circular Rules)

IS: SOFA for Indonesia
SEACOM: South East Asia Command ROE
OGS: Operation Good Samaritan ROE
JE: Joint Exercise ROE
(a) Inconsistent Rules

The sets of mutually exclusive concepts, which were manually identified for the test case, were provided as an input to Program 1 (of Appendix C). The identified sets of mutually exclusive concepts have already been presented in Section 5.3.1 (see Table 5.1). The execution of the program uncovered that Rule 1(b) of SEACOM ROE and Rule 1(b) of Operation Good Samaritan ROE were applicable in the given situation, both leading to two conflicting conclusions (P13 and P15). The textual as well as formal representation for the two identified rules is provided below.

“It is not permitted to pursue a pirate ship into territorial waters unless the hostile act was committed against U.S. forces.” (Rule 1b, SEACOM ROE)

Formally,

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \]

(Rule 1b, SEACOM ROE)

“Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.” (Rule 1b, Operation Good Samaritan ROE)

Formally,

\[ P_5 \land P_9 \land P_{10} \rightarrow P_{15} \]

(Rule 1b, Operation Good Samaritan ROE)
Table 5.3: Symbols Corresponding to Propositions of Rules 1(b) of Operation Good Samaritan ROE and SEACOM ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
<tr>
<td>P15</td>
<td>not_pursue(V1, C1)</td>
</tr>
</tbody>
</table>

(b) Redundant Rules

Similarly, the SML program for redundant case (Program 2 of Appendix C) was executed to identify the presence of redundancy in rules. There were two sets of redundant rules identified by the program. The first set included Rule 2 from SOFA for Indonesia and Rule 2 from SEACOM ROE, whereas the second set consisted of Rule 1(b) from SEACOM ROE and Rule 1(b) from Joint Exercise ROE. The identified sets of rules along with their formal representation are given below.

First Set of Rules:

“Only ships that have been declared pirate ships by the policing entity are allowed to be searched.” (Rule 2, SOFA for Indonesia)
Formally,

\[ P_5 \rightarrow P_6 \] \hspace{1cm} \text{(Rule 2, SOFA for Indonesia)}

“U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.” \text{(Rule 2, SEACOM ROE)}

Formally,

\[ P_5 \rightarrow P_6 \] \hspace{1cm} \text{(Rule 2, SEACOM ROE)}

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
</tbody>
</table>

Table 5.4: Symbols Corresponding to Propositions of Rule 2 of SOFA for Indonesia and Rule 2 of SEACOM ROE in Appendix B

Second Set of Rules:

“It is prohibited to follow any suspected pirate ship into territorial waters unless the pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.” \text{(Rule 1b, Joint Exercise ROE)}

Formally,

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \] \hspace{1cm} \text{(Rule 1b, Joint Exercise ROE)}

“It is not permitted to pursue a pirate ship into territorial waters unless the hostile act was committed against U.S. forces.” \text{(Rule 1b, SEACOM ROE)}
Formally,

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \quad \text{(Rule 1b, SEACOM ROE)} \]

Table 5.5: Symbols Corresponding to Propositions of Rules 1(b) of Joint Exercise ROE and SEACOM ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P_9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P_{10}</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P_{13}</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P_{14}</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>

The Occurrence Graph shown in Figure 5.2 highlights the pattern corresponding to the second set of redundant rules identified by the new proposed technique.
Figure 5.2: Occurrence Graph Corresponding to Petri Net of Figure 5.1


5.4. Conclusion

A technique for formal verification of set of rules has been presented in this chapter. It employs an existing approach for structural analysis of Petri Nets representing a rule set and a new formal approach for behavioral analysis. The result of the technique is a report of identified set of rules that are inconsistent, redundant, circular, incomplete and/or useless.
CHAPTER 6: REASONING WITH MARITIME LAWS

This chapter presents another analysis technique that can assist in the process of decision-making from the set of applicable Maritime Laws.

6.1. Introduction

Once the formally represented set of rules, selected from multiple Maritime Laws, is verified against the specified properties, it can be used to answer certain queries. Several reasoning engines are available that can provide such a functionality e.g. CLIPS [16], Jess [33], Algernon [17] etc. However, it is also possible to use the Petri Net constructed for the verification algorithms for the reasoning purpose. Since a Petri Net is an executable model, it can be used to develop a reasoning mechanism for the rule-base. Such an approach can be useful, in particular, to answer queries about:

- Identification of all possible actions that can be taken in a given situation.
- Identification of applicable rules in a given situation.
- Appropriateness/suitability of a certain action in a given situation.
- Identification of required conditions for a certain action to be taken.

Additionally, the graphical structure of the Petri Net can also serve to visualize the transition firing sequence of the net, which can act as an explanation facility for a Maritime Lawyer.
This chapter presents the Petri Net representations that can be used to answer the above mentioned queries. It also discusses the usage of results from the proposed verification technique in decision-making process.

6.2. Petri Net Models for Reasoning

Two different Petri Net representations are required for forward and backward reasoning, the two most commonly used methods for reasoning with rules. The forward reasoning determines the conclusion using a rule and its input conditions whereas backward reasoning determines the condition using a rule and its conclusion. Figures 6.1 and 6.2 illustrate this concept using Rule 1(b) of Operation Good Samaritan ROE, which is stated below.

“Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.” (Rule 1b, Operation Good Samaritan ROE)

Formally,

\[ P5 \land P9 \land P10 \rightarrow P15 \]  
(Rule 1b, Operation Good Samaritan ROE)

Table 6.1: Symbols Corresponding to Propositions of Rule 1(b) of Operation Good Samaritan ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
</tbody>
</table>
The Petri Net shown in Figure 6.1 (a-b) corresponds to the model for forward reasoning, since it uses the information (tokens) in its input places (see Figure 6.1a) to determine new information (marked by a token in the output place) on execution (see Figure 6.1b). Similarly, the Petri Net model for backward reasoning is given in Figure 6.2 showing the flow of information in backwards direction (i.e. from conclusion of original rule to its input conditions). The rest of the thesis uses the terminology of reverse net to refer to the Petri Net model for backward reasoning. The two reasoning mechanisms are considered because some queries require forward reasoning to achieve the goal whereas others require the use of backward reasoning, as will be explained in the following discussion.

It should be noted that the possibility of using bi-directional arcs in Ordinary Petri Nets cannot be considered, since such a representation would only result in an infinite execution of the Petri Net, once it gets triggered. The Petri Net model using such an approach is shown in Figure 6.3.
Figure 6.1: Petri Net Model for Forward Reasoning

Figure 6.2: Petri Net Model for Backward Reasoning

Figure 6.3: Petri Net with Bi-directional Arcs
The use of forward or backward reasoning depends on the type of query being answered. From the description of both reasoning mechanisms, it can be seen that forward reasoning is appropriate for the first two queries (described in Section 6.1), since the goal is to identify all the possible set of facts (actions) and applicable rules. For the case where it is required to identify the suitability of an action in a given situation, a more reasonable approach is to use backward reasoning mechanism in order to avoid deducing irrelevant facts (actions). The approach can be used by adding a token in the place of reverse Petri Net model that corresponds to the action in question and executing it to find out if the new marked places correspond to the provided input situation, thus, giving support to the goal (action). However, it is also possible to use forward reasoning mechanism for this case by neglecting the execution of all those rules that have been identified as useless rules using verification approach, thus, preventing the mechanism to draw irrelevant conclusions. The last query (mentioned in Section 6.1) requires the use of backward reasoning mechanism, since it involves the identification of all possible combinations of input conditions that support a certain goal (action).

It should be noted that the Petri Net representation for the set of rules, which was described in Chapter 4, is not used for reasoning, since the generated Petri Net model has place(s) that are input to multiple transitions. The presence of such places, as described in Chapter 3, will lead to a conflict between its output transitions, once those places get marked. The problem is resolved by converting the generated Petri Net model into a conflict-free net. This is done by replacing each place having multiple transitions in its post-set with the same number of places as were the number of output transitions in the
original Petri Net model. An example of a Petri Net having conflict and its equivalent conflict-free net, using two sample rules (Rule 2 and Rule 3(a) from SEACOM ROE), is shown in Figure 6.4. The two rules are stated below.

“U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.” (Rule 2, SEACOM ROE)

Formally,

\[ P_5 \rightarrow P_6 \quad \text{(Rule 2, SEACOM ROE)} \]

“The collection of biometric data is authorized for ships declared to be pirate ships.” (Part of Rule 3a, SEACOM ROE)

Formally,

\[ P_2 \land P_5 \rightarrow P_4 \quad \text{(Rule 3a, SEACOM ROE)} \]

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>biometric_data(D1)</td>
</tr>
<tr>
<td>P4</td>
<td>collect(D1)</td>
</tr>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
</tbody>
</table>
Such a modification is required for both the models (i.e. the original one and its reversed version) before using them for reasoning.

### 6.3. Forward Reasoning

Once the modified Petri Net model for forward reasoning mechanism is obtained, it can be used to derive new set of information (actions). Tokens are added to the places that correspond to the given input situation. An execution of the Petri Net is performed to deduce new set of information from the existing one until no more rules (transitions) can be executed. Such an approach can be used to identify all the possible set of actions that can be taken in a given input situation. It can also be used to identify all the rules that are applicable in a given situation based on fired transitions. An execution sequence of Petri Net corresponding to Rule 2 of SEACOM ROE, Rule 1(b) of Joint Exercise ROE and first part of Rule 1(a) of Operation Good Samaritan ROE for the input situation (P1, P9, P10 and P14) is shown in Figure 6.5(a-d). The places that get marked after each
execution step corresponds to the new set of information derived. The three rules used to demonstrate forward reasoning mechanism are stated below.

“U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.” (Rule 2, SEACOM ROE)

Formally,

\[ P_5 \rightarrow P_6 \]  

(Rule 2, SEACOM ROE)

“It is prohibited to follow any suspected pirate ship into territorial waters unless pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.” (Rule 1b, Joint Exercise ROE)

Formally,

\[ P_5 \land P_9 \land P_{10} \land P_{14} \rightarrow P_{13} \]  

(Rule 1b, Joint Exercise ROE)

“A ship shall be considered a pirate vessel if hostile acts against U.S. forces is directly observed by U.S. forces.” (Rule 1a rephrased, Operation Good Samaritan ROE)

Formally,

\[ P_1 \land P_{14} \rightarrow P_5 \]  

(Rule 1a, Operation Good Samaritan ROE)
Table 6.3: Symbols Corresponding to Propositions of Rule 2 of SEACOM ROE, Rule 1(a) of Operation Good Samaritan ROE and Rule 1(b) of Joint Exercise ROE in Appendix B

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>normal_vessel(V1)</td>
</tr>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
</tbody>
</table>

**Input Facts:** P1, P9, P10, P14

Figure 6.5(a): Petri Net with Tokens (Corresponding to Initial Facts)
Input Facts: P1, P9, P10, P14
Derived Facts: P5
Rules Used: OGS_1a_1

Figure 6.5(b): Petri Net (After First Execution Step)

Input Facts: P1, P9, P10, P14
Derived Facts: P5, P6
Rules Used: OGS_1a_1, SEACOM_2

Figure 6.5(c): Petri Net (After Second Execution Step)
Such a reasoning mechanism can also be used to answer queries about the applicability of a certain action in a given situation. Since it looks at all the possible set of facts based on the existing ones; however, it is computationally more expensive to derive all conclusions in addition to the one action under consideration. A formal description of the useless rules and their detection methodology has already been presented in Chapter 2 and 5 respectively. The set of useless rules can assist, in this case, to decide which of the enabled rules need to be executed in order to reach the desired goal. Since these are the rules that do not produce useful conclusions, the forward reasoning mechanism can avoid them.
6.4. **Backward Reasoning**

Backward reasoning mechanism is used in a similar way for the remaining set of queries, but with a reversed Petri Net model. Tokens are added to the place corresponding to an action in question. An execution of the resulting Petri Net is performed to identify the set of input conditions that support the action, until no more transitions can be executed. The set of places marked by a token after every execution step corresponds to input conditions that support the desired action. Figure 6.6 demonstrates an example of such a case with the help of the second and the third rules used in the forward reasoning example in the previous section (i.e. Rule 1a of Operation Good Samaritan ROE and Rule 1b of Joint Exercise ROE).

The queries that involve identification of the applicability of a certain action can be worked out in a similar way. However, after each execution step, an additional process is performed that matches the set of identified input conditions against initial facts to see if they support the desired action.
Action in Question: P13

Figure 6.6(a): Initialized Petri Net for Backward Reasoning

Action in Question: P13

Applicable Rules: JE_1b
Required Input Conditions: \{P5, P9, P10, P14\}

Figure 6.6(b): Petri Net (After First Execution Step)
6.5. **Use of Verification Results in Decision Making**

The two reasoning mechanisms, presented in the previous sections, can be used to answer queries that have been mentioned in Section 6.1. However, the reasoning process alone cannot be used for decision-making. It has already been mentioned in Chapter 2 that in order to make correct decisions from Maritime Laws, it is required to consider various properties within them as well. This section presents the combined use of verification results and reasoning mechanism that can assist a Maritime Lawyer in decision-making process. In other words, the use of reasoning mechanism will help him/her in answering certain queries while the verification results will serve as the information that provide him/her assurance about the correctness of the answers. The rest of the section presents the usage of some of the verification results, which include
inconsistent and redundant rules, while finalizing the actions. The use of circular and useless rules has already been discussed previously, in the case of avoiding infinite execution of model checking algorithm (refer to Chapter 5) and preventing a forward reasoning mechanism from drawing irrelevant facts, respectively. The remaining case of incompleteness can be used to figure out if a certain query can be answered from the given set of input information.

6.5.1. Inconsistent Rules

The results from the reasoning mechanism can be checked in the list of inconsistent set to verify if there exists any rule contradicting the inferred information. Such a confirmation is required in order to be certain that there is no applicable rule opposing it. The situation in which no such rule is found, a Maritime Lawyer can conclude to go for the action, based on the fact that there does not exist any rule contradicting it, thus providing a consistency check for his/her recommendation. On the contrary, a situation in which there exists some rule contradicting it, he/she may take a more cautious approach in deciding about the applicability of an action.

6.5.2. Redundant Rules

The presence of redundant rules, especially across different law books, adds more support to the set of applicable actions. Once the results from the reasoning mechanism are verified from the inconsistent set, they can be checked in the redundant set to find out if there exists more rules leading to the same information, thus, providing additional support to the concluded action.
6.6. Conclusion

A reasoning mechanism has been presented in this chapter to answer certain type of queries that would be of interest to the Maritime Lawyer. It is capable of performing both forward and backward reasoning. The presented reasoning mechanism along with the results from verification technique will assist the Maritime Lawyer in decision-making from the set of applicable Maritime Laws.
CHAPTER 7: APPLICATION: MARITIME LAW MANAGER AND ANALYZER

An implementation of the approach given in the earlier chapters, for the analysis and providing a decision support to a Maritime Lawyer, is presented in this chapter.

7.1. Introduction

A Maritime Law Handbook (Maritime Law source document) is a collection of rules used by a Maritime Lawyer. If these rules are extracted from the source documents and stored in an organized way (e.g. database), then it becomes possible for the Maritime Lawyer to search and extract the ones that are relevant to a situation. He/she can use the selected set of rules and perform various analyses on them, such as the ones presented in the previous two chapters.

In this chapter, a suite of tools is presented that has been developed to provide a computer-aided solution for the task of a Maritime Lawyer. The suite consists of a Maritime Law Management System, a Rule Evaluation Routine, and a Rule Execution Engine providing the functionalities for management of Maritime Laws, automated verification of rules, and reasoning from them, respectively. The motivation for the computer-aided approach is to make the task easier and less cumbersome for the Maritime Lawyer.
7.2. Maritime Law Management System

In the previous chapters, it was assumed that the collection of Maritime Laws and rules within them were available for the analytical techniques that were presented. The analyses were applied on their formal representation. This section presents an application known as Maritime Law Management System, which provides storage facility for various Maritime Laws and rules within them and facilitates selection of rules specific to a particular task, serving as the source of input to the already presented analytical techniques.

An effort was made to identify the organization of different set of information, available in the Maritime Laws, within a database and discover the set of functionalities that would be required by a Maritime Lawyer for its management. The following set of functionalities was identified to be required in the system:

- A facility to introduce/delete law sources and rules or make editorial changes in the existing ones.
- Storage for both a textual as well as a formal representation of rules.
- A search mechanism to facilitate the selection of rules specific to a task.
- A facility to store and attach additional important information to rules (e.g. maps, images, notes).
- A report generation facility.

Some knowledge management tools were also studied, in parallel, to find out if there was an already available tool that could provide the identified set of functionalities.
An application, named Zotero [34], was chosen as a possible candidate for Maritime Law Management System, since it provided most of the above mentioned functionalities. It is a free, open source add-on for the Firefox browser, developed by Center for History and New Media at George Mason University. It enables users to manage bibliographic data and other electronic objects such as books and articles. The application was further customized and changes were made in the database structure at its back-end to fully meet the identified requirements. The resulting data model of the application is presented in Figure 7.1, which shows the organization of various entities within the application and the connections among them.

![Figure 7.1: Static Entity Model (Data Model) of Maritime Law Management System](image-url)
The rest of the section describes various features and functionalities provided by the Maritime Law Management System with the help of some of its screenshots.

Figure 7.2 shows the user interface of Maritime Law Management System which has four basic sections (or panels). Section 1 displays the list (or collection) of Maritime Laws stored in the repository. The list of rules within the selected Maritime Law source is displayed in the area of Section 2. Selection of a rule shows its details in the area of Section 3, which include information about the title of rule, its textual as well as logical representation, the set of attachments, and tags attached to it. Tags are keywords or phrases, created by the user, for cataloging (or indexing) the rules. A rule can have multiple tags attached to it. The set of tags, attachments, and the two representations are assigned to the rules at the time they are entered in the system. Finally, section 4 is the area that displays the entire list of tags assigned to the stored rules. The list of tags assists in searching the rules from the repository by enabling a user to select a tag or a combination of them, which extracts all the rules having those tags assigned to them. In other words, it can be said that they provide a more efficient search mechanism compared to other search procedures. For example, the tag of “piracy” on selection will pull all the rules having that tag assigned to them. The presented example is based on the assumption that the tag was created and assigned to rules from various Maritime Law sources that talk about piracy.
Figure 7.2: User Interface of Maritime Law Management System

Figure 7.3 presents another view of the first three sections, showing the interconnection between various data entities.
So far, the set of features were presented based on the assumption that the information from various Maritime Law sources, which include their names, rules within them, and their formal representation has already been introduced and populated in the rule-base. However, users can enter all this information using additional features provided by the system. The screenshots in Figure 7.3 show menus on the top side, represented by ‘+’, and buttons that can be used to perform such operations. In addition to these features, the system offers other important functionalities for a Maritime Lawyer, which are stated below.
• It facilitates text-based searching, which can be used to search the rule-base by entering keywords or phrases, in addition to tag-based search mechanism.

• The selected set of rules, along with their textual as well as formal representations, can be exported as a file to other applications.

• A detailed report about the selected set of rules can be generated, which includes their source information and the two representations.

The system was tested using some sample queries, presented in Appendix B. The set of rules, provided in Appendix B, were entered in the system and tags were created manually and assigned to them based on their textual description. An example of a test case is presented that uses the same query which was used to illustrate the verification technique in Chapter 5. The referred query is stated below.

“What are the permissible actions that can be taken against a hostile act of pirate vessel located in territorial waters?” (Rephrased Query 5, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

Since the query refers to a situation in which a pirate vessel is encountered, a search was made in the rule-base by selecting the tag “Pirate Vessel”. The selected tag extracted all the set of rules that talk about various actions, whether permitted or prohibited, against the pirate vessel. A screenshot of the application showing the resulting rules, from different applicable law sources, is shown in Figure 7.4. The highlighted Maritime Law, shown in the figure, represents the source of the selected rule. A detailed report of the resulting set of rules was also generated, a portion of which is shown in Figure 7.5. It should be noted that the current implementation of the suite of tools uses “&” symbol for
the conjunctive connective instead of “∧”, which is used in the formal representation provided in Appendix B.

Figure 7.4: GUI of Maritime Law Management System Showing Search Results
1. Piracy

<table>
<thead>
<tr>
<th>Type</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual Form</td>
<td>It is prohibited to follow any suspected pirate ship into territorial waters unless owner of territorial waters has specifically requested U.S. assistance, and has given permission for U.S. vessel to enter their water, or pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.</td>
</tr>
<tr>
<td>ROE Source</td>
<td>Joint Exercise ROE</td>
</tr>
<tr>
<td>Date Added</td>
<td>Wednesday, September 24, 2008 2:22:59 PM</td>
</tr>
<tr>
<td>Modified</td>
<td>Tuesday, January 06, 2009 7:56:53 PM</td>
</tr>
</tbody>
</table>

Logical Representation:

\[
pirate\_vessel(V_1) \& \text{territorial\_water}(C_1) \& \text{position}(V_1,C_1) \& \text{entry\_request}(C_1) \& \text{permitted}(C_1) \Rightarrow \text{pursue}(V_1,C_1);
\]

\[
pirate\_vessel(V_1) \& \text{territorial\_water}(C_1) \& \text{position}(V_1,C_1) \& \text{hostile\_act}(V_1) \Rightarrow \text{pursue}(V_1,C_1);
\]

2. Search/Detain

<table>
<thead>
<tr>
<th>Type</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual Form</td>
<td>Only ships that have been declared pirate ships by the policing entity may be stopped/boarded.</td>
</tr>
<tr>
<td>ROE Source</td>
<td>Indonesian SCFA</td>
</tr>
<tr>
<td>Date Added</td>
<td>Wednesday, September 24, 2008 1:47:31 PM</td>
</tr>
<tr>
<td>Modified</td>
<td>Tuesday, January 06, 2009 7:43:11 PM</td>
</tr>
</tbody>
</table>

Logical Representation:

\[
pirate\_vessel(V_1) \Rightarrow \text{search}(V_1);
\]

Figure 7.5: Generated Report for the Selected Rules
7.3. **RULER (Rule Evaluation Routine)**

The technique for formal verification of a set of rules, selected from various Maritime Laws, has already been presented in Chapter 5. The behavioral analysis in it, which addresses portion of the verification problem, is performed by CPN Tools [35]. A utility named “RULER (Rule Evaluation Routine)” was developed in Java to implement the proposed verification technique. It uses an open source library, named Britney Suite [36], to interact with CPN Tools. The referred library is a set of functions written in Java that uses CPN Tools simulator, making it possible to integrate Petri Net simulation (i.e. Petri Net execution mechanism) and its state space analysis in Java applications. The implemented utility works by taking the file of the exported rule set (generated by Maritime Law Management System), the set of facts, and mutually exclusive concepts from the user and performs verification analyses using the given information, consequently, providing a report on identified inconsistent, redundant, incomplete, and useless rules. A Petri Net model is also generated by the application corresponding to the set of rules that can be used by the Rule Execution Engine, presented in the next section, to assist in decision-making process. The generation of Petri Net by RULER and its exportation to Rule Execution Engine is, however, transparent to the user. Figures 7.6, 7.7 and 7.8 show some of the screenshots of RULER, displaying some results from the implemented verification technique.
**Figure 7.6: Input Rules from Maritime Law Management System**
Figure 7.7: Results showing Identified Redundant Cases
Figure 7.8: Results showing Identified Inconsistent Cases

7.4. **RulEx (Rule Execution Engine)**

The reasoning mechanism, presented in Chapter 6, was incorporated into the suite by developing a reasoning engine known as “Rule Execution Engine (RulEx)”. This application makes use of the Britney Suite library [36] to perform simulation on the Petri Net model that was generated by RULER. Some of the screenshots of Rule Execution Engine are presented in Figures 7.9 and 7.10.

Figure 7.9 shows the results of rule execution process that derives new set of information based on the given facts. The interface consists of five sections in which Section 1 displays the set of rules triggered, Section 2 shows the entire set of facts (input
and resulting), Sections 3 and 4 correspond to the graphical representation of the rule set, and Section 5 presents the summary of the entire execution process.

Similarly, the screenshot in Figure 7.10 presents the process of identifying input conditions that support a certain action. The selected concept (i.e. P12) in the top-left panel of the interface corresponds to the action proposition, whereas the bottom-left panel displays all the identified set of input conditions that support it.
7.5. Conclusion

A suite of tools has been presented in this chapter that was developed to provide a computer-aided approach for the task of Maritime Lawyer. The suite consists of three separate applications which provide the functionalities of management and verification of Maritime Laws as well as decision-making from them. The functionalities were demonstrated with the help of some example queries and the results obtained by the software.
CHAPTER 8: CONCLUSION AND FUTURE WORK

This chapter concludes the thesis and suggests some directions for future work.

8.1. Conclusion

A computer-aided approach for providing a comprehensive support to a Maritime Lawyer is presented in this thesis. The approach will provide considerable help to him/her in performing his/her job, which is performed manually at present and is a challenging and strenuous activity due to the time critical nature of the task. The presented approach, being automated for the most part, overcomes these limitations. It makes use of several tools and techniques, some of which already exists, to address the defined problem. It was tested with a sample scenario to verify the degree of accomplishment of the specified goal. The tests, which have been presented in various parts of the thesis showed the achievement of most of the goals.

The proposed solution is not restricted to Maritime Law Analysis problem. The individual components of the presented approach, meant to address specific portions of the problem, can be used in other problem domains as well that require those specific functionalities. For example, the verification technique presented in this thesis can be used to address the general V&V problem for Production Systems, which has been addressed by several researchers in the past. Similarly, the reasoning mechanism presented in this thesis can be used as an alternative to existing reasoning engines. It is
capable of performing forward and backward reasoning, and provides visualization capabilities for both the processes, which can serve as an explanation tool to the user. Moreover, the two analytical techniques, which include verification methodology and reasoning mechanism, can also serve in domains that require both the functionalities together, such as the kinds of problem addressed in this thesis.

8.2. Future Work

It has already been mentioned that the proposed solution accomplishes most of the defined goal. However, there exists some room for improvements and enhancements. The rest of the section discusses some of the possible future work.

The solution proposed in this thesis performs reasoning using production rules. An obvious future step is to incorporate an ontological representation of the domain information for the reasoning mechanism that will help derive new information based on semantic relationships among domain concepts. Since semantic knowledge is not explicitly defined within the rules, enhancements are required to the proposed solution to employ a formalism that provides such a reasoning capability. For example, consider a set of rules that talk about piracy, some of which are defined using the terminology of “pirate vessel” and the rest use the vocabulary of “pirate ship”. Since the two concepts are related to one another (second one being the special case of the first one), and such an information is also not defined in the rule-base, it is not possible to make use of all the rules about piracy in situations when one of the two concepts is known to be true. Such a capability can, however, be achieved by making use of an ontology that defines the relationships between various concepts, and an ontology reasoner that derives new
information based on these defined relationships. Moreover, the use of an ontology can also help to explicitly represent additional knowledge about Maritime Laws e.g. regions of their applicability, missions to which they apply etc. Since such information is also not defined in rules, the ontology can help to encode such knowledge and provide reasoning capabilities over it. Such an enhancement will facilitate the processing of the following types of queries:

“What is the OPLAW applicable to U.S. Forces during this joint exercise?”
(Query 1, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

“What are the important changes in ROE as U.S. forces depart to carry out Operation Good Samaritan?” (Query 2, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

“How does the transition across the equator and out of the Indonesian SOFA region change ROE?” (Query 3, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

The analytical techniques, presented in this thesis, assume that the formal representation of the set of rules is already available. However, such a representation does not exist, in actual, for Maritime Laws and needs to be constructed manually by transforming the rules, expressed in the natural language, into a machine-readable format. Same procedure was adopted, in this thesis, by manually transforming Maritime Laws (provided in Appendix B) to a formal representation, for the verification and
demonstration of various analytical techniques. A possible enhancement can be to provide a methodology that performs or assists in performing the translation task.

Finally, the verification technique presented in this thesis did not address the case of subsumed rules within Maritime Laws. A possible future step is to provide a formal approach for the verification of subsumption in the formally represented set of rules.
This appendix presents a brief introduction to temporal logic, linear time logic and computation tree logic. The discussion that follows has been compiled from [37] with minor editorial changes.

A.1. Temporal Logic

The concept that makes temporal logic different from propositional and predicate logic is that a formula in the former one is not statically true or false in a model, as opposed to the later ones. The model of temporal logic has several states and a formula can be true in some states and false in others. In case of propositional and predicate logic, a formula is either true or false in a model and it stays that way. Thus, the static notion of truth value is replaced by a dynamic one in temporal logic. In model checking, systems are modeled (or represented) as state-transition systems and properties are expressed as temporal logic formulae to verify if they exist in the model as the system evolves from state to state. There are many classes of temporal logic that have been proposed. However, they can all be divided into two classes based on their view of time: linear-time logic (or LTL) and branching-time logic.
A.2. Linear-time Temporal Logic (LTL)

LTL is a class of temporal logic, which models time as a sequence of states, going infinitely into the future. In other words, it views time as a path.

An LTL formula is defined by the following Backus Naur form:

$$\phi ::= p \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \to \phi) \mid (X \phi) \mid (F \phi) \mid (G \phi) \mid (\phi \cup \phi) \mid (\phi \circ \phi)$$

where $p$ represents any atomic formula e.g. *The traffic light is green*, *The printer is idle*, *The switch is on*. The symbols $\land$, $\lor$, $\to$ are logical connectives whereas $X$, $F$, $G$, $U$ and $R$ are temporal connectives or temporal operators, which will be described in the following discussion.

It has already been mentioned that the systems to be verified by model checking methodology are represented as state-transition systems.

**Definition** (State-transition System):- A state-transition system is defined by a set of states, a relation describing how the system moves from one state to another and a mapping which assigns a truth value (*True/False*) to every atomic formula within a state. Mathematically, a state-transition system $\mathcal{M}$ can be expressed as:

$$\mathcal{M} = (S, \rightarrow, L)$$

where $S$ is the set of states, $\rightarrow$ is the relation among the states in $S$ and $L$ is the assignment of truth values to the atomic formulae in every $s \in S$.

Graphically, such a system can be represented by a directed graph whose nodes are the states in $S$ containing all the atomic formulae that are true and the arcs are the transition relation. An example of a state-transition system is provided in Figure A.1,
which consists of four states \((s_0, s_1, s_2 \text{ and } s_3)\) corresponding to transition relation \((s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_3, s_2 \rightarrow s_3)\) and mapping defined as \(L(s_0) = \{p, q\}, L(s_1) = \{p, r\}, L(s_2) = \{q, r\} \text{ and } L(s_3) = \{r\}\).

![Diagram of state-transition system](image)

**Figure A.1: A directed graph representing state-transition system**

**System Verification Using LTL**

Let \(M\) be the model of state-transition system. Given \(\pi = s_0 \rightarrow s_1 \rightarrow \ldots\) as a path in \(M\), whether an LTL formula is satisfied by \(\pi\) is defined by the relation as follows:

- \(\pi \models p \iff p \in L(s_0)\).
- \(\pi \models \neg \phi \iff \pi \not\models \phi\).
- \(\pi \models \phi_1 \land \phi_2 \iff \pi \models \phi_1 \text{ and } \pi \models \phi_2\).
- \(\pi \models \phi_1 \lor \phi_2 \iff \pi \models \phi_1 \text{ or } \pi \models \phi_2\).
- \(\pi \models \phi_1 \rightarrow \phi_2 \iff \pi \models \phi_2 \text{ whenever } \pi \models \phi_1\).
- \(\pi \models X \phi \iff \pi^2 \models \phi\), where \(\pi^2 = s_1 \rightarrow s_2 \rightarrow \ldots\) is the sub-path of \(\pi\) starting from \(s_1\).

Thus, \(X\) means: ‘\(\phi\) holds in next state’.

- \(\pi \models G \phi \iff \text{for all } i \geq 1, \pi^i \models \phi\). Thus, \(G\) means: ‘\(\phi\) holds in all future states’.
• $\pi \models F \phi$ iff there is some $i \geq 1$ such that $\pi^i \models \phi$. Thus, $F$ means: ‘$\phi$ holds in some future state’.

• $\pi \models \phi_1 U \phi_2$ iff there is some $i \geq 1$ such that $\pi^i \models \phi_2$ and for all $j=1,\ldots,i-1$ it is the case that $\pi^j \models \phi_1$. Thus, $U$ means: ‘$\phi_1$ holds until $\phi_2$ starts to hold’.

• $\pi \models \phi_1 R \phi_2$ iff either there is some $i \geq 1$ such that $\pi^i \models \phi_1$ and for all $j=1,\ldots,i$ it is the case that $\pi^j \models \phi_2$, or for all $k \geq 1$ it is the case that $\pi^k \models \phi_2$. Thus, $R$ says: ‘$\phi_2$ holds until and including the point where $\phi_1$ starts to hold. If $\phi_1$ does not hold in a future state, then $\phi_2$ will hold forever’.

An interpretation of some of the above mentioned temporal operators is depicted graphically in Figures A.2-A.6.

---

**Figure A.2:** A state-transition system satisfying $X \phi$

**Figure A.3:** A state-transition system satisfying $G \phi$

**Figure A.4:** A state-transition system satisfying $F \phi$
Figure A.5: A state-transition system satisfying $\phi_1 U \phi_2$

Figure A.6: State-transition systems satisfying $\phi_1 R \phi_2$

**A.3. Branching-time Logic**

In previous section, it was shown that an LTL formula is interpreted over a path. It is also possible to consider a set of paths using LTL, however, an LTL formula quantifies universally over all possible set of paths i.e. an LTL formula has to be true for all the paths considered. Therefore, properties that assert the existence of a path cannot be expressed in LTL. Branching-time logics solve this problem by providing quantification over the set of paths. This section presents a branching-time logic known as Computation Tree Logic (CTL), describing temporal operators it offers in addition to the ones presented in previous section.
A.3.1. Computation Tree Logic (CTL)

CTL is a branching-time logic, which means that model of time is a tree like structure in which the future is not determined. In other words, there are multiple paths in future anyone of which can be the actual path.

A CTL formula is defined by the following Backus Naur form:

\[ \phi ::= p \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \rightarrow \phi) \mid AX \phi \mid EX \phi \mid AF \phi \mid EF \phi \mid AG \phi \mid EG \phi \mid A[U] \phi \mid E[U] \phi \]

It introduces two new operators A (for all futures) and E (for some future) in addition to the ones presented in previous section. It should be noticed that every CTL temporal connective is composed of one universal quantifier (A or E) and one path quantifier (F, G, X or U). The universal quantifiers are used to make statements that range over all possible paths in future and path quantifiers are used to make statements that range over all moments of time along a particular path [38].

System Verification Using CTL

Let \( M \) be the model of state-transition system and \( s \) be any state in \( M \). Whether a CTL formula \( \phi \) holds in \( s \) is defined by the relation as follows:

- \( M, s \models p \) iff \( p \in L(s) \).
- \( M, s \models \neg \phi \) iff \( M, s \not\models \phi \).
- \( M, s \models \phi_1 \land \phi_2 \) iff \( M, s \models \phi_1 \) and \( M, s \models \phi_2 \).
- \( M, s \models \phi_1 \lor \phi_2 \) iff \( M, s \models \phi_1 \) or \( M, s \models \phi_2 \).
- \( M, s \models \phi_1 \rightarrow \phi_2 \) iff \( M, s \not\models \phi_1 \) or \( M, s \models \phi_2 \).
\[ M, s \models \text{AX } \phi \text{ iff for all } s_1 \text{ such that } s \rightarrow s_1 \text{ we have } M, s_1 \models \phi \]. Thus, AX says: ‘in every next state’.

\[ M, s \models \text{EX } \phi \text{ iff for some } s_1 \text{ such that } s \rightarrow s_1 \text{ we have } M, s_1 \models \phi \]. Thus, EX says: ‘in some next state’.

\[ M, s \models \text{AG } \phi \text{ holds iff for all paths } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \text{ where } s_1 \text{ equals } s \text{ and all } s_i \text{ along the path, we have } M, s_i \models \phi \]. Thus, it says: ‘for all paths beginning in } s \text{ the property } \phi \text{ holds globally’}.

\[ M, s \models \text{EG } \phi \text{ holds iff there is a path } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \text{ where } s_1 \text{ equals } s \text{ and for all } s_i \text{ along the path, we have } M, s_i \models \phi \]. Thus, it says: ‘there exists a path beginning in } s \text{ such that } \phi \text{ holds globally along it’}.

\[ M, s \models \text{AF } \phi \text{ holds iff for all paths } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \text{ where } s_1 \text{ equals } s, \text{ there is some } s_i \text{ such that } M, s_i \models \phi \]. Thus, it says: ‘for all paths beginning in } s \text{ there will be some future state where } \phi \text{ holds’}.

\[ M, s \models \text{EF } \phi \text{ holds iff there is a path } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \text{ where } s_1 \text{ equals } s, \text{ and for some } s_i \text{ along the path } M, s_i \models \phi \]. Thus, it says: ‘there exists a path beginning in } s \text{ such that } \phi \text{ holds in some future state in that path’}.

\[ M, s \models \text{A[} \phi_1 \text{ U } \phi_2 \text{]} \text{ holds iff for all paths } s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \text{ where } s_1 \text{ equals } s, \text{ the path satisfies } \phi_1 \text{ U } \phi_2, \text{ i.e. there is some } s_i \text{ along the path, such that } M, s_i \models \phi_2 \text{ and for each } j<i, \text{ we have } M, s_j \models \phi_1 \]. Thus, it says: ‘all paths beginning in } s \text{ satisfy that } \phi_1 \text{ Until } \phi_2 \text{ holds on it’}.
$M, s \models E[\phi_1 \mathbf{U} \phi_2]$ holds iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, and the path satisfies $\phi_1 \mathbf{U} \phi_2$. Thus, it says: 'there exists a path beginning in $s$ such that $\phi_1$ Until $\phi_2$ holds on it'.

An interpretation of some CTL formulae, using some of the temporal connectives presented above, is shown graphically in Figures A.7-A.12.

![Figure A.7: A state-transition system whose starting state satisfies AF $\phi$](image-url)
Figure A.8: A state-transition system whose starting state satisfies AG $\phi$

Figure A.9: A state-transition system whose starting state satisfies EF $\phi$
Figure A.10: A state-transition system whose starting state satisfies EG \( \phi \)

Figure A.11: A state-transition system whose starting state satisfies AX \( \phi \)
Figure A.12: A state-transition system whose starting state satisfies $E[\phi_1 U \phi_2]$
APPENDIX B

This appendix presents a hypothetical maritime scenario [21], using a small set of Operational Laws (OPLAWS). The OPLAWS are Maritime Laws used by the U.S. Navy. The scenario also includes some sample queries that a Maritime Lawyer, in this case Staff Judge Advocate, may be interested in.

B.1. Scenario Overview

In this scenario, the United States is conducting a multi-force joint exercise in Indonesia. When a large earthquake occurs in the Philippines, U.S. forces cancel their exercise and provide Humanitarian-Assistance and Disaster-Relief to the affected people. Staff Judge Advocates are aboard the USS Temolu (LHA 1), command ship of the U.S. forces, advising the commander. The Temolu is moored in Jakarta’s Tanjung Priok Port for the multi-force exercise. Upon receiving orders for the relief mission, she travels north out of the Java Sea, towards the South China Sea, with a fleet comprised of various surface and amphibious ships. A vessel is encountered near Malaysia that commits a hostile act against them, and the U.S. commander must decide how to respond based on all the laws that are applicable in that region.
B.2. **Sample Rules**

Below are some OPLAWS applicable in the region, each of which contains a set of rules relevant to the scenario.

**Sofa for Indonesia**

1. **Biometric Data.**
   
   No biometric data is to be gathered without prior oral consent from individuals of interest.

2. **Search/Detain**

   Only ships that have been declared pirate ships by the policing entity are allowed to be searched.

3. **Non-compliant ships.**

   Labeling a ship as non-compliant does not authorize U.S. forces to search vessel.

**Joint Exercise ROE**

1. **Piracy**

   It is prohibited to follow any suspected pirate ship into territorial waters unless owner of territorial waters has specifically requested U.S. assistance, and has given permission for U.S. vessel to enter their water, or pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.
**Operation Good Samaritan ROE**

1. Piracy
   a. A ship shall be considered a pirate vessel if and only if hostile acts or hostile intents against U.S. forces are directly observed by U.S. forces.
   b. Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.

**SEACOM ROE**

1. Piracy
   It is not permitted to pursue a pirate ship into territorial waters unless express consent has been given by the coastal state prior to entry into their territorial waters, unless the hostile act was committed against U.S. forces, U.S. property, or committed in U.S. territorial waters.

2. Search
   U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.

3. Biometric Data
   a. The collection of biometric data is authorized for ships declared to be pirate ships or defined as non-compliant.
   b. The collection of biometric data is authorized for all other ships, per the consent of the individuals of interest.
B.3. Formal Representation

This section provides a logical representation for the set of rules mentioned above. The logical representation was not a part of the scenario [21] and was created manually for various analyses proposed in this thesis.

Sofa for Indonesia

1. normal_vessel(V1) \land biometric_data(D1) \land permitted(P1) \rightarrow collect(D1)

2. pirate_vessel(V1) \rightarrow search(V1)

3. non_compliant_vessel(V1) \rightarrow not_search(V1)

Joint Exercise ROE

1. (a) pirate_vessel(V1) \land territorial_water(C1) \land position(V1, C1) \land entry_request(C1) \land permitted(C1) \rightarrow pursue(V1, C1)

   (b) pirate_vessel(V1) \land territorial_water(C1) \land position(V1, C1) \land hostile_act(V1) \rightarrow pursue(V1, C1)

Operation Good Samaritan ROE

1. (a) normal_vessel(V1) \land hostile_act(V1) \rightarrow pirate_vessel(V1)

   pirate_vessel(V1) \rightarrow normal_vessel(V1) \land hostile_act(V1)

   (b) pirate_vessel(V1) \land territorial_water(C1) \land position(V1, C1) \land not_pursue(V1, C1)

SEACOM ROE

1. (a) pirate_vessel(V1) \land territorial_water(C1) \land position(V1, C1) \land entry_request(C1) \land permitted(C1) \rightarrow pursue(V1, C1)
(b) \(\text{pirate_vessel}(V1) \land \text{territorial_water}(C1) \land \text{position}(V1, C1) \land \text{hostile_act}(V1)\) → pursue(V1, C1)

(c) \(\text{pirate_vessel}(V1) \land \text{territorial_water}(US) \land \text{position}(V1, US) \land \text{hostile_act}(V1)\) → pursue(V1, US)

2. \(\text{pirate_vessel}(V1) \rightarrow \text{search}(V1)\)

3. (a) \(\text{pirate_vessel}(V1) \land \text{biometric_data}(D1) \rightarrow \text{collect}(D1)\)

\(\text{non_compliant_vessel}(V1) \land \text{biometric_data}(D1) \rightarrow \text{collect}(D1)\)

(b) \(\text{normal_vessel}(V1) \land \text{biometric_data}(D1) \land \text{permitted}(P1) \rightarrow \text{collect}(D1)\)

**B.4. Shorthand Representation of Formal Rules**

The long propositional statements of machine-readable rules, presented in the previous section, are represented by a shorthand representation using symbols. The symbols and their corresponding propositions are listed in Table B.1. These symbols will be used in the entire thesis instead of their corresponding propositions.

**Table B.1: Set of Propositions Used in the Formal Representation of Maritime Laws**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>normal_vessel(V1)</td>
</tr>
<tr>
<td>P2</td>
<td>biometric_data(D1)</td>
</tr>
<tr>
<td>P3</td>
<td>permitted(P1)</td>
</tr>
<tr>
<td>P4</td>
<td>collect(D1)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>P5</td>
<td>pirate_vessel(V1)</td>
</tr>
<tr>
<td>P6</td>
<td>search(V1)</td>
</tr>
<tr>
<td>P7</td>
<td>not_search(V1)</td>
</tr>
<tr>
<td>P8</td>
<td>non_compliant_vessel(V1)</td>
</tr>
<tr>
<td>P9</td>
<td>territorial_water(C1)</td>
</tr>
<tr>
<td>P10</td>
<td>position(V1, C1)</td>
</tr>
<tr>
<td>P11</td>
<td>entry_request(C1)</td>
</tr>
<tr>
<td>P12</td>
<td>permitted(C1)</td>
</tr>
<tr>
<td>P13</td>
<td>pursue(V1, C1)</td>
</tr>
<tr>
<td>P14</td>
<td>hostile_act(V1)</td>
</tr>
<tr>
<td>P15</td>
<td>not_pursue(V1, C1)</td>
</tr>
<tr>
<td>P16</td>
<td>territorial_water(US)</td>
</tr>
<tr>
<td>P17</td>
<td>position(V1, US)</td>
</tr>
<tr>
<td>P18</td>
<td>pursue(V1, US)</td>
</tr>
</tbody>
</table>
B.5. Sample Queries

Following are some sample queries that will be of interest to a Staff Judge Advocate officer.

1. What is the OPLAW applicable to U.S. Forces during this joint exercise?

2. What are the important changes in ROE as U.S. forces depart to carry out Operation Good Samaritan?

3. How does the transition across the equator and out of the Indonesian SOFA region change ROE?

4. Are there any restrictions as to where U.S. forces are allowed to moor their vessels?

5. Is it allowed to follow a pirate vessel in territorial waters, if it commits a hostile act?

6. Is it permitted to stop and search a vessel that commits a hostile act?

7. Is it allowed to collect biometric data of individuals boarding a vessel?
APPENDIX C

A formal description of the ASK-CTL formulae to capture redundant and inconsistent cases has been presented in Chapter 5. This appendix provides a description concerning the implementation of the formally described ASK-CTL formulae. It presents the pseudo codes of two SML programs (in Pascal style code); which are, in actual, implementations of the two ASK-CTL formulae and the state functions within them. The state functions, as mentioned in Chapter 5, are symbols in italics within the parentheses of both the defined ASK-CTL formulae. They require an explicit definition within a program to check certain properties in the states of an Occurrence Graph. The two pseudo codes of programs provide definitions for all the state functions used in the defined ASK-CTL formulae (5.1 and 5.2), each meant for the verification of a specific property within a state. For example, “function hasMultipleOutArcs” in Program 2 (below) corresponds to the expression $a_1$ of formula 5.2, which is defined to verify the property of multiple arcs going out from a state of an OG. Some pre-defined SML functions are also available in CPN Tools [39], to check some basic properties within the individual states of an Occurrence Graph, which can be used as state functions. However, the available set of functionalities in [39] does not cover verification of complex properties within a state, which require an explicit definition of these complex properties using the basic ones available. The pseudo code of the two programs (below) highlights all such pre-defined
functions that are used in state functions by indicating them in comments. The comments in the following code are statements enclosed within braces { and }. The rest of the section presents pseudo codes for the two SML programs: Program 1 and Program 2. The input to Program 1 is a list of mutually exclusive concepts. For each set in the list, the ASK-CTL formula is executed, which verifies the property of inconsistency defined by the set of state functions within it. The only addition to ASK-CTL expression in the pseudo code is the use of keyword “NF” which is a pre-defined function in ASK-CTL library that handles call and response from the state functions. The resulting set of rules leading to any identified inconsistency is displayed as an output. Similarly, Program 2 implements the case of redundancy which verifies the said property within an OG and displays the set of rules leading to such a case.

(a) **Program 1 (Verification of Inconsistent Rules)**

```sml
var stateOne, stateTwo : integer;
var conceptOne, conceptTwo : string;

function main (var mutuallyExclusiveSet : array[1..n][1..2] of string)
    var askctlResult : boolean;
    for i := 1 to n
        conceptOne := mutuallyExclusiveSet[i][1];
        conceptTwo := mutuallyExclusiveSet[i][2];
        askctlResult := AND(POS(NF("-", checkMutuallyExclusiveConceptOne)),
            POS(NF("-", checkMutuallyExclusiveConceptTwo)));
```

{ ASK-CTL expression in SML for inconsistency check that returns true/false. The functions AND, POS and NF are pre-defined in ASK-CTL library, where NF is the call to the function declared within}
it having the current state number as its argument value

if askctlResult = true
    if stateOne = 1
        print “Contradiction identified with original facts”;
    else
        displayRule(stateOne, conceptOne);
    endif

    if stateTwo = 1
        print “Contradiction identified with original facts”;
    else
        displayRule(stateTwo, conceptTwo);
    endif
endif
endfor
endmain

function checkMutuallyExclusiveConceptOne (var stateNumber : integer)
    if val(conceptOne, stateNumber) >= 1
        {val represents a pre-defined function in CPN SML to get the marking of conceptOne within the stateNumber}
        stateOne := stateNumber;
        return true;
    else
        return false;
    endif
endcheckMutuallyExclusiveConceptOne

collection function checkMutuallyExclusiveConceptTwo (var stateNumber : integer)
    if val(conceptTwo, stateNumber) >= 1
        stateTwo := stateNumber;
        return true;
    else
        return false;
    endif
endcheckMutuallyExclusiveConceptTwo

function displayRule (var stateNumber : integer, var concept : string)
    var parentNodes : list;
    parentNodes := InNodes(stateNumber);
    {InNodes is a pre-defined function in CPN SML}
for i := 1 to length(parentNodes)
    if val(concept, parentNodes[i]) = 0
        print ArcsInPath(parentNodes[i], stateNumber);  \{ArcsInPath is a pre-defined function in CPN SML\}
    endif
endfor
enddisplayRule

(b) Program 2 (Verification of Redundant Rules)

var occurrenceSequenceList, redundantRulesList, processedNodesList : list;
var stateOne : integer;

function main ()
do
    occurrenceSequenceList := null;
    stateOne := 0;

    askctlResult := POS(AND(NF("-", hasMultipleOutArcs),
        POS(AND(NF("-", hasMultipleReachableInArcs),
            NF("-", hasDistinctPath)))));

        \{ASK-CTL expression in SML for redundancy check that returns true/false. The functions AND, POS and NF are pre-defined in ASK-CTL library, where NF is the call to the function declared within it having the current state number as its argument value\}

    if askctlResult = true
        print redundantRulesList;
    endif
    while askctlResult = true;
endmain

function hasMultipleOutArcs (var stateNumber : integer)
    if length(OutArcs(stateNumber))>1
        \{OutArcs is a pre-defined function in CPN SML\}
        stateOne := stateNumber;
        return true;
    else
stateOne := 0;
return false;
endif
dendhasMultipleOutArcs

function hasMultipleReachableInArcs (var stateNumber : integer)
var parentNode : list;
if existsInList(processedNodesList, [stateOne, stateNumber])
   return false;
else
   if length(InArcs) > 1 {InArcs is a pre-defined function in CPN SML}
      parentNode := InNodes(stateNumber); {InNodes is a pre-defined function in CPN SML}
      for i := 1 to length(parentNode)
         if Reachable(stateOne, stateNumber) = false {Reachable is a pre-defined function in CPN SML}
            return false;
         endif
      endfor
      return false;
   endif
endfor
else
   return true;
endif
dendhasMultipleReachableInArcs

function hasDistinctPath (var stateNumber : integer)
var distinctPath : boolean;
depthFirstTraversal(stateOne, stateNumber);
redundantRulesList := null;
for i := 1 to length(occurrenceSequenceList)
distinctPath := false;
for j := i+1 to length(occurrenceSequenceList)
   if hasCommonElements(occurrenceSequenceList[i], occurrenceSequenceList[j]) = true
      distinctPath := false;
      break;
   else
      distinctPath := true;
      break;
   endif
endfor

128
distinctPath := true;
endif
endfor
if distinctPath = true
    redundantRulesList := append(redundantRulesList, occurrenceSequenceList[i]);
endif
endfor
if length(redundantRulesList) > 1
    processedNodesList := append(processedNodesList, [stateOne, stateNumber]);
endif
endhasDistinctPath

function depthFirstTraversal (var startNode : integer, var endNode : integer)
var outgoingarcs, occurrenceSequence : list;
outgoingarcs := OutArcs(startNode); {OutArcs is a pre-defined function in CPN SML}
for i := 1 to length(outgoingarcs)
    if DestNode(outgoingarcs[i]) = endNode {DestNode is a pre-defined function in CPN SML}
        occurrenceSequence := append(occurrenceSequence, outgoingarcs[i]);
        occurrenceSequenceList := append(occurrenceSequenceList, [occurrenceSequence]);
    else
        if Reachable(DestNode(outgoingArcs[i]), endNode)
            occurrenceSequence := append(occurrenceSequence, outgoingarcs[i]);
            depthFirstTraversal(DestNode(outgoingArcs[i]), endNode);
        endif
    endif
endfor
enddepthFirstTraversal

function hasCommonElements (var list1 : list, var list2 : list)
for i := 1 to length(list1)
    for j := 1 to length(list2)
        if list1[i] = list2[j]
            return true;
        endif
    endfor
endfor
endhasCommonElements
endif
endfor
endfor
endfor

return false;
endhasCommonElements
REFERENCES
REFERENCES


[34] “Zotero”, Center for History and New Media (CHNM) at George Mason University (GMU), http://www.zotero.org/


Muzammil Sagheer received his Bachelor of Science (major in Computer Science) degree in 2004 from Mohammad Ali Jinnah University, Karachi, Pakistan. He served as an Undergraduate Teaching Assistant in the courses of Computer Programming and Compiler Construction at Mohammad Ali Jinnah University and later worked as a Software Developer for two years at Infinilogic (Pvt.) Ltd., Karachi, Pakistan. Nowadays, he is working as a Graduate Research Assistant at System Architectures Laboratory, George Mason University.